VJEŽBA 6: KLASIFIKACIJA PRIMJENOM LOGISTIČKE REGRESIJE. KLASIFIKACIJA METODOM K NAJBLIŽIH SUSJEDA

I. Cilj vježbe:

Upoznati se s problemom klasifikacije. Klasifikacija logističkom regresijom. Klasifikacija metodom k najbližih susjeda.

II. Opis vježbe:

II.1. Nadgledano učenje. Klasifikacija

U ovoj vježbi razmatra se problem nadgledanog učenja gdje je cilj odrediti nepoznatu funkcionalnu ovisnost između *m* ulaznih veličina $X = [x_1, x_2, \dots, x_m]$ i izlazne veličine *y* na temelju podatkovnih primjera. Promatra se slučaj kada je izlazna veličina *y* diskretna veličina, tj. problem klasifikacije. Velik je broj primjera klasifikacije u praksi, npr. detekcija rukom pisanih brojeva, odvajanje neželjene pošte (spam), kategorizacija članaka i sl.
Podatkovni primjeri su parovi koji se sastoje od vektora vrijednosti ulaznih veličina i vrijednosti izlazne veličine, stoga se *i*–ti podatkovni primjer ili uzorak može prikazati kao uređeni par $(\mathbf{x}^{(i)}, y^{(i)})$. Vektor ulaznih veličina zapisuje u obliku:

$$\mathbf{x}^{(i)} = \left[x_1^{(i)}, x_2^{(i)}, \dots, x_m^{(i)}\right]^T . \tag{6-1}$$

Skup koji se sastoji od *n* raspoloživih mjernih podataka $\left(\mathbf{x}^{(1)}, y^{(1)}\right), \left(\mathbf{x}^{(2)}, y^{(2)}\right), \dots, (\mathbf{x}^{(n)}, y^{(n)})$ može se zapisati u matričnom obliku:

$$\mathbf{X} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_m^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_m^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{(n)} & x_2^{(n)} & \dots & x_m^{(n)} \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}. \tag{6-2}$$

Ako izlazna veličina ima samo dvije moguće vrijednosti (klase), npr. $y^{(i)} \in \{0,1\}$, tada se problem naziva binarna klasifikacija. U slučaju kada izlazna veličina može poprimit više od dvije vrijednosti , tada se problem naziva višeklasna klasifikacija. U slučaju višeklasne klasifikacije najčešće se koristi 1-od-*C* označavanje gdje je *C* broj klasa. U tom slučaju vrijednost uzorka izlazne veličine $y^{(i)} \in \mathbb{R}^{C\times 1}$ poprima jedno od *C* mogućih vrijednosti, ovisno kojoj klasi uzorak pripada:

$$y^{(i)} \in \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \dots, \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \right\} . \tag{6-3}$$

Logistička regresija

U slučaju binarne klasifikacije (dvije klase,"0" i "1"), model logističke regresije je oblika:

$$\hat{y}(\mathbf{x}) = h_\theta(\mathbf{x}) = g(\boldsymbol{\theta}^T\mathbf{x}) = \frac{1}{1+e^{-\boldsymbol{\theta}^T\mathbf{x}}} = \begin{cases} zaokruži\ na\ 1\ ako\ je\ \boldsymbol{\theta}^T\mathbf{x} \geq 0 \\ zaokruži\ na\ 0\ ako\ je\ \boldsymbol{\theta}^T\mathbf{x} < 0 \end{cases} \tag{6-4}$$

Iako je cjelokupni model nelinearan, granica odluke u ulaznom prostoru je hiperravnina:

$$\boldsymbol{\theta}^T\mathbf{x} = 0 . \tag{6-5}$$

Kako je izlaz ovakvog modela ograničen na interval (0,1), on se može interpretirati kao vjerojatnost klase "1":

$$p(y = 1|\mathbf{x}) = h_\theta(\mathbf{x}). \tag{6-6}$$

Vrijednosti parametara logističke regresije (6-4) određuje se na minimizacijom kriterijske funkcije uz dane podatke za učenje. Do kriterijske funkcije moguće je doći metodom maksimalne vjerojatnosti uz pretpostavku da su podaci za učenje nezavisni i jednoliko distribuirani. Kriterijska funkcija u tom slučaju glasi:

$$J(\boldsymbol{\theta}) = -\frac{1}{n}\sum_{i=1}^{n}\left\{y^{(i)}\ln h_\theta\left(\mathbf{x}^{(i)}\right) + \left(1-y^{(i)}\right)\ln\left(1-h_\theta\left(\mathbf{x}^{(i)}\right)\right)\right\},$$
$$\boldsymbol{\theta}_{\mathrm{ML}} = \operatorname{argmin}\left(J(\boldsymbol{\theta})\right). \tag{6-7}$$

Rješenje ovog problema ne postoji u zatvorenoj formi pa se moraju koristiti iterativni numerički postupci za optimizaciju.

Proširenje linearnog modela (6-4) moguće je u smislu nelinearne transformacije ulaznih veličina. Na taj način model može generirati granicu odluke u ulaznom prostoru koja je nelinearna funkcija. Često se koristi proširenje polinomima, na način da se ulazne veličine prikladno transformiraju i dodaju postojećim ulaznim veličinama. Npr. u klasifikacijskom problemu s dvije ulazne veličine moguće je dodati sve kvadratne članove pa je podatkovni primjer oblika:

$$\mathbf{x}^{(i)} = [x_1, x_2, x_1^2, x_2^2, x_1x_2]^T \ . \tag{6-8}$$

Iako se ovim postupkom mogu rješavati složeniji problemi klasifikacije, prevelik broj dodatnih izvedenih veličina može uzrokovati pretjerano usklađivanje na podatke (engl. *overfitting*) što će uzrokovati i smanjene predikcijske sposobnosti izgrađenog modela.

Metoda *K* najbližih susjeda

Klasifikacija metodom *K* najbližih susjeda pripada grupi neparametarskih metoda. U takvim modelima ne postoje parametri θ koje je najprije potrebno procijeniti na temelju skupa za učenje već se predikcija za neku novu vrijednost ulaznih veličina **x** (testni uzorak) određuje izravno na temelju raspoloživih podataka (skupa za učenje). Novi mjerni uzorak se klasificira na temelju klase njemu najbližih *K* susjeda. Npr. ako je od 5 najbližih susjeda (uzorka), 4 uzorka klase „0“, a 1 uzorak je klase „1“, tada se novi uzorak klasificira kao „0“. Pri tome se pronalaženje *K* najbližih susjeda svodi na izračunavanje udaljenosti novog (testnog) uzorka do svih uzoraka skupa za učenje. Najčešće se koristi euklidska udaljenost. Matematički se klasifikacija metodom *K* najbližih susjeda definira na način:

$$p(y=j|\mathbf{x}) = \frac{1}{K}\sum_{k\in S} I\left(y^{(k)}=j\right), \tag{6-9}$$

gdje skup *S* sadrži *K* najbližih susjeda uzorku **x**, a $I(a=b)$ je funkcija koja je jednaka 1 ako je $a=b$, a u suprotnom je jednaka 0.
K najbližih susjeda je vrlo jednostavan i popularni algoritam koji stvara nelinearnu granicu odluke u ulaznom prostoru i može se koristiti za složene probleme klasifikacije. Međutim, mane ovog algoritma su memorijska i računalna zahtjevnost.

II.2. Vrjednovanje modela

Testiranje predikcijskih sposobnosti odnosno sposobnosti generalizacije potrebno je provesti na zasebnom skupu podataka koji se naziva skup za testiranje. U slučaju klasifikacije vrlo je korisno prikazati matricu zabune (engl. confusion matrix) na testnim podacima. Ova matrica pokazuje koliko je točno i netočno klasificiranih primjera određenog skupa podataka primjenom odgovarajućeg modela strojnog učenja. U slučaju binarne klasifikacije s klasama {+,-}, matrica je oblika:

Matrica zabune		Stvarna klasa	
		Klasa +	Klasa -
Predviđeno modelom $h_\theta(\mathbf{x})$	Klasa +	**TP** (true positives)	**FP** (false positive)
	Klasa -	**FN** (false negatives)	**TN** (true negatives)

TP – broj primjera iz klase + koji su točno klasificirani modelom
FP – broj primjera iz klase – koji su pogrešno klasificirani modelom kao klasa +
TN – broj primjera iz klase – koji su točno klasificirani modelom
FN – broj primjera iz klase + koji su pogrešno klasificirani modelom kao klasa -

Osim matrice zabune često se izračunavaju i sljedeći pokazatelji nekog klasifikacijskog modela:

$$\begin{aligned} accuracy &= \frac{TP+TN}{TP+TN+FP+FN} \\ missclasification\,rate &= 1 - accuracy \\ precision &= \frac{TP}{TP+FP} \\ recall/sensitivity &= \frac{TP}{TP+FN} \\ specificity &= \frac{TN}{TN+FP} \end{aligned} \tag{6-10}$$

Točnost (engl. *accuracy*) je udio točno klasificirani primjera u cijelom skupu.
Učestalost pogrešne klasifikacije (engl. *missclasification rate*) definira se kao udio pogrešno klasificiranih primjera u cijelom skupu.
Preciznost (engl. *precision*) je udio točno klasificiranih primjera u skupu koje model klasificira kao klasa +.
Odziv (engl. *recall*) je udio točno klasificiranih primjera u skupu primjera koji pripadaju klasi +.
Specifičnost (engl. *specificity*) je udio točno klasificiranih primjera u skupu svih primjera koji pripadaju klasi -.

Primjer

Izgrađen je klasifikator koji na temelju podataka o osobi (pacijentu) zaključuje ima li osoba određenu bolest ili ne. Testni skup sastoji se od 20 uzoraka u kojima je u 12 slučajeva bila prisutna bolest, a u preostalih 8 ne. Rezultat klasifikacije se može prikazati matricom zabune.

Matrica zabune		Stvarna klasa	
		DA	NE
Predviđeno modelom $h_\theta(\mathbf{x})$	DA	9	1
	NE	3	7

accuracy = (9+7)/(20) = 80%
precision = 9/10 = 90%
recall = 9/12 = 75%
specificity = 7/10 = 70%

Zbroj brojeva u tablici je 20. Vidljivo je kako je od 12 bolesnih osoba model točno klasificirao njih 9 dok je 3 označio kao zdrave. U slučaju zdravih osoba, 7 je točno klasificirao dok je jednu osobu klasificirao kao bolesnu.

III. Priprema za vježbu:

Ponovite gradivo vezano uz logističku regresiju i metodu *K* najbližih susjeda.

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/LV6/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/LV6/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 6.1. koja služi za generiranje umjetnih podataka kako bi se demonstrirala binarna klasifikacija za slučaj dvije ulazne veličine. Funkcija prima cijeli broj koji definira željeni broju uzoraka u skupu, a vraća generirani skup podataka u obliku numpy polja pri čemu su prvi i drugi stupac ulazne veličine, a treći stupac klasa kojoj pojedini uzorak pripada. Pomoću ove funkcije generirajte skup za učenje veličine 200 uzoraka (pri tome koristite postavite *seed* generatora slučajnih brojeva na 242 pomoću naredbe *np.random.seed(242)*). Generirajte i skup za

testiranje veličine 100 uzoraka uzoraka (pri tome koristite postavite seed generatora slučajnih brojeva na 242 pomoću naredbe *np.random.seed(12)*). Uključite numpy biblioteku.

Zadatak 2

Prikažite generirane podatke za učenje pomoću *matplotlib* biblioteke u ravnini $x_1 - x_2$. Kako bi se dvije klase razlikovale, potrebno ih je obojati drugom bojom. Za ovaj zadatak vam može poslužiti naredba *scatter* koja osim podataka prima i parametre *c* i *cmap* kojima je moguće definirati boju svake klase.

Zadatak 3

Izgradite model logističke regresije pomoću scikit-learn biblioteke na temelju generiranih podataka za učenje. Koristite metodu *LogisticRegression()* iz biblioteke *sklearn.linear_model*. Kao i sve metode strojnog učenja unutar ove biblioteke, objekt klase logistička regresija ima na raspolaganju metode *fit* (određivanje parametara modela $\boldsymbol{\theta}$) i *predict* (klasifikacija uzoraka na temelju vrijednosti ulaznih veličina). Izgrađeni model sadrži parametre $\boldsymbol{\theta}$ u varijablama *intercept_* i *coef_*. Prikažite granicu odluke naučenog modela ravnini $x_1 - x_2$ zajedno s podacima za učenje. Što primjećujete?

Napomena: granica odluke u ravnini $x_1 - x_2$ definirana je kao krivulja: $\theta_0 + \theta_1 x_1 + \theta_1 x_2 = 0$

Zadatak 4

U prilogu vježbe nalazi se kod 6.2. koji će obojati sliku s obzirom na izlaz modela logističke regresije u obliku vjerojatnosti (6-6). Iskoristite ovaj kod kako biste prikazali izlaz logističke regresije u obliku vjerojatnosti zajedno s podacima za učenje.

Napomena: objekt naziva LogRegModel u kodu je model logističke regresije te ga zamijenite s odgovarajućim imenom vašeg modela.

Zadatak 5

Provedite klasifikaciju testnog skupa podataka pomoću izgrađenog modela logističke regresije i metode *predict*. Prikažite skup za testiranje u ravnini $x_1 - x_2$. Zelenom bojom označite dobro klasificirane uzorke dok pogrešno klasificirane uzorke označite crnom bojom.

Zadatak 6

Funkcija 5.3. u dodatku prima izračunatu matricu zabune pri čemu se klase označavaju brojevima 0-9. Prikažite pomoću ove funkcije matricu zabune izgrađenog modela logističke regresije na testnim podacima. Matricu zabune moguće je izračunati naredbom *confusion_matrix* koja se nalazi u *sklearn.metrics*. Izračunajte i pokazatelje dane izrazima (6-10) (napišite vlastitu funkciju ili koristite gotove iz *sklearn.metrics*).

Zadatak 7

Ponovite zadatke 3,4,5,6 za slučaj kada se u model logističke regresije proširuje s dodatnim ulaznim veličinama koje su polinomska transformacija postojećih. Prošireni skup ulaznih veličina (*data_train_new*) veličine moguće je dobiti iz osnovnog skupa za učenje (*data_train*) naredbama:

```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=3, include_bias = False)
data_train_new = poly.fit_transform(data_train[:,0:2])
```

pri čemu je *degree* cijeli broj koji označava do kojeg reda će se raditi proširenje skupa. Npr. za *degree* = 2 prošireni skup ima uzorke oblika (6-8). Što primjećujete? Kako komentirate dobivene rezultate? Što se događa kada mijenjate *degree* od malih prema većim vrijednostima?

Zadatak 8

Na temelju podataka za učenje izgradite model metodom *K* najbližih susjeda. Ulazne veličine je potrebno standardizirati prije postupka učenja modela naredbom *preprocessing.scale(X)* gdje je matrica *X* matrica ulaznih veličina (učitajte naredbu pomoću *from sklearn import preprocessing*). Granicu odluke ovog modela prikažite pomoću funkcije 6.4. iz dodatka koja za argumente ima model *K* najbližih susjeda, matricu ulaznih veličina i vektor izlazne veličine. Što primjećujete? Kako se granica mijenja ako se broj susjeda povećava/smanjuje? Što se događa ako isključite standardizaciju podataka? Prikažite matricu zabune izgrađenog modela na testnim podacima

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/LV6/solutions/`.

VI. Dodatak

Funkcija 6.1. – generiranje podataka

```
def generate_data(n):

    #prva klasa
    n1 = n/2
    x1_1 = np.random.normal(0.0, 2, (n1,1));
    #x1_1 = .21*(6.*np.random.standard_normal((n1,1)));
    x2_1 = np.power(x1_1,2) + np.random.standard_normal((n1,1));
    y_1 = np.zeros([n1,1])
    temp1 = np.concatenate((x1_1,x2_1,y_1),axis = 1)

    #druga klasa
    n2 = n - n/2
    x_2 = np.random.multivariate_normal((0,10), [[0.8,0],[0,1.2]], n2);
    y_2 = np.ones([n2,1])
    temp2 = np.concatenate((x_2,y_2),axis = 1)

    data  = np.concatenate((temp1,temp2),axis = 0)

    #permutiraj podatke
    indices = np.random.permutation(n)
    data = data[indices,:]

    return data
```

Kod 6.2. – prikaz izlaza logističke regresije

```
f, ax = plt.subplots(figsize=(8, 6))
x_grid, y_grid = np.mgrid[min(data_train[:,0])-0.5:max(data_train[:,0])+0.5:.05,
                          min(data_train[:,1])-0.5:max(data_train[:,1])+0.5:.05]
grid = np.c_[x_grid.ravel(), y_grid.ravel()]
probs = LogRegModel.predict_proba(grid)[:, 1].reshape(x_grid.shape)

cont = ax.contourf(x_grid, y_grid, probs, 60, cmap="Greys", vmin=0, vmax=1)

ax_c = f.colorbar(cont)
ax_c.set_label("$P(y = 1|\mathbf{x})$")
ax_c.set_ticks([0, .25, .5, .75, 1])
ax.set_xlabel('$x_1$', alpha=0.9)
ax.set_ylabel('$x_2$', alpha=0.9)
ax.set_title('Izlaz logisticke regresije')
plt.show()
```

Funkcija 6.3. – prikaz matrice zabune

```
def plot_confusion_matrix(c_matrix):

    norm_conf = []
    for i in c_matrix:
        a = 0
        tmp_arr = []
        a = sum(i, 0)
        for j in i:
            tmp_arr.append(float(j)/float(a))
        norm_conf.append(tmp_arr)

    fig = plt.figure()
    ax = fig.add_subplot(111)
    res = ax.imshow(np.array(norm_conf), cmap=plt.cm.Greys, interpolation='nearest')

    width = len(c_matrix)
    height = len(c_matrix[0])
```

```
    for x in xrange(width):
        for y in xrange(height):
            ax.annotate(str(c_matrix[x][y]), xy=(y, x),
                        horizontalalignment='center',
                        verticalalignment='center', color = 'green', size = 20)

    fig.colorbar(res)
    numbers = '0123456789'
    plt.xticks(range(width), numbers[:width])
    plt.yticks(range(height), numbers[:height])

    plt.ylabel('Stvarna klasa')
    plt.title('Predvideno modelom')
    plt.show()
```

Funkcija 6.4. – prikaz granice odluke metode K najbližih susjeda

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn import preprocessing

def plot_KNN(KNN_model, X, y):

    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1

    xx, yy = np.meshgrid(np.arange(x1_min, x1_max, 0.01),
                         np.arange(x2_min, x2_max, 0.01))

    Z1 = KNN_model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z1.reshape(xx.shape)

    plt.figure()
    plt.pcolormesh(xx, yy, Z, cmap='PiYG', vmin = -2, vmax = 2)
    plt.scatter(X[:,0], X[:,1], c = y, s = 30, marker= 'o' , cmap='RdBu',  edgecolor='white', label = 'train')
```