Prvi kolokvij

Wednesday, April 26, 2023 4:20 PM

```
from sklearn import datasets
baza_sklearn=datasets.load_iris()
        Ili       baza=datasets.fetch_olivetti_faces()
baza_faces=datasets.fetch_openml('olivetti_faces')
```

.keys()	- što je unutra
.data	- podaci, spremamo u X
.target	- klase, spremamo u y --> moramo imati X i y za train i test
.target_names	- nazivi tih klasa
.DESCR	
.shape	- stupaca i redaka

```
podaci[35:60]+=60  #od 35 do 60 se povećavaju za 60
```

Import

```
import numpy as np
import matplotlib.pyplot as plt
```

X=np.array(baza_faces.data)	- dohvaća podatke i odmah ih pretvara u array

Za sliku

```
#prikazati od 10 do 30 svaku drugu (1.način)   (ne treba  i+1 to automatski)
for i,podaci in enumerate(X[9:29:2],start=1):       - krećemo od jedan jer u plotu nema pozicije 0!          [početak: kraj: korak]
  plt.subplot(2,5,i)                                 - 2 retka, 5 stupaca
  slika=podaci.reshape(64,64)                        - reshapa sliku
  plt.imshow(slika, cmap="gray")
```

Random brojevi

```
#20 cijelih brojeva u rasponu od 5 do 30 (najmanji,najviši,size)
        podaci=np.random.randint(5,30,20)            - cijeli brojevi
                                (od : do : koliko)
        b=np.random.uniform(40,100,10)       - decimalni

# 50 brojeva u 2D koord, random cijele u intervalu od 1 do 500
        np.random.randint(1,500,(50,2))     - dimenzija tj broj stupaca
        np.random.rand(20,2)  #20 u dvije dimenzije, decimalni
```

PLOT

```
# u variajblu x5 spremiti 50 cijelih brojeva u 5D (raspon 20 do 250) i onda vizualizirati u koord
  2. dim kao x a 4. Dim kao y
        podaci=np.random.randint(20,250,(50,5))
        plt.scatter(podaci[:,2],podaci[:,4])  d
```

.scatter(x2,y2,marker='v',color='r', label='x2y2')	
.legend()	- da pokaže labele
.title('naslov')	
.show() .imshow(ime_varijable_za_prikazati)	- da ulovi sve naredbe prije
.xlabel('naziv x osi')	- naziv x osi
.xlim(150,200)	- koji opseg na x da pokaže

Nenadzirano učenje

1.K-means grupiranje

```
from sklearn.cluster import KMeans

kmeans=KMeans(n_clusters=2, random_state=42, n_init=1)     #n_init je koliko puta da izvrti
kmeans.fit(podaci)   #model fitamo na podacima
grupa=kmeans.predict(podaci)   # kako bi posložio naše podatke
kmeans.predict([[297,333]])   #nova točka, gdje bi ju stavio

centroidi=kmeans.cluster_centers_  #koordinate centroida
```

Plot

```
plt.scatter(podaci[:,0],podaci[:,1], c=grupa, cmap='cool')     #u c (color) definiramo što da stavlja različitih boja i onda cmap na koji način boja
```

```
plt.scatter(centroidi[:,0], centroidi[:,1], marker='x', color='r', s=200)   # da pokaže centroide
```

Silhouette analysis

```
# -1 znači krivo tj.loše clusterizirano, 0 na granici je kojem clusteru pripada, +1 u redu
    from sklearn import metrics
    score=metrics.silhouette_score(podaci, grupa, metric='euclidean')
    print('Silhouette score: %.3f' %score)
```

Elbow

```
from yellowbrick.cluster import KElbowVisualizer
 visualizer=KElbowVisualizer(kmeans, k=(2,11), locate_elbow=True)
 visualizer.fit(podaci)
 visualizer.show()
```

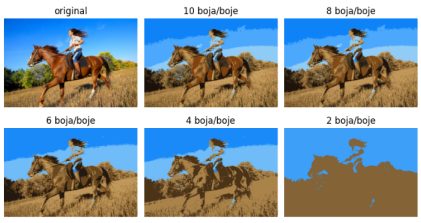
```
    #definiranje fije koja računa score i plota za različite brojeve clustera počevši od a do b, za dane podatke
    def vise_clustera(a,b,podaci):    #najveći a i najmanji b br. i podaci
     max_score=0
     broj_clustera=0  #da nam pamti koliko ih ima najveci score
     scores_po_broju=[]  #za sad prazno
     broj_clustera_po_broju=[] #spremljeno redom da bude i jedno i drugo
     for i in range(a,b+1):
      kmeans=KMeans(n_clusters=i, random_state=42, n_init=1)
      grupa=kmeans.fit_predict(podaci)
      score=metrics.silhouette_score(podaci, grupa, metric='euclidean')
      print('Silhouette score: %.3f' %score)
      scores_po_broju.append(score)  #spremili vrijednosti da ne nestanu kad nađe najveću
      broj_clustera_po_broju.append(i)
      if score>max_score:
       max_score=score
       broj_clustera=i
     print('najveci score imamo sa {} clustera i iznosi {}'.format(broj_clustera, '%.3f' %max_sco    re))  #%.3f na tri decimale
     plt.plot(broj_clustera_po_broju, scores_po_broju)
     plt.show()
```

1.1.K-means grupiranje – pikseli, segmentacija (slika konj)

```
X=image.reshape(-1,3)   #sliku reshape da dobimo 3D vektor
kmeans.fit(X)
kmeans.predict(X)
segmented_img=kmeans.cluster_centers_[kmeans.labels_]  #segmentirana slika spremljena u varijablu, svaki piksel ima vrijednost nekog od centroida (3 centroida - 3 b
segmented_img=segmented_img/255  #dobimo raspon od 0 do 1 tj pretvaramo u float
slika_boje=segmented_img.reshape(image.shape)  #segmented image reshapati da ima dimenzije početne slike
plt.imshow(slika_boje)  #vidimo da je svakom pikselu dodijelio jednu od 3 boje tj. clustera
```

For petlja – slike za različiti broj clustera

```
#for petlja koja radi kmeans za sve te boje i sve te slike sprema u segmented_imgs
segmented_imgs=[]  #definiramo prazno
n_colors=(10,8,6,4,2)  #prvo da u 10 klustera, pa u 8,... koji su u varijabli ncolors
for n_clusters in n_colors:
 kmeans=KMeans(n_clusters=n_clusters, n_init=1, random_state=42).fit(X)
 segmented_img=kmeans.cluster_centers_[kmeans.labels_]  #kad se fita moramo dohvatiti  centrc
 segmented_img=segmented_img.astype(int)
 segmented_imgs.append(segmented_img.reshape(image.shape)) #spremimo u listu gdje će se sp

plt.figure(figsize=(10,5))  #dimenzije plota?
plt.subplots_adjust(wspace=0.05, hspace=0.15) #razmak između slika

plt.subplot(2,3,1)
plt.imshow(image)
plt.title("original")
plt.axis('off')    #ovo je sve za prvu sliku a for petlja za ostale subplotove

for i,n_clusters in enumerate(n_colors):  #da prođe kroz sve boje
 plt.subplot(2,3,2+i)    #krenemo od 2. pozicije
 plt.imshow(segmented_imgs[i])
 plt.title("{} boja/boje".format(n_clusters))
 plt.axis('off')

plt.show()
```



Hijerarhijsko grupiranje - dendrogram

```
    #dendrogram tj. prikz stabla radi na principu udaljenosti - više grane su udaljeniji clusteri
    from scipy.spatial.distance import pdist
    udaljenosti=pdist(podaci)  #nađe udaljenosti naših podataka i spremi u udaljenosti
    from scipy.cluster.hierarchy import linkage
    stablo=linkage(udaljenosti)
    from scipy.cluster.hierarchy import dendrogram
    dendrogram_crtanje=dendrogram(stablo)
```

Rezanje po udaljenosti

```
    from scipy.cluster.hierarchy import fcluster
    poudaljenosti=fcluster(stablo,6,criterion='distance')  #režemo stablo gdje ima vrijednost 6
    plt.scatter(podaci[:,0], podaci[:,1], c=poudaljenosti, cmap="cool") #boja ih po parametru udaljenost
```

Rezanje po broju clustera

```
pobrojuclustera=fcluster(stablo, 3, criterion='maxclust')  #broj 3 znači koliko clustera
plt.scatter(podaci[:,0], podaci[:,1], c=pobrojuclustera, cmap="cool")
```

2.Linearna regresija

```
#dobivanje pravca, npr. kako visina(nezavisna varijabla, x os) utječe na težinu(zavisna, y os)
#metoda najmanjih kvadrata(pravac da su pogreške najmanje tj.praktikum lol)
from sklearn import datasets
baza=datasets.fetch_openml('bodyfat')
podaci=np.array(baza.data)
#dohvatimo visinu i tezinu (2 i 3 pozicija)
tezina=podaci[:,2]  #zavisna
visina=podaci[:,3]   #nezavisna varijabla
X=visina
y=tezina
```

Model linearne regresije

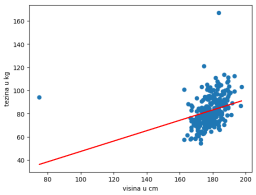
```
X=X.reshape(-1,1)  # samo doda drugi stupac,prazan ali algritam onda ima 2D
from sklearn.linear_model import LinearRegression
reg=LinearRegression().fit(X,y)
reg.coef_  #koeficijent tj nagib pravca
reg.intercept_  #gdje sječe x os
y_predviđeno=reg.predict(X)
reg.score(X,y)  # koliko % možemo objasniti lin regresijom 0,09 --> 9%
```

Rezidualno odstupanje

```
visina_21=X[20,0]  #za 21.osobu 1.stupac
stvarna_tezina_21=y[20]
dobivena_tezina_21=round(y_predviđeno[20],2)  #zaokruziti na dvije decimale
print("21. osoba u bazi je visoka {} cm, stvarna tezina je {} kg a tezina dobivena regresijkim modelom je {}.".format(visina_21, stvarna_tezina_21, dobivena_tezina_
rezidualno_odstupanje_21=round(abs(stvarna_tezina_21-dobivena_tezina_21),2)
```

Varijanca

```
#prosječno kvadratno odstupanje, što manja to bolje
from sklearn.metrics import mean_squared_error
mean_squared_error(y,y_predviđeno)  #između stvarnih (y) i pedvić
reg.predict([[180]]) #npr za visinu od 180 što predviđa
```



Grafički prikaz

```
plt.scatter(X,y)
plt.plot(X,y_predviđeno,'r') #naš predviđeni pravac, crvena boja
plt.xlabel('visina u cm')
plt.ylabel('tezina u kg')
plt.xlim(150,200) #koji opseg na x da prikaže
plt.show()
```

Micanje loših podataka – outliera

```
outlier_X=np.where(X<150)
outlier_X
outlier_y=np.where(y>120)
outlier_y    #moramo ih ispisati da vidimo index
outlier_index=[38,40,41] #spremimo dobivene indekse naših outliera u varijablu
X_novi=np.delete(X,outlier_index)   #brišemo te sporne točke
y_novi=np.delete(y,outlier_index)
X_novi=X_novi.reshape(-1,1) #opet stavljamo tu dimenziju, ne treba ako imamo više varijabli
reg_novi=LinearRegression().fit(X_novi,y_novi)  #napravimo sada novi model
y_predviđeno_novi=reg_novi.predict(X_novi)
```

S više varijabli

```
#utječe li konjska snaga, težina i ubrzavanje na potrošnju goriva
baza=datasets.fetch_openml('cars')
podaci=np.array(baza.data)
X=podaci[50:100,3:6] #od 50 do 100. podatka, od 3. do 5. stupca (konjska nsaga,težina. ubrzanje)(napišemo 6 jer taj ne uključi)
y=podaci[50:100,0]   #potrošnja goriva je 0ti stupac
```

Normalizacija

```
    from sklearn.preprocessing import normalize
    X_norm=normalize(X)
    y_norm=y/max(y)  #ne moramo normalizirati, samo u X je to problem
#SADA REGRSIJSKI moel
reg_car=LinearRegression().fit(X_norm,y_norm)
reg_car.score(X_norm,y_norm)
y_pred=reg_car.predict(X_norm)
reg_car.predict(normalize([[100,3000,20]]))   #provjeriti za slučaj koji nije u bazi
```

Nadzirano učenje

1.K-nearest neighbours

```
from sklearn import datasets
baza=datasets.load_iris()
X=baza.data #tu su značajke - duljina latice, petiljke...
y=baza.target #tu target tj koja je klasa, samo jedan parametar
```

```
from sklearn.model_selection import train_test_split
X_train,X_test, y_train,y_test=train_test_split(X,y,test_size=0.50,random_state=42) #na 50% podijelili
from sklearn.neighbors import KNeighborsClassifier
modelKNN=KNeighborsClassifier(n_neighbors=3)  #gledamo najbliža 3 susjeda pri klasifikaciji
modelKNN.fit(X_train, y_train)  #na čemu da trenira na ovaj gore definiran način
modelKNN.classes_    #pokaže koje klase ima
modelKNN.effective_metric_  #kako je računao udaljenosti
y_pred_KNN=modelKNN.predict(X_test)   #y_pred uzme x_test i predvidi klasu a y_test ima već spremljenu stvarnu klasu
```

Točnost

```
modelKNN.score(X_test, y_test)  #na testnim podacima
Ili
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred_KNN)

klasa={0:'setosa',1:'versicolor',2:'virginica'} #dali imena klasama
print("za prvi cvijet iz skupa za testiranje predviđena je klasa {} a svarna klasa je {}.".format(klasa[y_pred_KNN[0]],klasa[y_test[0]]))
```

Matrica konfuzije

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred_KNN)  #prvo stvarni, onda predict, koliko klasa tolika nxn
```

Precision i recall

```
#precision gleda false positive tj 100% znači nema false positive
#recall gleda false negative
from sklearn.metrics import precision_score,recall_score, average_precision_score
precision_score(y_test,y_pred_KNN, average=None)
recall_score(y_test,y_pred_KNN,average=None)
```

F1_score

```
# kombinira preciznost i odaziv -> ako povećavamo jedno, smanjuje se drugo
#samo dobri prikazuju = visoka preciznost a niski odaziv
# visok odaziv ali mala preciznost = manje fals negativa tj. manje da kaže nije virus a ust
from sklearn.metrics import f1_score
f1_score(y_test,y_pred_KNN, average=None)
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred_KNN, target_names=baza.target_names))
```

	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	29
versicolor	0.92	1.00	0.96	23
virginica	1.00	0.91	0.95	23
accuracy			0.97	75
macro avg	0.97	0.97	0.97	75
weighted avg	0.98	0.97	0.97	75

Biranje optimalnog modela

```
tocnosti=[]
for i in range(1,21):
  modelKNN = KNeighborsClassifier(n_neighbors=i)
  modelKNN.fit(X_train,y_train)
  tocnosti.append(modelKNN.score(X_test,y_test))
tocnosti
plt.plot(range(1,21),tocnosti)
```

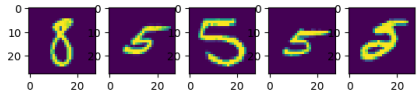
2.GaussianNB

```
from sklearn.naive_bayes import GaussianNB
modelNB= GaussianNB()
modelNB.fit(X_train, y_train)
modelKNN.score(X_test,y_test)
y_pred_NB=modelNB.predict(X_test)
print(classification_report(y_test,y_pred_NB,target_names=baza.target_names))
```

3.Logistička regresija

```
from sklearn.datasets import fetch_openml
mnist=fetch_openml('mnist_784', version=1) #dohvaćanje baze
mnist.keys()  #da vidimo što je u bazi
podaci=np.array(mnist.data)
X=podaci[0:1000,:]/255
y=np.array(mnist.target[0:1000])  #u ispitu paziti na ovaj raspon da je isti kao i gore
      #da tamo gdje je broj 5 bude true a ostali false tj 0 i 1 ako da radimo samo s brojem 5 tj jednom klasom (dvije? o i 1 ?)
      y[np.where(y!='5')]=0
      y[np.where(y=='5')]=1
y=y.astype(int)   #pretvoriti u numerički oblik
from sklearn.model_selection import train_test_split
X_train,X_test, y_train, y_test=train_test_split(X,y,test_size=0.30,random_state=42)

from sklearn.linear_model import LogisticRegression
model_logR=LogisticRegression()
model_logR.fit(X_train,y_train)
model_logR.score(X_test,y_test)

from sklearn.metrics import classification_report #u support piše koliko ih spada u koju klasu
print(classification_report(y_test, model_logR.predict(X_test)))

#matrica konfuzije
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, model_logR.predict(X_test))
```

```
y_pred_logR=model_logR.predict(X_test)
y_test.shape  #da vidimo koja je dimenzija da znamo for petlju
misclassified_index=[] # tu ćemo spremiti indekse krivo klasificiranih
for i in range (y_test.shape[0]):  #[0] prvi stupac tj 300, drugi stupac je prazan
  if y_test[i]!=y_pred_logR[i]:
    misclassified_index.append(i)

#prvih 5 u 1 redak i 5 stupaca
for i,index in enumerate(misclassified_index[0:5]):
  plt.subplot(1,5,i+1)  # i+1 je pozicija na kojoj crta (zato jer plt nema poziciju 0)
  plt.imshow(X_test[misclassified_index[i]].reshape(28,28))
plt.title('Predvidio {}, stvarna {}'.format(y_pred_LogR[misclassified_index[i]],y_test[misclassified_index[i]]))
plt.show()
```



Specificity i Positivity – bolji je kad je veća površina ispod grafa

4.Support vector machine

```
from sklearn import svm
model_svm=svm.SVC()
model_svm.fit(X_train,y_train)
model_svm.score(X_test,y_test)
y_pred_svm=model_svm.predict(X_test)
print(classification_report(y_test,y_pred_svm))
confusion_matrix(y_test, y_pred_svm)
```