

UNIVERZITET U NOVOM SADU  
FAKULTET TEHNIČKIH NAUKA  
KATEDRA ZA TELEKOMUNIKACIJE I OBRADU SIGNALA

## PRAKTIKUM IZ MAŠINSKOG UČENJA

Tijana Nosek  
Branko Brkljač  
Danica Despotović  
Milan Sečujski  
Tatjana Lončar-Turukalo

UNIVERZITET U NOVOM SADU  
2020.

Naziv udžbenika:  
Praktikum iz mašinskog učenja

Autori:  
Mast. inž. Tijana Nosek, Fakultet tehničkih nauka, Univerzitet u Novom Sadu  
Doc. dr Branko Brkljač, Fakultet tehničkih nauka, Univerzitet u Novom Sadu  
Mast. inž. Danica Despotović, Institut za viziju, Univerzitet Sorbona  
Prof. dr Milan Sečujski, Fakultet tehničkih nauka, Univerzitet u Novom Sadu  
Prof. dr Tatjana Lončar-Turukalo, Fakultet tehničkih nauka, Univerzitet u Novom Sadu

Recenzenti:  
Dr Jelena Nikolić, vanredni profesor, Elektronski fakultet, Univerzitet u Nišu  
Dr Nikša Jakovljević, vanredni profesor, Fakultet tehničkih nauka, Univerzitet u Novom Sadu

Izdavač:  
Univerzitet u Novom Sadu

Dizajn, priprema i štampa:  
Štamparija Futura

Sva prava zadržana. Preštampavanje zabranjeno u celini i u delovima.

CIP-Каталогизација у публикацији  
Библиотека Матице српске, Нови Сад

BROJ

**PRAKTIKUM iz mašinskog učenja** / Tijana Nosek ... [et al.]. - 1.izd. - Novi Sad, Univerzitet u Novom Sadu. 2020. ...  
Tiraž 80.

ISBN 978-86-499-0245-9

1. Тијана Носек, 1992. [автор]  
а) Машино учење - практикум  
COBISS AAAAААААА

This project has been funded with support from the European Commission. This publication reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Ovaj projekat finansijski je podržan od strane Evropske komisije. Podrška Evropske

---

komisije ovoj publikaciji ne odražava podršku sadržaju koji predstavlja isključivo stavove autora i Komisija ne može biti odgovorna za upotrebu sadržanih informacija.

# Sadržaj

<b>Predgovor</b>	
<b>Notacija</b>	
<b>1 Uvod</b>	<b>1</b>
1.1 <i>Python</i> u mašinskom učenju . . . . .	3
<b>2 Osnovni pojmovi</b>	<b>10</b>
2.1 Tipovi obeležja . . . . .	10
2.2 Normalizacija obeležja . . . . .	11
2.3 Izbor modela . . . . .	13
2.4 Procena greške na nezavisnom test skupu . . . . .	14
2.5 Trostruka podela podataka . . . . .	16
2.6 Unakrsna validacija sa $K$ particija . . . . .	17
2.7 Mere uspešnosti klasifikatora . . . . .	19
2.8 Mere uspešnosti regresora . . . . .	25
<b>3 Analiza obeležja</b>	<b>27</b>
3.1 Predobrada nepotpunih podataka . . . . .	27
3.2 Univarijantna analiza obeležja . . . . .	29
3.3 Bivarijantna i multivarijantna analiza obeležja . . . . .	32
<b>4 Bajesovo pravilo i kvadratni klasifikatori</b>	<b>39</b>
<b>5 Estimacija GRV: KDE i <math>k</math>NN</b>	<b>47</b>
5.1 Procena GRV na osnovu kernela . . . . .	48
5.2 Procena GRV metodom $k$ najbližih suseda . . . . .	54
<b>6 Naivni Bajesov klasifikator</b>	<b>58</b>
<b>7 Linearna regresija</b>	<b>65</b>
<b>8 Logistička regresija</b>	<b>75</b>
<b>9 Klasifikator metodom <math>k</math> najbližih suseda</b>	<b>79</b>
<b>10 Mašina na bazi vektora nosača</b>	<b>84</b>

<b>11 Stabla odluke</b>	<b>93</b>
11.1 Metoda slučajne šume . . . . .	97
<b>12 Metode za smanjenje dimenzionalnosti: PCA i LDA</b>	<b>101</b>
12.1 Razlaganje na glavne komponente - PCA . . . . .	101
12.2 Linearna diskriminantna analiza - LDA . . . . .	105
<b>13 Uvod u neuronske mreže</b>	<b>110</b>

## Predgovor

„Praktikum iz Mašinskog učenja“ predstavlja pomoćni udžbenik za predmet Mašinsko učenje, koji se izučava na različitim studijskim programima osnovnih i master studija na Fakultetu tehničkih nauka u Novom Sadu. Praktikum, pored toga, može biti koristan i svim drugim studentima koji se u okviru svojih studijskih programa bave ovom oblašću. Za savladavanje izloženog gradiva čitaocu su neophodna osnovna znanja iz linearne algebre i teorije verovatnoće.

U prvom poglavlju dat je uvod u oblast mašinskog učenja i definisani su osnovni problemi i tipovi algoritama u ovoj oblasti. Poseban osvrt dat je na upotrebu programske jezike *Python* u rešavanju problema mašinskog učenja.

U drugom poglavlju definisani su osnovni pojmovi mašinskog učenja, kao što su obeležje, uzorak i model. Pored toga, opisane su najčešće metode za evaluaciju performansi sistema mašinskog učenja, kako u rešavanju problema klasifikacije tako i u rešavanju problema regresije.

U trećem poglavlju sa posebnim osvrtom na parametre kojima se jednodimenzionalni i višedimenzionalni skupovi uzoraka mogu opisati.

U četvrtom poglavlju dat je uvod u odlučivanje na osnovu Bajesovog pravila, minimizacijom verovatnoće greške, ili u opštijem slučaju, minimizacijom Bajesovog rizika. Posebna pažnja posvećena je kvadratnim klasifikatorima, koji se dobijaju kada se uslovne gustine raspodele verovatnoće po klasama modeluju Gausovim raspodelama.

Peto poglavlje posvećeno je neparametarskim metodama procene gustine raspodele verovatnoće, odnosno, metodama zasnovanim na upotrebni kernela (KDE) i metodama zasnovanim na položaju  $k$  najbližih suseda u prostoru obeležja ( $k$ NN). Analizirane su specifičnosti obe metode i ukazano je na prednosti i nedostatke svake od njih.

U šestom poglavlju opisan je naivni Bajesov klasifikator, koji, u cilju prevezilaženja problema retkosti podataka, polazi od pretpostavke da su obeležja unutar svake klase međusobno statistički nezavisna.

U sedmom poglavlju predstavljena je linearna regresija, kao metoda za predviđanje nepoznate vrednosti izlazne promenljive zasnovana na prepostavci da se ona može predstaviti linearnom kombinacijom ulaznih obeležja. Predstavljene su i dve često korišćene regularizacione tehnike, čiji je cilj da smanje varijansu procene na račun malog povećanja pristrasnosti.

U osmom poglavlju predstavljena je logistička regresija, kao metoda klasifikacije na osnovu predviđene kontinualne vrednosti izlazne promenljive, uz pretpostavku da je ona statistički povezana sa odgovarajućom nelinearnom kombi-

nacijom ulaznih obeležja. Uz pogodan izbor nelinearnosti, izlazna promenljiva može se interpretirati kao verovatnoća pripadanja uzorka određenoj klasi.

Deveto poglavlje bavi se klasifikacijom na osnovu  $k$  najbližih suseda u prostoru obeležja ( $k$ NN). Ova metoda klasifikacije pripada metodama kasnog učenja, izuzetno je jednostavna, a postiže relativno dobre rezultate. Ipak, kao i sve metode kasnog učenja, zbog velikih zahteva u pogledu potrebne procesorske snage i memorijskog prostora, njena direktna primena u rešavanju visokodimenzionalnih problema nije praktična.

U desetom poglavlju dat je uvod u klasifikatore koji se nazivaju mašinama zasnovanim na vektorima nosačima. Ovi klasifikatori, koji se prvenstveno koriste za rešavanje problema binarne klasifikacije, zasivaju se na identifikovanju hiperpovrši koja treba da razdvoji uzorke u prostoru obeležja tako da između oblasti popunjениh uzorcima različitih klasa postoji što širi prostor.

Jedanaesto poglavlje bavi se primenom stabala odluke u rešavanju klasifikacionih i regresionih problema. Pored toga, na primeru ovog modela uveden je i koncept ansambalskog učenja, u kom se više nezavisnih klasifikatora, odnosno, regresora, obučava na različitim skupovima za obuku, da bi se donošenje krajnje odluke o klasi ili nepoznatoj vrednosti vršilo glasanjem u slučaju klasifikacije, odnosno usrednjavanjem rezultata u slučaju regresije.

U poslednjem, dvanaestom poglavlju, izložene su dve najčešće korišćene metode za smanjenje dimenzionalnosti – razlaganje na glavne komponente (PCA) i linearna diskriminantna analiza (LDA). Prva ima za cilj da predstavi uzorke iz visokodimenzionalnog prostora u prostoru sa manjim brojem dimenzija uz što manji gubitak informacije, dok druga nastoji da pri tome očuva i što više diskriminatorskih informacija.

Novi Sad, avgust 2020.

## Notacija

Osnovne oznake koje će biti korišćene u praktikumu date su u Tabeli 1.

Značenje	Oznaka
$\mathbf{x}$	vektor obeležja tj. uzorak
$N$	ukupan broj uzoraka
$D$	ukupan broj obeležja (dimenzionalnost prostora)
$\mathbf{X}_{N \times D}$	matrica podataka ( $N$ uzoraka dimenzionalnosti $D$ )
$\mathbf{x}^{(n)}$	$n$ -ti vektor obeležja ( $n$ -ti uzorak)
$\mathbf{x}_i$	vrednosti $i$ -tog obeležja ( $i$ -te dimenzijske prostore)
$x_i^{(n)}$	vrednost $i$ -tog obeležja uzorka $\mathbf{x}^{(n)}$ ( $i$ -ta dimenzija $n$ -tog uzorka)
$K$	ukupan broj klase
$K_k$	oznaka (labela) $k$ -te klase
$K_k^{(n)}$	oznaka (labela) $n$ -tog uzorka $k$ -te klase

**Tabela 1:** Notacija osnovnih pojmova korišćenih u praktikumu.

# 1. Uvod

Mašinsko učenje (engl. *machine learning* – ML) predstavlja oblast veštacke inteligencije. Mašinsko učenje se bavi algoritmima koji se, umesto da prate eksplicitno date korake, obučavaju za određen zadatak na osnovu podataka, na sličan način kao što to rade ljudi kada uče na osnovu svojih zapažanja da bi kasnije primenjivali zaključke do kojih su došli. Na primer, autonomna vozila se obučavaju na osnovu podataka sa senzora koji su pratili kretanje automobila koji vozi čovek i kasnije primenjuju naučena pravila vožnje. Mašine uče odnose među dostupnim podacima, otkrivaju određene pravilnosti u njima i na osnovu tih saznanja formiraju matematičke modele odlučivanja koje mogu da primene za predviđanja na novim primerima istog problema. Baza podataka na osnovu kojih se obučavaju ML algoritmi naziva se *skup za obuku*. Veći skup podataka za obuku, u opštem slučaju, rezultovaće matematičkim modelom boljih performansi. Proces kreiranja matematičkog modela naziva se *obuka modela*.

U mašinskom učenju postoje dva osnovna pojma: *uzorak* i *obeležje*. Skup za obuku sačinjavaju uzorci, a svaki uzorak u skupu opisan je vektorom čiji su elementi karakteristike (obeležja) uzorka. Broj obeležja određuje dimenzionalnost prostora u kom se uzorci (vektori) nalaze. Obeležja predstavljaju osobine koje opisuju uzorce i omogućuju njihovo poređenje i razlikovanje. Na primer, u rešavanju problema predviđanja cene nekretnina stambeni objekat predstavlja jedan uzorak, dok obeležja mogu biti kvadratura, godina izgradnje, grad ili deo grada u kojem se dati objekat nalazi i sl. U problemu utvrđivanja potencijalnih bolesnika od neke konkretne bolesti, jedan uzorak će biti jedan ispitanik, a obeležja mogu biti starost, pol, krvni pritisak, indeks telesne mase i sl. Obično se baza podataka posmatra kao matrica  $\mathbf{X}$  čiji svaki red predstavlja uzorak, dok svaka kolona odgovara jednom obeležju. Dimenzija takve matrice je  $N \times D$ , gde je  $N$  broj uzoraka u bazi podataka, a  $D$  broj obeležja kojima je svaki uzorak opisan. Primer nekoliko uzoraka iz jedne baze podataka (deo opisane matrice) prikazan je na Slici 1.1.

Algoritmi mašinskog učenja mogu se podeliti u dve velike grupe, a to su algoritmi nadgledanog i nenadgledanog učenja. Osnovna razlika između ove dve grupe algoritama jeste u dostupnosti informacije o tačnoj vrednosti izlazne veličine za svaki raspoloživi uzorak. Kod algoritama nadgledanog učenja za svaki uzorak iz skupa za obuku algoritma data je i tačna izlazna vrednost za taj uzorak (postoji određen broj stambenih objekata koji su već prodati pa se zna i njihova cena, ili postoji baza pacijenata za koje je već poznato da li imaju određenu bolest ili ne). Podaci o izlaznim vrednostima koriste se u obuci algoritama koji

	kvadratura	god. izgradnje	garaža	broj soba	deo grada	cena
kuća1	150	1990	da	5	Telep	120000
kuća2	125	1998	ne	3	Adice	100000
kuća3	230	2001	ne	6	Telep	200000
kuća4	190	1990	ne	5	Podbara	180000
kuća5	170	2003	da	4	Podbara	230000
:	:	:	:	:	:	:

**Slika 1.1:** Baza podataka predstavljena kao matrica čije vrste predstavljaju uzorke, a kolone obeležja.

će za neki novi neobeleženi uzorak predvideti vrednost izlaza (cena stambenog objekta koji se upravo stavlja na prodaju, ili klasna pripadnost novog pacijenta). Metode nenadgledanog učenja ne koriste informaciju o tačnoj izlaznoj vrednosti za svaki uzorak, koje vrlo često nisu ni raspoložive. Ove metode nastoje da otkriju strukturu u podacima na osnovu raspoloživih obeležja koja opisuju uzorke. Tri osnovna tipa problema mašinskog učenja su *regresija* i *klasifikacija*, kao problemi nadgledanog učenja, i *klasterizacija*, kao primer problema nenadgledanog. Problemi nadgledanog učenja tretiraju se kao regresioni ukoliko je izlazna promenljiva kontinualnog tipa. U ovom slučaju cilj je formiranje modela koji će na osnovu dostupnih uzoraka za obuku i odgovarajućih izlaznih vrednosti predviđati (kontinualnu) vrednost izlazne promenljive za novi uzorak (na primer, predviđanje cene stambenog objekta na osnovu kvadrature, godine izgradnje i dr.). Kod klasifikacionih problema izlazne promenljive za dostupne uzorke su označke određenih kategorija, na primer bolestan/zdrav. Rezultat obuke je model klasifikacije na osnovu kojeg bi se utvrdila klasna pripadnost za uzorke za koje ona nije poznata. Najzad, kod rešavanja problema klasterizacije cilj je grupisati uzorke tako da se u istoj grupi nađu međusobno slični uzorci, pri čemu se sličnost procenjuje na osnovu vrednosti dostupnih obeležja (na primer, formiranje grupa međusobno sličnih kupaca na osnovu informacija kog su pola, koliko imaju godina, koje proizvode kupuju i dr.). U ovom praktikumu prikazani su neki od algoritama nadgledanog učenja sa akcentom na detalje implementacije uz kraći opis metoda kako bi se značenje parametara u implementaciji povezalo sa parametrima u matematičkom modelu. Činjenica da se obrađuju samo algoritmi nadgledanog učenja podrazumeva da se u bazi

uzoraka za obuku pored ulaznih obeležja, kojima se opisuje svaki uzorak, mora jasno definisati i obezbediti vrednost izlazne promenljive koja se predviđa (u primeru sa kućama to su cene kuća, a u primeru sa pacijentima su to oznake klase sa mogućim vrednostima *zdrav* i *bolestan*).

### 1.1. Python u mašinskom učenju

*Python* je interpreterski programski jezik visokog nivoa, što kod čini čitljivijim i jednostavnijim za tumačenje. Takođe je veoma fleksibilan, što ga čini pogodnim i za početnike u programiranju, ali i za programere iskusne sa drugim programskim jezicima. Posebno je popularan u oblasti obrade podataka i mašinskog učenja i ima velik broj specijalizovanih biblioteka. Biblioteke ključne za primenu u mašinskom učenju su: NumPy, SciPy, Pandas, Matplotlib, Seaborn i Scikit-learn. NumPy i SciPy služe za numerička izračunavanja. Biblioteka NumPy između ostalog podržava višedimenzionalne nizove i osnovne operacije i funkcije linearne algebre nad njima. Biblioteka SciPy nudi razne numeričke algoritme i specijalizovane alate za obradu signala, optimizaciju, statistiku i dr. Biblioteke Matplotlib i Seaborn koriste se za vizuelizaciju podataka. Scikit-learn je biblioteka namenjena isključivo za oblast mašinskog učenja i sadrži implementirane algoritme mašinskog učenja za klasifikaciju, regresiju, klasterizaciju, smanjenje dimenzionalnosti, kao i alate za izbor modela i predobradu podataka. Biblioteka Pandas namenjena je analizi podataka i nudi pogodne strukture podataka i alate za njihovu analizu. Osnovna struktura koju ona koristi je DataFrame, koja se može posmatrati kao tabela sa nazivima kolona i vrsta. Ova struktura podataka podržava efikasnu manipulaciju podacima u smislu njihovog izdvajanja, transformisanja, uklanjanja, dopune i sl. korišćenjem odgovarajućih alata biblioteke Pandas. Pored toga, ova biblioteka nudi alate za jednostavno učitavanje podataka iz tekstualnih fajlova, Excel fajlova i drugih formata. Funkcije i delovi koda u okviru praktikuma zasnivaće se na verziji *Python3*.

Postoje različita okruženja za razvoj *Python* programa. Jedno od popularnijih okruženja, koje obuhvata editor, interpreter i debager, jeste PyCharm. Drugo popularno okruženje je Jupyter, čija se specifičnost ogleda u tome što je smešteno na serveru, gde se kod i izvršava, tako da je za korišćenje ovog okruženja potrebna veza sa internetom. Jupyter je vrlo jednostavan za korišćenje, interaktivan je i često se koristi u nastavi i na prezentacijama. Razlog za to je što se jedan skript u Jupyter-u, tzv. notebook, sastoji od ćelija, a svaka ćelija može da sadrži ne samo kod, nego i običan tekst, slike i drugo, što ga čini interesantnim za predstavljanje rezultata ili izlaganje gradiva. Zbog svega nave-

denog, rešenja zadataka iz praktikuma biće data u okviru *Jupyter* skriptata ovde: [www.github.com/ftn-ktios/MUpraktikum](https://www.github.com/ftn-ktios/MUpraktikum).

Kroz jednostavan zadatak u nastavku će biti prikazana upotreba pojedinih funkcija iz pomenutih biblioteka koje su neophodne za rešavanje zadataka datih u praktikumu. U zadatku će biti kreirana mala baza podataka koja će biti sačuvana kao CSV fajl, zatim će biti učitana iz tog fajla, da bi na kraju bila izvršena kratka analiza podataka i vizualizacija.

1. Učitati biblioteke **Pandas** i **NumPy** i uvesti skraćenice za njihovo kasnije korišćenje `pd` i `np`, respektivno. Takođe učitati i **pyplot** kolekciju funkcija biblioteke **Matplotlib** i uvesti skraćenicu `plt` za njeno dalje korišćenje.

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

2. Formirati inicijalni skup imena kao listu 5 različitih podataka tipa *string*.

```
names = [ 'Nikola' , 'Sara' , 'Marko' , 'Milica' , 'Luka' ]
```

3. Na slučaj odabrati ime iz generisane liste `names` i upisati ga u novu listu. To ponoviti 1000 puta, tako da na kraju nova lista sadrži 1000 imena (od kojih se neka ponavljaju).

```
for i in range(1000):
    n = np.random.randint(low=0, high=len(names))
    random_names.append(names[n])
print('Prvih 10: ', random_names[:10])
print('Od 15. do 20. elementa: ', random_names[15:21])
```

Ovde je iskorišćeno sledeće:

- Funkcija `range(start, stop, step)` generiše niz brojeva od `start` do `stop` sa korakom `step`. Kada se prosledi jedan parametar, po-drazumeva se da je `start=0`, a `step=1`. Ova funkcija podržava isključivo celobrojne vrednosti, a u slučaju necelobrojnih vrednosti koristi se funkcija `arange(start, stop, step)` iz biblioteke NumPy, za čije ulazne parametre važe ista pravila.
- Funkcija `len(x)` vraća dužinu liste `x`.

- Indeksiranje liste vrši se navođenjem indeksa u uglastim zagradama. Prvi element liste `names` je `names[0]`, poslednji je `names[-1]`, a  $n$ -ti je `names[n]`. Prvih  $n$  elemenata izdvaja se pozivom `names[:n]`, a poslednjih  $n$  pozivom `names[-n:]`.
  - Funkcija `randint(low,high)` generiše slučajan ceo broj iz opsega `[low, high]`.
  - Metoda `list.append(x)` dodaje element `x` na kraj liste `list`.
4. Neka lista imena predstavlja imena beba rođenih 2016. godine u Srbiji. Potrebno je napraviti listu od 1000 slučajnih brojeva tako da se za svako ime sa liste generiše jedan ceo broj u opsegu [1, 500] koji će predstavljati broj rođenih beba sa tim imenom u jednom porodilištu. S obzirom da lista obuhvata decu rođenu u više porodilišta u raznim gradovima, imena u bazi se ponavljaju jer se odnose na svako porodilište zasebno.

```
births = []
for i in range(1000):
    births.append(np.random.randint(low=1, high=500))
```

5. Spojiti napravljene liste `names` i `births`. Potom dobijenu strukturu prebaciti u `DataFrame` i sačuvati u CSV fajl.

```
BabyDataSet=zip(random_names, births)
BabyDataSet = pd.DataFrame(BabyDataSet)
BabyDataSet.to_csv('births2016.txt',
                   index=False, header=False)
# BabyDataSet.to_excel('births1880.xlsx',
                     index=False)
```

Ovde je iskorišćeno sledeće:

- Funkcija `zip` namenjena je za pravljenje iteratora koji istovremeno prolazi kroz više lista. U ovom zadatku poslužila je da udruži članove lista `random_names` i `births` na istim pozicijama.
- Primenom funkcije `DataFrame(x)` iz biblioteke `Pandas` matrica (ili rečnik) `x` konvertuje se u istoimenu strukturu.
- Metode `to_csv(filename)` i `to_excel(filename)` primenjuju se direktno na `DataFrame` i služe za čuvanje podataka u CSV i Excel

fajlovima. Obavezan parametar je ime fajla (sa putanjom i ekstenzijom) u kom će podaci biti snimljeni. U zadatku su navedeni i parametri `index=False` i `header=False` kako se nazivi kolona i vrsta ne bi snimili u fajl.

- Učitati podatke iz CSV fajla u `DataFrame` i pogledati prvih i poslednjih nekoliko redova. Dodeliti imena kolonama. Proveriti dimenzije `DataFrame-a`.

```
df = pd.read_csv('births2016.txt', header=None)
df.columns=['Names', 'Births']
print(df.shape)
df.head()
df.tail()
```

Ovde je iskorišćeno sledeće:

- Funkcija `read_csv(filename)` služi za učitavanje podataka iz CSV fajla u `DataFrame`. Obavezan parametar je ime fajla (sa putanjom i ekstenzijom) odakle se podaci učitavaju. U zadatku je naveden i parametar `header=None` jer u fajlu ne postoji nazivi kolona, a kako se u opštem slučaju podrazumeva da postoje, bez toga bi podaci iz prvog reda fajla bili interpretirani kao nazivi kolona.
- Svaki `DataFrame` ima ugrađene metode `columns` i `index`. Mogu se koristiti za proveru imena kolona i vrsta, respektivno, ili za njihovo dodavanje/menjanje, kao što je u dato u zadatku. Imena se navode u obliku liste čija se dužina mora poklapati sa brojem kolona/vrsta `DataFrame-a`.
- Za proveru dimenzija `DataFrame-a` koristi se metoda `shape`. Metoda vraća uređeni par (`broj_vrsta, broj_kolona`).
- Funkcije `head(x)` i `tail(x)` služe za prikaz prvih  $x$  i poslednjih  $x$  redova `DataFrame-a`, respektivno. Ukoliko ulazni parametar  $x$  nije naveden, smatra se da je  $x=5$ .

- Izdvojiti jedinstvena imena iz baze i ispisati ih. Izbaciti one vrste koje imaju vrednost kolone `Births` jednaku 0.

```
print(pd.unique(df['Names']))
indexes_to_drop = df.loc[df['Births']==0].index
df.drop(indexes_to_drop, axis=0, inplace=True)
print(df.shape)
```

Ovde je iskorišćeno sledeće:

- Indeksiranje DataFrame-a može se izvršiti na različite načine. Ovde je pristupljeno koloni navođenjem njenog imena u uglastim zagradama: `df['Names']`. Međutim, već pristupanje vrsti korišćenjem njenog imena nije moguće upotrebom samo zagrada, ali je moguće upotrebom metode `loc: df.loc['ime_vrste']`. Ova metoda se koristi kada je cilj pristupiti kolonama, vrstama ili konkretnom elementu DataFrame-a korišćenjem imena kolona i vrsta. Određenoj koloni može se pristupiti pozivom `df.loc[:, 'ime_kolone']`, dok se konkretnom elementu može pristupiti pozivom `df.loc['ime_vrste', 'ime_kolone']`. Ova metoda takođe se koristi i za logičko indeksiranje, što je i slučaj u ovom zadatku: `df['Births']==0` će vratiti vektor logičkih vrednosti sa pozitivnim vrednostima na mestima gde je uslov ispunjen, tako da će `df.loc[df['Births']==0]` vratiti vrste za koje je ispunjen uslov da u koloni pod nazivom Births imaju vrednost 0. DataFrame se može indeksirati i rednim brojevima vrsta i kolona koristeći metodu `iloc`. Recimo, elementu iz 4. vrste i 3. kolone može se pristupiti pozivom `df.iloc[4,3]`.
  - Funkcija `unique(x)` iz biblioteke Pandas vraća listu jedinstvenih vrednosti iz vektora `x`. Ista funkcija postoji i u biblioteci NumPy.
  - Metoda `df.drop(list_num, axis)` primenjena na DataFrame izbacuje vrste pod rednim brojevima navedenim u listi `list_num` pod uslovom da je parametar `axis=0`, odnosno kolone pod rednim brojevima navedenim u `list_num` pod uslovom da je parametar `axis=1`. Ako je parametar `inplace=True` tada će promena biti izvršena u samom DataFrame-u, a inače se formirani DataFrame mora snimiti u novu promenljivu.
8. Sumirati podatke tako da u bazi ostanu samo jedinstvena imena, ali da se broj rođenja deteta pod tim imenom odnosi na ukupan broj, a ne na broj po unisu. Proveriti dimenzije tako dobijenog DataFrame-a.

```
df = df.groupby('Names').sum()
print(df.shape)
```

Ovde je iskorišćeno sledeće:

- Metoda `groupby(by='ime_kolone')` formira objekat u kom su grupirani redovi `DataFrame`-a prema vrednostima iz kolone čije je ime specificirano kao vrednost parametra `by`. Nakon kreiranja objekta nad njim se mogu izvršiti različite metode u zavisnosti od toga da li je cilj grupisane podatke sabrati, usrednjiti, naći njihov maksimum ili nešto drugo.
- Metoda `sum` služi za sumiranje podataka iz `DataFrame`-a, a podrazumeva se da sumira sve elemente, mada se parametrom `axis` može specificirati sumiranje po kolonama ili po vrstama. Ekvivalentna funkcija postoji i u biblioteci NumPy.

9. Pronaći najpopularnije ime sortiranjem po popularnosti.

```
value_sorted = np.sort(df[ 'Births' ]) [-1]
index_sorted = np.argsort(df[ 'Births' ]) [-1]
print('Najpopularnije ime ', df.index[index_sorted],
      ' javlja se ', value_sorted, ' puta.')
```

Ovde je iskorišćeno sledeće:

- Funkcija `sort(x)` iz biblioteke NumPy sortira vrednosti iz `x` u rastućem redosledu i vraća ih u vidu liste.
- Funkcija `argsort(x)` iz biblioteke NumPy sortira vrednosti iz `x` u rastućem redosledu i vraća njihove indekse u vidu liste.

10. Prikazati stubičasti grafik popularnosti svakog od imena u opadajućem redosledu.

```
df_sort = df.sort_values(by='Births')
plt.figure(figsize=(16,9))
df_sort[ 'Births' ][:: -1].plot.bar()
```

Ovde je iskorišćeno sledeće:

- Metoda `sort_values` sortira vrednosti u strukturi `DataFrame`, a po redak elemenata zavisi od prosleđenih parametara. Parametar `by` prihvata naziv kolone (vrste) date strukture i elementi se sortiraju prema vrednostima te kolone (vrste). Ako je vrednost `by` ime kolone, vrednost parametra `axis` mora biti 0 ili `index` (što je i podrazumevana vrednost) jer se tada vrši sortiranje vrsta, a ako se vrši sortiranje

kolona, `axis` mora biti 1 ili `columns`, a `by` sadrži ime vrste. Podrazumevani redosled je rastući, a može se tražiti i opadajući postavljanjem `ascending` parametra na `False`.

- Funkcija `figure` služi za otvaranje novog prozora za prikaz grafika, a njegova dimenzija podešava se parametrom `figsize`, čija se vrednost navodi u obliku uređenog para (`širina, visina`), pri čemu su obe vrednosti izražene u inčima.
- Metoda `plot.bar()` prikazuje grafik sa stubićima čije visine odgovaraju vrednostima iz strukture na koju je funkcija primenjena.
- Notacija `[::-1]` označava iščitavanje vrednosti počev od poslednje.

## 2. Osnovni pojmovi

### 2.1. Tipovi obeležja

Obeležja, karakteristike kojima opisujemo uzorke, mogu se podeliti u dve grupe, a to su numerička i kategorička obeležja. Numerička obeležja imaju numeričke, odnosno, brojne vrednosti. Primeri su: krvni pritisak, starost, kvadratura kuće, temperatura, itd. S druge strane, kategorička obeležja imaju vrednosti koje predstavljaju kategorije. Primeri su: pol (muški ili ženski), grad (Novi Sad, Čačak, Niš, Subotica ...), pušač (jesti ili nije), dan u nedelji (ponedeljak, utorak, sreda ...), itd. Iako i vrednosti kategoričkih obeležja mogu biti označene numeričkim vrednostima (npr. pol: 0 ili 1), to ipak ne znači da je obeležje numeričko po svojoj prirodi. Neki algoritmi zahtevaju da sva obeležja imaju numeričke vrednosti kako bi se model mogao kreirati (npr. linearna regresija), a nekada su algoritmi dizajnirani tako da mogu da rade i sa kategorijama (npr. stabla odluke). Sa druge strane, prilikom implementacije algoritama uglavnom se zahteva da prosleđena obeležja imaju numeričke vrednosti. To je slučaj i u biblioteci `Scikit-learn`, koja će u ovom praktikumu biti korišćena za implementaciju algoritama mašinskog učenja.

Postoje razni načini da se vrednosti kategoričkih obeležja pretvore u numeričke vrednosti i svaki način ima prednosti i mane. Svakoj kategoriji potrebno je, dakle, dodeliti neku brojnu vrednost, i taj zadatak postaje jednostavniji ako među kategorijama postoji logičan poredak. Na primer, ako je obeležje *kvalitet usluge*, koji može biti *loš*, *osrednji* i *dobar*, kategorije se mogu zameniti brojevima 0, 1 i 2, respektivno. Nadalje neće imati velik uticaj da li su to npr. brojevi 0, 1 i 2 ili 1, 2 i 3. Za neke kategorije utvrđivanje logičnog poretka nije jednoznačno ili nije moguće. Na primer, ako je u pitanju obeležje *grad* u problemu predviđanja cene kuće, kategorije bi se mogle poslagati od onog grada u kom je najmanja potražnja za kućama, do onog u kom je potražnja najveća, pa dodeliti vrednosti od 1 do ukupnog broja kategorija  $K$ . Međutim, i kada je takav pristup moguć, za njega je često potrebno domensko znanje, što bi u ovom slučaju značilo da je potreban ekspert iz oblasti prodaje nekretnina. Ukoliko ne postoji domensko znanje ili dostupan ekspert, jedna od ideja je da se pronađe srednja vrednost obeležja koje se predviđa (u problemu predviđanja cene kuća, to je cena kuće) po kategorijama, te da se vrednost 1 dodeli kategoriji koja ima najmanju srednju vrednost obeležja koje se predviđa, a vrednost  $K$  kategoriji sa najvećom srednjom vrednošću obeležja koje se predviđa. Ovakav pristup očuvaće broj obeležja i jednostavan je, iako redosled kategorija može značajno da utiče na odnos sa drugim obeležjima, što može ponekad da navede

na pogrešne zaključke. Na primer konverzija kategorija obeležja **kvalitet** u numeričke vrednosti na osnovu srednje vrednosti obeležja **cena** data je u sledećem kodu.

```
x1 = x.drop(['cena'], axis=1)
prosecna_cena = x.groupby('kvalitet').mean()['cena']
redosled = prosecna_cena.sort_values().index.values
x1['kvalitet'] = x1['kvalitet'].astype('category')
x1['kvalitet'] = x1['kvalitet'].cat.reorder_categories
    (redosled, ordered=True)
x1['kvalitet'] = x1['kvalitet'].cat.codes
```

Drugi često korišćen pristup jeste formiranje tzv. *dummy* varijabli ili kodovanje *one-hot* vektorima. Ideja je da se od obeležja  $p$  koje ima  $K$  mogućih vrednosti (kategorija) formira  $K$  novih obeležja, pri čemu svako predstavlja jednu kategoriju. U tom slučaju za svaki uzorak  $K - 1$  novih obeležja ima vrednost 0, dok samo jedno novo obeležje koje ukazuje na kategoriju ovog uzorka (odnosno stvarnu vrednost obeležja  $p$  za dati uzorak) ima vrednost 1. Lako se zaključuje da je jedno od  $K$  novih obeležja redundantno, jer slučaj kada svih  $K - 1$  obeležja ima vrednost nula, podrazumeva da  $K$ -to obeležje mora imati vrednost 1 i može da koduje taj slučaj. Iz tog razloga je preporučljivo da se jedno od  $K$  obeležja izostavi. Ovakvim pristupom se izbegava potreba za domenskim znanjem i problem eventualno pogrešnog redosleda kategorija, ali ako obeležje ima mnogo kategorija, ovakav pristup može značajno da poveća dimenzionalnost. Na primer pretvaranje kategoričkog obeležja **kvalitet** sa  $K$  kategorija u  $K - 1$  binarno obeležje izvršeno je u sledećem kodu.

```
x1 = x.drop(columns=['kvalitet'])
x_temp = pd.get_dummies(x['kvalitet'], prefix='kvalitet')
x1 = pd.concat([x1, x_temp.iloc[:, :-1]], axis=1)
```

### 2.2. Normalizacija obeležja

Normalizacija obeležja podrazumeva skaliranje vrednosti obeležja i predstavlja važan korak u predobradi podataka za primenu u algoritmima mašinskog učenja. Nekada je korisna za ubrzavanje obuke (npr. algoritam opadanja gradijenta), a nekada je neophodna da bi se obeležja skalirala na isti opseg vrednosti i time se sprečila mogućnost odlučivanja na osnovu manjeg broja obeležja koja variraju u većem opsegu vrednosti (npr. metoda  $k$  najbližih suseda). Postoji

mnogo različitih metoda normalizacije, a jedna od njih je normalizacija vrednosti obeležja na opseg  $[0,1]$  korišćenjem sledeće formule:

$$\mathbf{x}_i^{norm} = \frac{\mathbf{x}_i - x_i^{min}}{x_i^{max} - x_i^{min}}, \quad (2.1)$$

gde je  $\mathbf{x}_i$  vektor vrednosti  $i$ -tog obeležja za svih  $N$  uzoraka, dok su  $x_i^{max}$  i  $x_i^{min}$  maksimalna i minimalna vrednost  $i$ -tog obeležja u datom skupu. Druga često korišćena metoda je Z-normalizacija ili standardizacija, gde se obeležja normalizuju tako da imaju srednju vrednost 0 i standardnu devijaciju 1, korišćenjem sledeće formule:

$$\mathbf{x}_i^{std} = \frac{\mathbf{x}_i - \mu_i}{\sigma_i}, \quad (2.2)$$

gde je  $\mathbf{x}_i$  vektor vrednosti  $i$ -tog obeležja za svih  $N$  uzoraka, dok su  $\mu_i$  i  $\sigma_i$  srednja vrednost i standardna devijacija  $i$ -tog obeležja, respektivno. Treća česta vrsta normalizacije je svođenje na jediničnu normu. Za vektor  $\mathbf{x}$   $p$ -norma definiše se sa:

$$\|\mathbf{x}\|_p = \left( \sum_{i=1}^D |x_i|^p \right)^{\frac{1}{p}}. \quad (2.3)$$

Kada je vrednost  $p$  jednaka 1, dobija se apsolutna norma, poznata i kao  $L_1$  norma, a kada je  $p$  jednako 2, dobija se euklidska ili  $L_2$  norma. Normalizacija svođenjem na jediničnu normu data je sa:

$$\mathbf{x}_i^{norm} = \frac{\mathbf{x}_i}{\|\mathbf{x}_i\|_p}. \quad (2.4)$$

Budući da je svako obeležje reprezentovano vektorom, često će se sretati pojam norme kao metrike (funkcije rastojanja) i mere veličine (intenziteta) vektora.

Ne postoji univerzalno pravilo kada treba koristiti koju vrstu normalizacije, već to zavisi od same baze podataka koja se koristi i od algoritama mašinskog učenja koji će biti primjenjeni. Pri normalizaciji, parametre za normalizaciju ( $x_i^{max}$  i  $x_i^{min}$  ili  $\mu_i$  i  $\sigma_i$ ) treba izračunati koristeći samo uzorke za obuku i sačuvati ih, jer će se istim parametrima skalirati i uzorci pri obuci, kao i uzorci pri testiranju i kasnije primeni obučenog modela. Na ovaj način se postiže da uzorci odvojeni za testiranje nemaju uticaj na postupak obuke modela ni u ovom segmentu.

U okviru `Scikit-learn` biblioteke postoje implementirane klase `MinMaxScaler`, `StandardScaler` i `Normalizer` za potrebe normalizacije na

opseg  $[0,1]$ , standardizacije i normalizacije normama, respektivno, i nalaze se u modulu `preprocessing`. Za inicijalizaciju se koriste metode čiji se nazivi poklapaju sa nazivima klasa, dok se za izračunavanje parametara za normalizaciju koristi metoda `fit`, a za skaliranje vrednosti metoda `transform`. Klasa `MinMaxScaler` sadrži parametar za inicijalizaciju `feature_range` koji služi za podešavanje opsega na koji će obeležja biti normalizovana, a podrazumevana vrednost parametra je  $(0, 1)$ , odnosno opseg  $[0,1]$ . Klasa `StandardScaler` ima ulazne parametre `with_mean` i `with_std` koji mogu imati vrednosti `True` ili `False` i određuju da li da se vrši standardizacija (oba parametra su `True`, što je i podrazumevana vrednost), samo centriranje uzorka (vrednost parametra `with_std` je `False`) ili samo podešavanje jedinične varijanse (vrednost parametra `with_mean` je `False`). Klasa `Normalizer` sadrži parametar `norm` čijom postavkom na vrednost `l1`, `l2` ili `max` se bira norma. Odabirom `max`, vrednosti će biti skalirane na maksimalne absolutne vrednosti. U nastavku je dat primer koda za standardizaciju obeležja.

```
s=StandardScaler()
s . fit (X_train)
X_train_std = s . transform (X_train)
X_test_std = s . transform (X_test)
```

### 2.3. Izbor modela

Prilikom obuke modela klasifikatora ili regresora potrebno je proceniti (estimirati) stvarnu stopu greške modela na osnovu ograničenog broja dostupnih uzoraka. S obzirom na to da će se model bolje obučiti na većem broju uzoraka, očekivan pristup je da se svi dostupni uzorci iskoriste za obuku modela. Međutim, ako svi uzorci učestvuju u obuci modela, a potom se performanse modela procenjuju na istim uzorcima, procenjena greška biće manja od stvarne greške ovog modela, jer su parametari modela utvrđeni upravo minimizacijom greške nad uzorcima za obuku. Dodatno, ukoliko je dostupan skup uzoraka relativno mali, a model koji se koristi kompleksan (odnosno, opisuje se velikim brojem parametara), prilikom obuke modela, zbog njegove velike fleksibilnosti, može se postići izuzetno mala greška na skupu uzoraka korišćenom za obuku. U ovom slučaju, smatra se da se model „natprilagodio“ podacima za obuku (engl. *overfitting*) i u opštem slučaju može se očekivati velika greška predviđanja za nove, neviđene uzorke. Da bi se objektivno procenio kvalitet obučenog modela, potrebno je raspolagati uzorcima koji nisu korišćeni za obuku modela, odnosno,

određivanje vrednosti njegovih parametara. Za model koji ima malu grešku predviđanja na nevidenom skupu uzoraka kaže se da ima dobru moć *generalizacije*, odnosno, dobre performanse.

Prilikom primene neke od metoda mašinskog učenja, obično postoji mogućnost izbora vrednosti parametara koji približnije određuju model koji će biti korišćen. Na primer, ukoliko je potrebno proceniti vrednost nepoznate funkcije u proizvoljnoj tački, potrebno je doneti odluku o redu modela, odnosno hoće li se predviđanje vršiti polinomom reda 2, 3 ili nekog višeg reda. U ovom slučaju koeficijenti polinoma su nepoznati parametri modela koji se određuju u postupku obuke modela, a red polinoma ukazuje na to koji oblik funkcije se koristi i predstavlja tzv. *hiperparametar* ovog modela. Prema tome, pre postupka obuke modela, potrebno je odrediti i hiperparametre modela. Time se precizno definišu parametri modela, čije je vrednosti potrebno proceniti u postupku obuke. Kvalitet obučenog modela zavisi i od dostupnog skupa uzoraka i od izbora modela i njegovih hiperparametara. U primeru sa određivanjem vrednosti funkcije, nakon određivanja reda polinoma  $p$  kao hiperparametra modela, radi se procena vrednosti ukupno  $p$  koeficijenata polinoma tokom obuke.

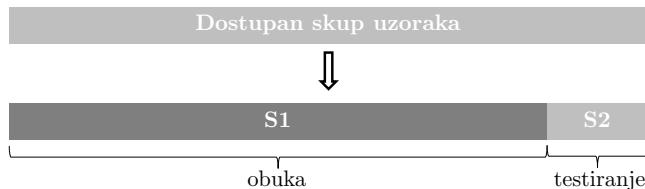
## 2.4. Procena greške na nezavisnom test skupu

U cilju što objektivnije procene kvaliteta obučenog modela predlaže se nekoliko pristupa proceni stavarne stope greške datog modela. Prvi pristup koji će biti pomenut je *holdout*, odnosno procena greške na nezavisnom skupu uzoraka. Ovakav pristup podrazumeva podelu dostupnog skupa uzoraka na dva dela (Slika 2.1), pri čemu veći deo uzoraka čini skup za obuku, a manji deo uzoraka predstavlja test skup, koji služi isključivo za procenu stope greške obučenog modela. U slučaju da je dostupno malo podataka, korišćenjem *holdout* pristupa broj uzoraka namenjen obuci se dodatno smanjuje, što utiče na kvalitet obučenog modela. Pored toga, uzorci koji ulaze u test skup se obično biraju na slučaj, zbog čega je moguće da se u tom skupu nađu vrlo specifični uzorci i da procenjena greška usled toga bude ili vrlo mala ili vrlo velika. Kada je skup podataka za obuku dovoljno velik, test skup će sačinjavati takođe više uzoraka izabranih na slučaj, čime se navedeni problem prevaziđa. Dodatni problem u ovom pristupu predstavlja i potreba da se izaberu što adekvatnije vrednosti hiperparametara modela (npr. pre formiranja stabla odluke potrebno je definisati minimalnu veličinu čvora). U praksi ove vrednosti nisu unapred poznate i zato je potrebno odrediti sa kojim vrednostima hiperparametara model postiže najbolje rezultate. Proveru tačnosti modela dobijenih na osnovu različitih vrednosti hiperparametara moguće je vršiti samo na nezavisnom test skupu jer su

## 2 OSNOVNI POJMOVI

---

podaci iz dostupnog skupa uzoraka već iskorišćeni da bi se svaki od tih modela obučio. Konačan model formiraće se na osnovu vrednosti hiperparametara koje su na test skupu dale najbolji rezultat, iako treba primetiti da u tom slučaju više nije moguće objektivno proceniti tačnost konačnog modela. Razlog je upravo to što su raspoloživi uzorci iskorišćeni ili za obuku modela ili za izbor optimalnih vrednosti hiperparametara, tako da više nisu „novi“.



Slika 2.1: Procena greške modela sa podelom dostupnog skupa uzoraka na dva dela.

Za implementaciju podele dostupnog skupa uzoraka na dva dela u `Scikit-learn` biblioteci postoji funkcija `train_test_split` u modulu `model_selection`. Ova funkcija pored obavezognog ulaznog parametra koji predstavlja niz koji će biti podeljen na dva podskupa ima i nekoliko opcionih ulaznih parametara, a to su:

- **test\_size**: ako je vrednost parametra broj između 0 i 1, predstavljaće ideo uzoraka koji će biti smešteni u test skup; ako je vrednost celobrojna, predstavljaće broj uzoraka koji će biti smešteni u test skup; ako parametar nije podešen, biće jednak  $1 - \text{train\_size}$ , a ako je parametar `train_size` nepodešen, dodeliće se vrednost 0,25.
- **train\_size**: ako je vrednost parametra broj između 0 i 1, predstavljaće ideo uzoraka koji će biti smešteni u skup za obuku; ako je vrednost celobrojna, predstavljaće broj uzoraka koji će biti smešteni u skup za obuku; ako parametar nije podešen, biće jednak  $1 - \text{test\_size}$ .
- **shuffle**: ukoliko je vrednost parametra `False`, u test skup biće smešteno poslednjih `test_size` uzoraka iz baze, a ako je vrednost `True`, `test_size` uzoraka biće na slučaj odabранo iz baze i smešteno u test skup.
- **random\_state**: parametar se koristi radi ponovljivosti izlaza generatora slučajnih brojeva, tj. njegova jedinstvena vrednost obezbeđuje uvek isti

pseudoslučajan niz brojeva koji se koristi pri podeli podataka na podskupove ako je parametar `shuffle` postavljen na `True`; vrednost parametra može se podesiti na bilo koju celobrojnu vrednost, a ako se ne podesi, vrednost će pri svakom pokretanju biti dodeljena na slučaj;

- `stratify`: ovaj parametar je značajan kada se rešava problem klasifikacije i može se postaviti na `True` samo ako je `shuffle` postavljeno na `True`, čime se omogućava balansirana podela, tj. da odnos zastupljenosti klase u podskupovima bude isti.

Primer podele dostupnog skupa uzoraka na dva podskupa, tako da se u test skupu nađe 10% uzoraka:

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.1, random_state=15, stratify=y)
```

Ako se podaci podele kao u primeru datom u kodu, obuku modela treba izvršiti pomoću `X_train` i pripadajućih klasnih labela `y_train`, a procenu performansi modela poređenjem tačnih vrednosti izlazne promenljive za test podatke `y_test` i vrednosti koje su predviđene modelom za `X_test`.

## 2.5. Trostruka podela podataka

Test skup je uveden kako bi se performanse obučenog modela procenile na podacima koje model nije imao priliku da vidi pri obuci. Ipak, ovakav pristup i dalje može da da lošu procenu performansi modela iz više razloga koji su navedeni u 2.4. Poslednji pomenut problem o natprilagođenju hiperparametara skupu na kojem se procenjuju performanse obučenog modela može se prevazići podelom dostupnog skupa uzoraka na tri disjunktna podskupa namenjena za obuku, validaciju i testiranje, respektivno. Model se obučava na skupu za obuku, dok se provera performansi obučenog modela sa raznim vrednostima hiperparametara vrši na validacionom skupu i na osnovu dobijenih rezultata podešavaju se hiperparametri (Slika 2.2, korak 1). Kada se utvrdi koje vrednosti hiperparametara rezultuju modelom sa najboljim performansama na validacionom skupu, skupovi za obuku i validaciju se spajaju te se izabrani model obučava korišćenjem uzorka iz tako dobijenog skupa (Slika 2.2, korak 2). Konačne performanse modela procenjuju se na osnovu testiranja na test skupu i nije dozvoljeno bilo kakvo dalje podešavanje modela. Ovakav pristup i dalje ne rešava problem eventualne specifičnosti uzorka u test skupu (pa i u

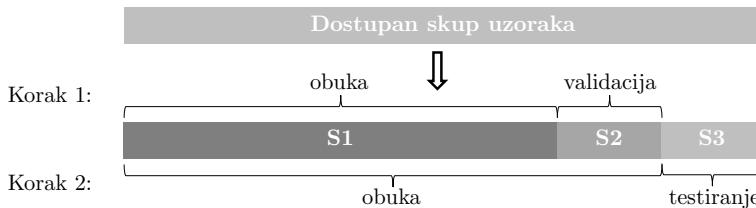
validacionom) kao i problem što se neki uzorci koriste samo za testiranje, ali daje nepristrasnu procenu performansi modela.

Za implementaciju validacionog postupka sa podelom dostupnog skupa uzoraka na tri dela ne postoji posebna funkcija u **Scikit-learn** biblioteci nego je moguće dva puta uzastopno upotrebiti podelu skupa na dva podskupa funkcijom `train_test_split` što je prikazano u kodu koji sledi.

```
# podela podataka na podatke za obuku i test
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.1, random_state=10, stratify=y)

# izdvajanje dela iz podataka za obuku za validaciju
X_train1, X_val, y_train1, y_val = train_test_split(X_train,
                                                    y_train, test_size=0.1, random_state=10, stratify=y_train)
```

Ako se podaci podele kao u primeru datom u kodu, obuka i testiranje radi utvrđivanja adekvatnih hiperparametara treba da se izvrši pomoću `X_train1` i `X_val`, a obuka konačnog modela i procena njegovih performansi sa `X_train` i `X_test`.

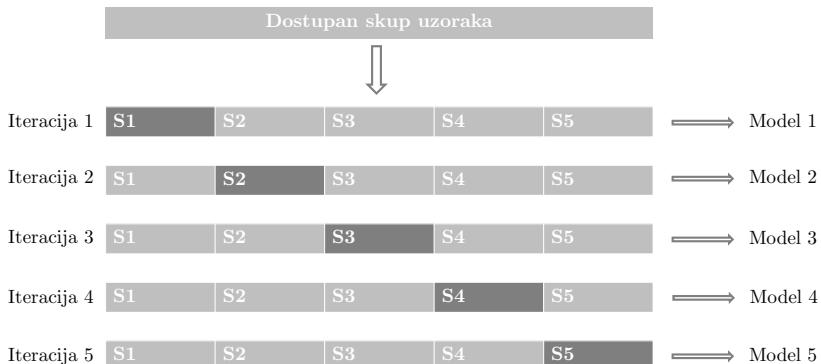


**Slika 2.2:** Procena stope greške podelom dostupnog skupa uzoraka na tri dela.

## 2.6. Unakrsna validacija sa $K$ particija

Prevazilaženje problema korišćenja nekih uzoraka samo u svrhu testiranja kao i natprilagođenje jednom od podskupova uzoraka prevazilazi se upotrebom unakrsne validacije (engl. *cross-validation*). Ovaj pristup se zasniva na principu podele dostupnog skupa uzoraka na  $M$  disjunktnih podskupova (rutinski korišćene vrednosti za  $M$  su 5 ili 10). Kada je  $M$  jednako ukupnom broju uzoraka, radi se o unakrsnoj validaciji sa jednim izdvojenim elementom (engl. *leave-one-out*). Kod unakrsne validacije proces obuke i testiranja za svaki model (sa određenim vrednostima hiperparametara) ponavljaju se  $M$  puta, tako što se

u svakoj od iteracija  $M - 1$  podskupova koristi za obuku modela, dok se testiranje izvršava na preostalom podskupu (Slika 2.3). Na ovaj način svaki od uzoraka se više puta koristi za treniranje i samo jednom za testiranje. Procena greške za model sa određenim vrednostima hiperparametara dobija se kao srednja vrednost procena dobijenih prilikom testiranja  $M$  treniranih modela u postupku unakrsne validacije. Procena greške je samim tim pouzdanija nego u slučaju procene na nezavisnom skupu uzoraka, ali se i trajanje postupka povećava  $M$  puta. Na ovaj način utvrđuje se model i vrednosti hiperparametara, dok se konačni model dobija ponovnim treniranjem nad celokupnim skupom podataka. Metoda unakrsne validacije je preporučljiva naročito kada je broj dostupnih uzoraka manji. U slučaju dovoljno velikog broja dostupnih uzoraka, preporučuje se korišćenje metode unakrsne validacije u kombinaciji sa podelom skupa uzoraka na tri dela. Prvo se izdvoji skup za testiranje, pa se unakrsna validacija radi sa preostalim podacima, a performanse konačnog modela se procenjuju na izdvojenom test skupu.



**Slika 2.3:** Ilustracija unakrsne validacije sa pet podskupova. Svetlosivom bojom označeni su uzorci koji se koriste za obuku, a tamnosivom uzorci za testiranje.

U cilju realizacije unakrsne validacije potrebno je podeliti raspoloživi skup podataka i za to se koriste sledeće klase iz `Scikit-learn` biblioteke: `KFold`, `StratifiedKFold` ili `LeaveOneOut` iz modula `model_selection`.

- `KFold` je klasa koja služi za podelu dostupnog skupa uzoraka na  $M$  (približno) jednakih disjunktnih delova. Pri inicijalizaciji (metoda `KFold`) se koristi parametar `n_split`, koji određuje na koliko podskupova se dele podaci, i parametar `shuffle`, koji određuje da li će se podela raditi na slučaj

ili prema redosledu iz dostupnog skupa uzoraka. Potom se konkretna podela vrši metodom `split` nad inicijalizovanim objektom prosleđujući uzorke. Izlazi ove podele su indeksi uzoraka za obuku i uzoraka za testiranje. Dakle, skup za obuku se dobija indeksiranjem skupa podataka indeksima dodeljenim za obuku, dok se test skup dobija indeksiranjem skupa podataka indeksima dodeljenim za testiranje (analogno se dobijaju odgovarajuće labele).

- `StratifiedKFold`, za razliku od `KFold`, vodi računa o tome da očuva zastupljenost svake klase podataka u svakom od  $M$  podskupova, što je čini pogodnom za primenu u klasifikaciji, pogotovo u slučaju neravnomjerne raspodele uzoraka po klasama. Ova klasa ima iste parametre za inicijalizaciju i koristi se na isti način kao i `KFold`, uz jednu bitnu razliku: pri pozivu metode `fit` prosleđuju se i klasne labele uzoraka kako bi podela na podskupove bila balansirana.
- `LeaveOneOut` klasa nema parametre za inicijalizaciju. Koristi se na isti način kao i `KFold`, i ekvivalentna je sa `KFold(n_splits=N)`, gde je  $N$  jednako ukupnom broju uzoraka u skupu za obuku.

Primer unakrsne validacije sa pet podskupova:

```
skf = StratifiedKFold(n_splits=5)
indexes = skf.split(X, y)
for train_index, test_index in indexes:
    X_train, X_test = X[train_index, :], X[test_index, :]
    y_train, y_test = y[train_index], y[test_index]
    classifier = #inicijalizacija klasifikatora
    classifier.fit(X_train, y_train) #obuka
    y_predicted = classifier.predict(X_test) #testiranje
```

### 2.7. Mere uspešnosti klasifikatora

U postupku testiranja klasifikatora za svaki test uzorak model predviđa odgovarajući izlaz – klasnu labelu. Pored toga, za svaki test uzorak raspolaže se i informacijom o stvarnoj (tačnoj) labeli koja je obezbeđena za raspoloživ skup podataka, pa samim tim i za test uzorke. Kako bi se procenile performanse klasifikatora potrebno je uporediti predviđene i stvarne labele. Ukoliko je u pitanju binarni problem, jedna klasa se proglašava pozitivnom (oznaka 1), a druga negativnom (oznaka 0) i definišu se sledeći pojmovi koji zbirno opisuju tačnu, odnosno pogrešnu klasifikaciju kao i vrstu greške do koje je došlo:

- stvarni pozitiv (engl. *true positive* – TP) je onaj uzorak čija i stvarna i predviđena labela imaju vrednost 1; dakle uzorak koji potiče iz pozitivne klase i klasifikator je ispravno predvideo da pripada pozitivnoj klasi.
- stvarni negativ (engl. *true negative* – TN) je onaj uzorak čija i stvarna i predviđena labela imaju vrednost 0; dakle uzorak koji potiče iz negativne klase i klasifikator je ispravno predvideo da pripada negativnoj klasi.
- lažni pozitiv (engl. *false positive* – FP) je onaj uzorak čija stvarna labela ima vrednost 0, a predviđena labela ima vrednost 1; dakle uzorak koji potiče iz negativne klase, a klasifikator je greškom predvideo da pripada pozitivnoj klasi.
- lažni negativ (engl. *false negative* – FN) je onaj uzorak čija stvarna labela ima vrednost 1, a predviđena labela ima vrednost 0; dakle uzorak koji potiče iz pozitivne klase, a klasifikator je greškom predvideo da pripada negativnoj klasi.

Za uporedni prikaz uskladenosti stvarnih i predviđenih labela koristi se matrica konfuzije. U matrici konfuzije pojedine vrste odgovaraju stvarnim vrednostima labela, dok pojedine kolone odgovaraju predviđenim vrednostima (oznaka klase koju je uzorku dodelio klasifikator). Kolone i vrste mogu biti i obrnuto definisane, te je uobičajeno da se prilikom prikaza matrice naznače stvarne i predviđene vrednosti labela. Iz matrice konfuzije jasno se vidi koje klase klasifikator meša, odakle potiče i naziv ove matrice. Na glavnoj dijagonali matrice konfuzije nalaze se ispravno klasifikovani uzorci. Matrica konfuzije je kvadratna i njena dimenzija odgovara broju klasa. Na Slici 2.4 prikazana je matrica konfuzije za binarnu klasifikaciju (dve klase), od kojih je jedna od klasa proglašena za pozitivnu, a druga za negativnu. Na osnovu matrice konfuzije računaju se mere uspešnosti klasifikatora, a to su: tačnost, preciznost, osetljivost (odziv), specifičnost i F-mera, definisane u Tabeli 2. Veća vrednosti mera ukazuju na bolje performanse, a značaj pojedinih mera kao što su osetljivost i specifičnost, zavisi od problema koji se rešava, i u skladu sa time se prave određeni kompromisi. Na primer, u binarnoj klasifikaciji bolestan (1) ili zdrav (0), veća je greška bolesnog ispitanika svrstati u klasu zdravih (FN), nego zdravog u klasu bolesnih (FP), stoga je u ovom problemu mera osetljivosti klasifikacije značajnija od mere specifičnosti.

Često je u problemima binarne klasifikacije različita cena pogrešne klasifikacije pozitiva u negativne i obrnuto. Samim tim nekada je bolje povećati

## 2 OSNOVNI POJMOVI

---

		predviđeni	
		poz.	neg.
stvarni	poz.	TP	FN
	neg.	FP	TN

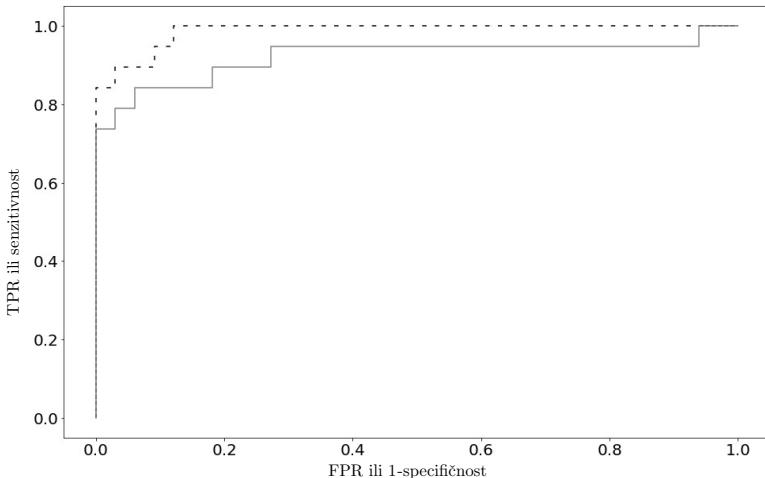
**Slika 2.4:** Definicija stvarnih i lažnih pozitiva i negativa predstavljena kroz matricu konfuzije za dve klase.

Mera	Formula	Definicija
Tačnost	$\frac{TP+TN}{TP+TN+FP+FN}$	Udeo ispravno klasifikovanih uzoraka u celoj populaciji.
Preciznost	$\frac{TP}{TP+FP}$	Udeo ispravno klasifikovanih uzoraka među uzorcima predviđenim kao pozitivnim.
Osetljivost	$\frac{TP}{TP+FN}$	Udeo ispravno klasifikovanih uzoraka iz klase pozitiva.
Specifičnost	$\frac{TN}{TN+FP}$	Udeo ispravno klasifikovanih uzoraka iz klase negativa.
F-mera	$2 \cdot \frac{\text{preciznost} \cdot \text{osetljivost}}{\text{preciznost} + \text{osetljivost}}$	Harmonijska sredina preciznosti i osetljivosti.

**Tabela 2:** Mere uspešnosti klasifikatora sa dve klase.

osetljivost na račun specifičnosti ili obrnuto, što se može učiniti podešavanjem praga za donošenje odluke u algoritmima koji kao izlaz daju verovatnoću pripadnosti klasi. Odnos osetljivosti i specifičnosti, tačnije zavisnost udela ispravno klasifikovanih pozitiva u svim pozitivima (engl. *true positive rate* – TPR) od udela lažnih pozitiva u svim negativima (engl. *false positive rate* – FPR), što je isto što i *1-specifičnost*, za različite pragove odlučivanja grafički se predstavlja pomoću operativne karakteristike, odnosno ROC krive (engl. *receiver operating characteristic*), kao što je prikazano na Slici 2.5). Kako bi se na osnovu ROC krive kvantifikovao uspeh klasifikatora, koristi se vrednost površine ispod krive (engl. *area under the curve* – AUC). Što je ta površina bliža 1, veća je mogućnost da se, izborom odgovarajućeg praga odlučivanja, radna tačka klasifi-

fikatora postavi bliže idealnom položaju, a to je ( $TPR = 1, FPR = 0$ ).



**Slika 2.5:** Primeri ROC krivih za različite parametre klasifikatora, od kojih se kao bolja pokazuje ona prikazana isprekidanom linijom.

Za prikaz ROC krive koristi se funkcija `roc_curve(y_true, y_score)` iz modula `metrics` biblioteke `Scikit-learn`, gde `y_true` predstavlja originalne klasne labele uzorka, a `y_score` predstavlja verovatnoću koju je klasifikator predviđao za pripadnost uzorka pozitivnoj klasi. Za računanje površine ispod krive koristi se funkcija `roc_auc_score(y_true, y_score)` iz istog modula sa istim parametrima.

Kada postoji više od dve klase, uspešnost klasifikatora i dalje se računa na osnovu matrice konfuzije, ali ne postoji jedinstveni TP, TN, FP i FN nego se oni definišu za svaku od klasa ponaosob, te se i sve mere računaju za svaku klasu ponaosob. Jedinstvene mere uspešnosti klasifikatora najčešće predstavljaju prosek mera po klasi. Ako se posmatra primer tri klase, A, B i C, matrica konfuzije biće dimenzija  $3 \times 3$ . Za klasu A (Slika 2.6, levo), TP će biti oni uzorci koji potiču iz klase A i klasifikovani su u A. FP će biti oni koji ne potiču iz klase A (potiču iz B ili C), a greškom su klasifikovani u A. FN će biti oni koji potiču iz klase A, ali su greškom klasifikovani u B ili C. TN će biti svi oni koji ne potiču iz A i nisu klasifikovani u A. Analogno navedenom, definišu se TP, TN, FP i FN za klasu B (Slika 2.6, sredina), kao i klasu C (Slika 2.6, desno). Kada su za svaku klasu određene vrednosti TP, TN, FP i FN, onda za svaku klasu mogu da se

## 2 OSNOVNI POJMOVI

---

		predvidene					predvidene					
		A	B	C			A	B	C			
prave	A	TP	FN	FN	prave	TN	FP	TN	prave	TN	TN	FP
	B	FP	TN	TN		FN	TP	FN		TN	TN	FP
	C	FP	TN	TN		TN	FP	TN		FN	FN	TP

**Slika 2.6:** Matrica konfuzije za tri klase sa označenim TP, TN, FP i FN za klasu A (levo), klasu B (sredina) i klasu C (desno).

izračunaju tačnost, preciznost, osetljivost, specifičnost i F-mera. Kao konačne mere uspešnosti klasifikatora sa više klasa računaju se prosečna tačnost, kao prosek tačnosti po klasama, i stopa greške, kao prosek greške klasifikacije po klasi (Tabela 3). Za osetljivost, preciznost i F-meru klasifikatora sa više klasa razlikuju se dva tipa mera: mikroprosečne (engl. *micro-average*) i makroposečne (engl. *macro-average*) mere. Mikroposečne mere svode se na sumiranje TP, FP i FN za sve klase, nakon čega se računaju mere uspešnosti na standardan način, dok makroposečne mere predstavljaju proseke mera dobijenih po klasi (Tabela 3). Ukoliko je broj uzoraka u različitim klasama neujednačen, preferiraju se mikroposečne mere.

Iako se sve navedene mere mogu implementirati korišćenjem datih objašnjenja i formula iz tabela, za mnoge postoje i ugrađene funkcije u Scikit-learn biblioteci u okviru modula `metrics`, a neke od njih su:

- Pomoću funkcije `confusion_matrix(y_true, y_pred)` se formira matrica konfuzije, a obavezni ulazni parametri su lista tačnih klasnih labela i lista predviđenih klasnih labela, respektivno.
- Pomoću funkcije `accuracy_score(y_true, y_pred)` se računa tačnost za binarnu klasifikaciju, a obavezni ulazni parametri su isti kao i u prethodno navedenoj funkciji. U slučaju klasifikacionog problema sa više od dve klase, ova funkcija računaće ideo ispravno predviđenih uzoraka, bez obzira na klasu (količnik zbiru brojeva na glavnoj dijagonali matrice konfuzije i ukupnog broja uzoraka).
- Pomoću funkcije `precision_score(y_true, y_pred)` se računa preciznost. Obavezni parametri su isti kao i u prethodno navedenim funkcijama. Ukoliko je reč o klasifikacionom problemu sa više od dve klase, treba proslediti

Mera	Formula
Prosečna tačnost	$\frac{\sum_{i=1}^K \frac{tp_i + tn_i}{tp_i + fn_i + fp_i + tn_i}}{K}$
Stopa greške	$\frac{\sum_{i=1}^K \frac{fp_i + fn_i}{tp_i + fn_i + fp_i + tn_i}}{K}$
Preciznost $_{\mu}$	$\frac{\sum_{i=1}^K tp_i}{\sum_{i=1}^K (tp_i + fp_i)}$
Osetljivost $_{\mu}$	$\frac{\sum_{i=1}^K tp_i}{\sum_{i=1}^K (tp_i + fn_i)}$
F-mera $_{\mu}$	$2 \cdot \frac{Preciznost_{\mu} Osetljivost_{\mu}}{Preciznost_{\mu} + Osetljivost_{\mu}}$
Preciznost $_M$	$\frac{1}{K} \sum_{i=1}^K \frac{tp_i}{tp_i + fp_i}$
Osetljivost $_M$	$\frac{1}{K} \sum_{i=1}^K \frac{tp_i}{tp_i + fn_i}$
F-mera $_M$	$2 \cdot \frac{Preciznost_M Osetljivost_M}{Preciznost_M + Osetljivost_M}$

**Tabela 3:** Mere uspešnosti klasifikatora sa više od dve klase. Oznaka u supskriptu  $M$  označava makromeru, a  $\mu$  mikromeru.  $K$  predstavlja ukupan broj klasa, a  $tp_i, fp_i, tn_i, fn_i$  su stvarni i lažni pozitivi i negativi za  $i$ -tu klasu.

i ulazni parametar `average='micro'` za izračunavanje mikropreciznosti ili `average='macro'` za izračunavanje makropreciznosti.

- Funkcija `recall_score(y_true, y_pred)` koristi se za računanje osetljivosti na isti način kao i `precision_score`.
- Funkcija `f1_score(y_true, y_pred)` koristi se za računanje F-mere na isti način kao i `precision_score`.

## 2.8. Mere uspešnosti regresora

Kod regresionih modela predviđa se vrednost određene kontinualne promenljive. Postoji niz mera za procenu tačnosti obučenog regresionog modela. Najčešće korišćena mera je srednja kvadratna greška (engl. *mean squared error* – MSE). Kvadrat se koristi kako bi se velike greške penalizovale i kako se greške različitog znaka ne bi potirale. Prilikom opisa performansi modela pogodnije je koristiti mere koje se izražavaju u jedinicama izlazne promenljive kao što su koren srednje kvadratne greške (engl. *root mean square error* – RMSE) i srednja apsolutna greška (engl. *mean absolute error* – MAE). Navedene mere definisane su sledećim formulama:

$$MSE = \frac{1}{N} \sum_{n=1}^N (h(\mathbf{x}^{(n)}) - y^{(n)})^2, \quad (2.5)$$

$$RMSE = \sqrt{\frac{1}{N} \sum_{n=1}^N (h(\mathbf{x}^{(n)}) - y^{(n)})^2}, \quad (2.6)$$

$$MAE = \frac{1}{N} \sum_{n=1}^N |h(\mathbf{x}^{(n)}) - y^{(n)}|, \quad (2.7)$$

gde je  $N$  broj uzoraka,  $\mathbf{x}^{(n)}$  je  $n$ -ti vektor obeležja,  $y^{(n)}$  je tačna vrednost zavisne promenljive za  $n$ -ti uzorak, dok je  $h(\mathbf{x}^{(n)})$  predviđena vrednost zavisne promenljive za  $n$ -ti uzorak. Pored navedenih mera u čestoj upotrebi su i  $R^2$  skor i rezidualna standardna greška (engl. *residual standard error* – RSE), definisane izrazima:

$$R^2 = \frac{TSS - RSS}{TSS}, \quad (2.8)$$

$$RSE = \sqrt{\frac{1}{N - D - 1} RSS}, \quad (2.9)$$

gde je RSS suma kvadrata razlika tačnih i predviđenih vrednosti zavisne promenljive (engl. *residual sum of squares*), dok TSS (engl. *total sum of squares*) predstavlja sumu kvadrata razlika stvarnih vrednosti i srednje vrednosti zavisne promenljive:

$$RSS = \sum_{n=1}^N (h(\mathbf{x}^{(n)}) - y^{(n)})^2, \quad (2.10)$$

$$TSS = \sum_{n=1}^N (h(\mathbf{x}^{(n)}) - \bar{y})^2, \quad (2.11)$$

gde je  $N$  broj uzoraka, a  $D$  broj obeležja. RMSE i MSE su mere za koje se ne može reći koje konkretne vrednosti se smatraju dobrim ili lošim jer zavise od opsega u kom se kreće zavisna promenljiva.  $R^2$  ukazuje na ideo ukupne varijanse koji obučen model pokriva (objašnjava) i ima maksimalnu vrednost jedan, dok bi se vrednost 0 postigla ukoliko bi model za svaki ulaz predviđao istu vrednost koja odgovara srednjoj vrednosti zavisne promenljive nad skupom za obuku. Veće vrednosti  $R^2$  su bolje, ali ne mora nužno da znači da je model sa visokom vrednošću  $R^2$  dobar, niti da je model sa niskom  $R^2$  merom loš, jer na to utiče mnogo parametara. Kada model vraća istu vrednost bez obzira na ulaz,  $R^2$  je 0. Postoji i prilagođeni  $R^2$  skor (engl.  $R^2$  adjusted), koji uzima u obzir i broj obeležja, jer bi direktno poređenje modela sa različitim brojem obeležja na osnovu  $R^2$  skora bilo nemoguće. Prilagođeni  $R^2$  skor računa se na sledeći način:

$$R_{adj}^2 = 1 - \frac{(1 - R^2)(N - 1)}{N - D - 1}. \quad (2.12)$$

Iako se sve mere uspešnosti regresora mogu implementirati korišćenjem datih objašnjenja i formula, za mnoge postoje i ugrađene funkcije u **Scikit-learn** biblioteci u okviru modula **metrics**, a neke od njih su sledeće:

- MSE mera: funkcija `mean_squared_error(y_true, y_pred)`, gde su obavezni ulazni parametri lista tačnih vrednosti zavisne promenljive i lista predviđenih vrednosti zavisne promenljive, respektivno.
- MAE mera: funkcija `mean_absolute_error(y_true, y_pred)` koja ima iste obavezne ulazne parametre kao i funkcija za računanje MSE.
- $R^2$  skor: funkcija `r2_score(y_true, y_pred)` koja takođe ima iste obavezne ulazne parametre kao i funkcija za računanje MSE.

### 3. Analiza obeležja

Mašinsko učenje je moguće posmatrati kao proces kroz koji se na osnovu ulaznih podataka formira obučen model, koji se kasnije može upotrebiti za rešavanje problema kao što su klasifikacija ili regresija. Pre usvajanja bilo kakvih pretpostavki i izbora i obuke modela potreбno je prvo prikupiti podatke i upoznati se sa njihovom strukturom i statističkim osobinama. Deskriptivna statistika i eksplorativna analiza podataka predstavljaju oblasti koje se bave analizom dostupnih obeležja i vizuelnim pregledom skupa podataka. Pretpostavka je da su podaci već dati u formi matrice, u kojoj svaki red predstavlja uzorak za koji su različita obeležja data po kolonama. U ovoj vežbi opisuju se osnovne metode analize podataka, koju je poželjno izvršiti pre nego što se podaci iskoriste kao ulaz u algoritme mašinskog učenja. Ova analiza doprinosi boljem sagledavanju problema koji treba da se reši, ali ukazuje i na potencijalne propuste u procesu prikupljanja podataka. U prvom delu vežbe biće prikazana moguća rešenja problema nedostajućih podataka. Naime, u slučaju da neka od obeležja za neke uzorke nedostaju, matrica podataka je nepotpuna i zavisno od razloga iz kojih ovi podaci nisu poznati, strategije rešavanja problema nedostajućih podataka će se razlikovati. U drugom delu vežbe analiziraju se statističke osobine pojedinačnih obeležja (univarijantna analiza), kao i odnosa dva ili više obeležja (multivarijantna analiza). U nastavku se ukazuje i na značaj vizualizacije skupa podataka, pre svega u cilju boljeg razumevanja izračunatih statističkih osobina i izbora prikladnog modela.

#### 3.1. Predobrada nepotpunih podataka

Ponekad skup podataka nije potpun i vrednosti pojedinih obeležja nisu poznate za sve uzorke. Postoje razni uzroci pojave da u skupu neke vrednosti nedostaju, i umesto njih obično стоји ознака `NaN` (engl. *not a number*). Postoje tri moguća uzroka nedostatka određenog podatka:

1. Podatak nedostaje na potpuno slučajan način (engl. *missing completely at random* – MCAR), nezavisno od samog uzorka i načina uzorkovanja. Primer može biti upitnik, gde su obeležja odgovori na pitanja. Ako ispitanik slučajno preskoči pitanje i samim tim u uzorku ne postoji zabeležen odgovor, u pitanju je ova vrsta nedostajuće vrednosti. Uzorci koji sadrže nedostajuće podatke predstavljaju slučajan podskup svih uzoraka i ne postoje značajne razlike u vrednostima preostalih, potpunih obeležja.

2. Podatak nedostaje na slučajan način, koji ne zavisi od same vrednosti obeležja, ali jeste povezan sa vrednošću nekog drugog obeležja (engl. *missing at random* – MAR). Na primer, u studijama o depresiji ispitanici određenog uzrasta ili pola obično ređe popune dijagnostička pitanja (što nije u direktnoj vezi sa njihovim mentalnim zdravljem, već pritiskom društva na njih).
3. Podatak nedostaje iz razloga koji nije slučajan, već je povezan sa samom vrednošću obeležja koje nedostaje (engl. *missing not at random* – MNAR). Samim tim, vrednost obeležja koja nedostaje ne može se pretpostaviti na osnovu vrednosti drugih obeležja, što čini da se ovaj tip nedostajućih podataka najteže prevazilazi. Na primeru ankete o primanjima, MNAR tip nedostajućih podataka bio bi prisutan ukoliko bi ispitanici koji smatraju svoja primanja malim sa manjom verovatnoćom odgovarali na pitanje o iznosu primanja. Razlog izostanka odgovora bio bi, dakle, povezan sa samom vrednošću obeležja, u ovom slučaju visinom primanja.

Imajući u vidu različite uzroke nedostatka podataka, postavlja se pitanje koji je najbolji način da se takvi skupovi podataka pripreme za dalju analizu. Odgovor zavisi od uzroka odsustva podatka, koji je često teško, a nekada i nemoguće odrediti. Idealno rešenje je ponoviti merenje i saznati vrednost, međutim to najčešće nije moguće. Najjednostavnije rešenje jeste zanemarivanje onih uzoraka za koje nisu poznate vrednosti svih obeležja, čime u skupu ostaju samo uzorci kod kojih su vrednosti svih obeležja dostupne. Iako je time omogućena analiza različitih uzoraka po svim obeležjima, veličina skupa se smanjuje, a može se uneti i pristrasnost (osim ako nije u pitanju prvi uzrok nedostatka podataka, potpuno slučajan način tj. MCAR). Primera radi, ako bi u upitniku o potrošačkim navikama svi nezaposleni ispitanici ostavili polje namenjeno za iznos plate nepotpunjeno, i ako bi istraživači zbog toga izostavili sve članove navedene grupacije, dobijeni podaci bi predstavljali nerealnu sliku stvarne populacije. Druga mogućnost je ograničavanje na podskup uzoraka za koji su poznate vrednosti obeležja koja se analiziraju, uz zanemarivanje eventualnih nepostojećih vrednosti u drugim, nerazmatranim obeležjima. Naravno, manu ovakvog pristupa je što se može desiti da se za svako obeležje ili podskup obeležja razmatra i različit podskup uzoraka, što čini poređenje različitih analiza nemogućim. Još jedna opcija jeste izostavljanje obeležja za koja vrednosti za mnoge uzorke ne postoje.

Nasuprot eliminaciji uzoraka, jedan od načina za rešavanje problema nedostajućih podataka jeste i dopunjavanje nedostajućih vrednosti na osnovu dostupnih vrednosti istog obeležja kod drugih uzoraka. Jedna mogućnost je da se

### 3 ANALIZA OBELEŽJA

---

nedostajuće vrednosti zamene nekom od mera centralne tendencije (engl. *mean-mode imputation*), što automatski smanjuje varijansu datog obeležja. Takođe, korelacije između obeležja su u ovom slučaju zanemarene. U slučaju numeričkih obeležja, mera centralne tendencije koja se koristi može biti srednja vrednost ili medijana, a ukoliko su obeležja kategorička, upotrebljava se modus, koji odgovara najčešćoj vrednosti obeležja. U ovom slučaju, nadalje se mogu koristiti svi uzorci, ali se smanjuje procena varianse pojedinih obeležja, a i kovarijansa među obeležjima može biti potcenjena jer se prilikom dopune obeležja tretiraju kao nezavisna (iako to ne moraju biti). Osim ovog jednostavnog izbora, vrednost nepoznatog obeležja može se prepostaviti na osnovu dostupnih obeležja korišćenjem regresije (više o linearnoj regresiji u Poglavlju 7). Osim što je ovakva procena sporija, mana ovog pristupa je i moguće natprilagođenje modela. U slučaju predobrade podataka zbog nedostajućih vrednosti, treba proveriti robustnost svih daljih analiza kako bi se potvrdilo da izabrani način korekcije nepotpune matrice podataka nema značajan uticaj na konačan ishod.

Sledi ilustracija predobrade nad skupom podataka učitanim u `DataFrame` strukturu pod imenom `data`. Za proveru nedostajućih vrednosti se koristi funkcija `isnull(X)`, koja vraća logičku jedinicu na mestima gde je vrednost obeležja  $X$  jednaka `NaN`. Ukoliko nedostajuće podatke treba odstraniti, korisna je funkcija `dropna`, koja zavisno od prosleđenog parametra `axis` briše određene uzorce (`axis=0` za redove) ili obeležja (`axis=1` za kolone). Alternativa je dopunjavanje nedostajućih vrednosti obeležja, što se može uraditi funkcijom `fillna`. Prvi argument ove funkcije određuje kojom se vrednošću zamenjuje nedostajuća vrednost određenog obeležja, i u navedenom primeru je to medijana. Za obe navedene funkcije važi da ukoliko se prosledi vrednost argumenta `inplace = True`, funkcija menja sadržaj `DataFrame` strukture nad kojom je pozvana.

```
# ukupan broj nedostajućih vrednosti po obeležjima
print(data.isnull().sum())
# brisanje uzoraka sa nedostajućim vrednostima obeležja
data_delete = data.dropna(axis = 0, inplace = False)
# brisanje obeležja sa nedostajućim vrednostima
data_delete_2 = data.dropna(axis = 1, inplace = False)
# dopuna nedostajućih vrednosti obeležja
data_fill = data.fillna(data.median(), inplace = False)
```

#### 3.2. Univariantna analiza obeležja

Nakon predobrade podataka, koja se svodi na dopunu ili izostavljanje uzoraka i/ili obeležja iz razmatranog skupa, naredni korak u analizi je izračunavanje

osnovnih vrednosti koje opisuju pojedinačna obeležja. Ako posmatramo vektor vrednosti jednog od  $D$  obeležja  $\mathbf{x} = (x^{(1)}, x^{(2)}, \dots, x^{(n)}, \dots, x^{(N)})$ , empirijska raspodela verovatnoće  $f(x)$  može se proceniti na osnovu  $N$  uzoraka:

$$f(x) = \frac{1}{N} \sum_{n=1}^N \mathbb{I}(x^{(n)} = x). \quad (3.1)$$

U ovom izrazu,  $\mathbb{I}$  predstavlja indikatorsku funkciju, koja je jednaka 1 u slučaju kada je uslov u zagradi ispunjen, a inače je jednaka 0. Empirijska kumulativna funkcija raspodele verovatnoće  $F(x)$  izračunava se kao:

$$F(x) = \frac{1}{N} \sum_{n=1}^N \mathbb{I}(x^{(n)} \leq x). \quad (3.2)$$

Naredna korisna informacija jeste opseg vrednosti analiziranog obeležja. Empirijski opseg  $r$  računa se na osnovu maksimuma i minimuma vektora  $\mathbf{x}$ , koji sadrži vrednosti tog obeležja za svih  $N$  uzoraka:

$$r = \max \mathbf{x} - \min \mathbf{x}. \quad (3.3)$$

O podacima se može saznati i kroz mere centralne tendencije. Uzoračka srednja vrednost  $\mu$ , kao procena srednje vrednosti vektora obeležja  $\mathbf{x}$  na osnovu skupa od  $N$  uzoraka, definiše se na sledeći način:

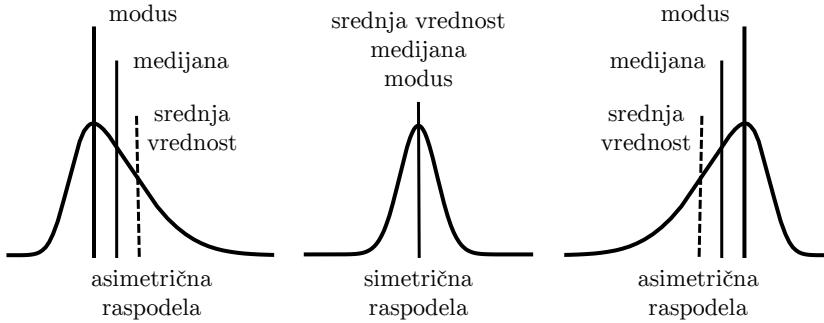
$$\mu = \frac{1}{N} \sum_{n=1}^N x^{(n)}. \quad (3.4)$$

Više o strukturi podataka može se saznati na osnovu vrednosti medijane i modusa. Na primeru niza brojeva, medijana je jednak središnjoj vrednosti u sortiranom nizu, dok modus odgovara vrednosti koja se najčešće pojavljuje u nizu. U opštem slučaju, medijana  $m_c$  i modus *mode* su brojevi određeni sledećim relacijama:

$$P(x \leq m_c) = P(x \geq m_c) = \frac{1}{2}, \quad F(m_c) = 0.5; \quad (3.5)$$

$$\text{mode}(x) = \operatorname{argmax}_x f(x). \quad (3.6)$$

U slučaju parnog broja elemenata vektora, medijana se izračunava kao srednja vrednost dve centralne vrednosti. O tome koliko su vrednosti obeležja rasute



**Slika 3.1:** Poredanje srednje vrednosti, medijane i modusa na primeru simetrične raspodele (sredina) i dve asimetrične raspodele (levo i desno).

može se saznati iz uzoračke varijanse,  $\sigma^2$ :

$$\sigma^2 = \frac{1}{N} \sum_{n=1}^N (x_i^{(n)} - \mu)^2. \quad (3.7)$$

Uzoračka standardna devijacija  $\sigma$  definiše se kao kvadratni koren varijanse  $\sigma^2$ .

Mere centralne tendencije i njihov odnos mogu se lakše razumeti kroz ilustraciju (Slika 3.1). Prikazane su tri raspodele od kojih je jedna simetrična (sredina), te su srednja vrednost, medijana i modus međusobno jednaki. Za druge dve raspodele se može videti da su drugačijeg oblika, što se može izraziti koeficijentom asimetrije  $a$  (engl. *coefficient of skewness*):

$$a = \mathbb{E} \left[ \left( \frac{x - \mu}{\sigma} \right)^3 \right]. \quad (3.8)$$

U slučaju leve asimetrične raspodele sa Slike 3.1 važi da je  $a > 0$  dok u slučaju krajnje desne raspodele važi da je  $a < 0$ . Dodatne informacije o raspodeli nosi i tzv. koeficijent spljoštenosti  $s$  (engl. *coefficient of kurtosis*):

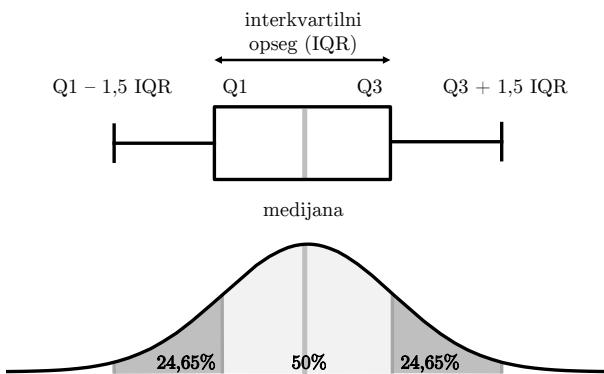
$$s = \mathbb{E} \left[ \left( \frac{x - \mu}{\sigma} \right)^4 \right]. \quad (3.9)$$

Još jedna informativna mera disperzije jeste interkvartilni opseg: razlika trećeg i prvog kvartila (kvartili dele raspodelu na 4 jednakata dela). Prvi ili donji kvartil naziva se i 25. percentilom, a treći ili gornji kvartil naziva se i 75. percentilom. U opštem slučaju,  $M$ -ti percentil je ona vrednost za koju važi da  $M\%$  uzoraka ima vrednost manju od nje. Interkvartilni opseg IQR je stoga određen izrazom:

$$IQR = F(0.75) - F(0.25). \quad (3.10)$$

Raspodela svakog obeležja se može približno predstaviti kroz navedenih pet brojeva (engl. *five-number summary*): medijanu, minimalnu i maksimalnu vrednost, kao i prvi i treći kvartil. Za prikaz nabrojanih statističkih osobina obeležja

može se koristiti tzv. *boxplot* dijagram (Slika 3.2). Za prikaz se koristi funkcija `boxplot` iz biblioteke `Matplotlib`. Na osnovu ovih vrednosti je moguće definisati koje se vrednosti smatraju netipičnim za datu raspodelu obeležja (engl. *outlier*), te potencijalno mogu biti izuzete iz razmatranja jer previše odstupaju od očekivanog (prosečnog) uzorka. Netipičnim vrednostima smatraju se sve one koje se nalaze van unapred utvrđenog intervala, npr. određenog percentila ili van opsega  $\mu \pm 3\sigma$ , gde je  $\sigma$  standardna devijacija. Za normalnu raspodelu važi sledeće pravilo: 68% uzoraka se nalazi u opsegu  $\mu \pm \sigma$ , 95% u opsegu  $\mu \pm 2\sigma$ , a 99.7% u opsegu  $\mu \pm 3\sigma$ . Netipične vrednosti imaju velik uticaj na vrednost opsega i srednju vrednost, a ne utiču na interkvartilni opseg niti na medijanu, zbog čega se interkvantilni opseg i medijana smatraju robustnijim merama u analizi podataka.



Slika 3.2: Ilustracija *boxplot* dijagrama i odnosa sa normalnom raspodelom.

### 3.3. Bivarijantna i multivarijantna analiza obeležja

U prethodnom odeljku analizirana su pojedinačna obeležja i njihove osobine (tzv. univarijantna analiza). Postavlja se pitanje kako se može opisati odnos između dva ili više obeležja (bivarijantna, odnosno multivarijantna analiza). Zavisnost između dva obeležja,  $i$  i  $j$ , može se izraziti kroz uzoračku kovarijansu  $\sigma_{ij}$  i uzoračku korelaciju  $\rho_{ij}$ :

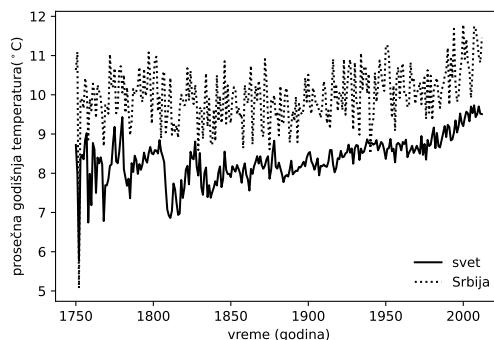
$$\sigma_{ij} = \frac{1}{N} \sum_{n=1}^N (x_i^{(n)} - \mu_i)(x_j^{(n)} - \mu_j), \quad (3.11)$$

### 3 ANALIZA OBELEŽJA

---

$$\rho_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_i^2 \sigma_j^2}}. \quad (3.12)$$

Iz definicije se može uočiti da je korelacija između dva obeležja ekvivalentna kovarijansi odgovarajućih standardizovanih obeležja za  $\sigma_i = \sigma_j = 1$ . Postojanje zavisnosti između dva obeležja može se uočiti i na osnovu njihovog grafičkog prikaza. Na Slikama 3.3 i 3.4 prikazana su dva korelisana obeležja. Na Slici 3.3



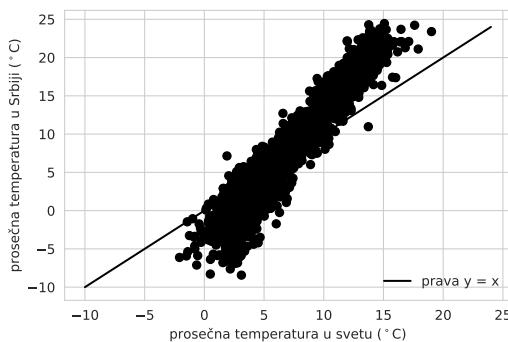
**Slika 3.3:** Zavisnost prosečne godišnje temperature od vremena u svetu i u Srbiji.

može se videti promena prosečne godišnje temperature na svetskom nivou (puna linija) i u Srbiji (ispukljana linija). Sudeći po sličnom ponašanju krivih, može se pretpostaviti da su ova dva obeležja korelisana. Grafik na Slici 3.3 prikazan je pomoću funkcije `plot` iz biblioteke `Matplotlib`:

```
plt.plot(god, np.mean(avgT, axis=0), 'k', label='svet')
plt.plot(god, np.mean(avgT_S, axis=0), ':k', label='Srbija')
plt.legend(loc='lower right', frameon=False)
```

Bolji uvid u njihovu vezu može pružiti tzv. grafik rasipanja (engl. *scatter plot*), prikazan na Slici 3.4 pomoću funkcije `scatter` iz biblioteke `Matplotlib`:

```
plt.scatter(avgT, avgT_S, color='black')
plt.plot(range(-10, 25), range(-10, 25), 'k', label='prava y=x')
ax1.set_xlabel('prosečna temperatura u svetu ($^\circ$C)')
ax1.set_ylabel('prosečna temperatura u Srbiji ($^\circ$C)')
plt.legend(loc='lower right', frameon=False)
```



**Slika 3.4:** Odnos između prosečne godišnje temperature na svetskom nivou i prosečne godišnje temperature u Srbiji.

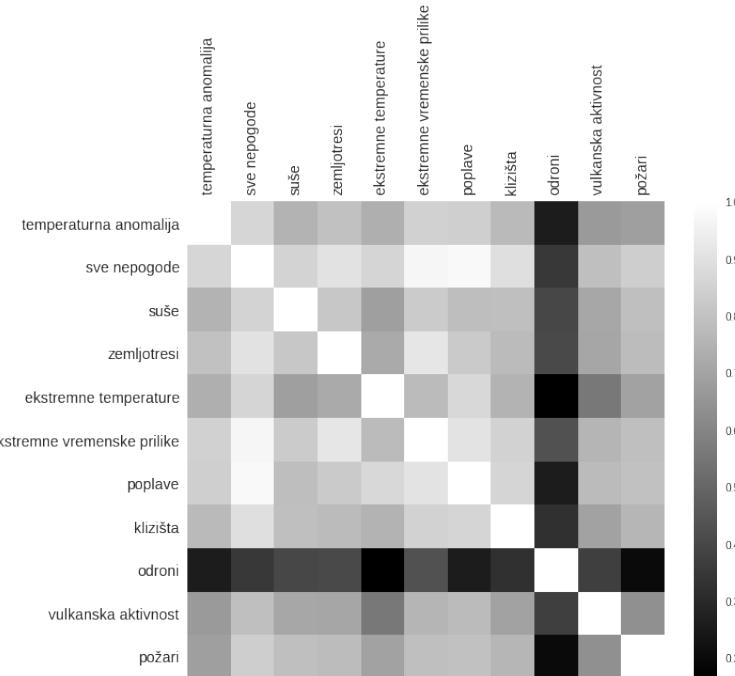
Još jedan način da se prikaže korelacija, pogodan i u slučaju većeg broja obeležja, jeste tzv. topotorna mapa (engl. *heatmap*). Na osama su prikazana različita obeležja, dok boja odgovarajućeg polja odgovara vrednosti njihove međusobne korelacije. Dat je primer na skupu podataka sa prosečnom temperaturom i prirodnim nepogodama (Slika 3.5). Korelaciona matrica izračunava se pozivom funkcije `corr` nad `DataFrame` strukturu, uz prikaz topotnom mapom pomoću funkcije `heatmap` iz biblioteke `Seaborn`, što je prikazano u sledećem kodu:

```
matrica_korelacija = data.corr()
seaborn.heatmap(matrica_korelacija)
```

Procena raspodele verovatnoće jednog obeležja na osnovu skupa uzoraka može se prikazati normalizovanim histogramom (na ordinati su prikazane verovatnoće čiji je zbir jednak 1). U problemu klasifikacije, prikazom raspodela obeležja uporedno za više klase može se utvrditi koliko se obeležja razlikuju zavisno od toga kojoj klasi uzorak pripada. Na primer, potrebno je napraviti sistem za automatsku detekciju vrste sove na osnovu njenog izgleda. Visina kao obeležje može pomoći u odlučivanju da li nepoznati primerak sove pripada vrsti mali čuk (lat. *Glaucidium passerinum*) ili velika ušara (lat. *Bubo bubo*). Na osnovu ovako jasne razlike između malog čuka i velike ušare, model može da nauči ovu pravilnost i uspešno izvrši klasifikaciju posmatrajući samo to jedno obeležje (Slika 3.6). Nasuprot tome, mali čuk i čuk (lat. *Otus scops*) ne mogu

### 3 ANALIZA OBELEŽJA

---

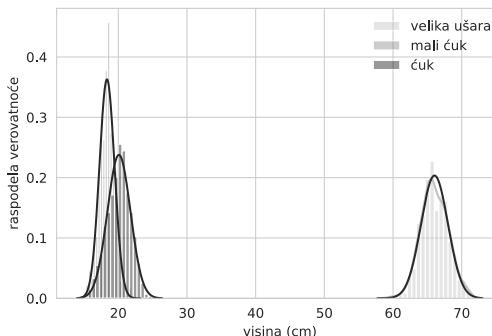


Slika 3.5: Prikaz koeficijenta korelacije po parovima obeležja pomoću toplotne mape.

se pouzdano razlikovati na osnovu visine, jer im se raspodele preklapaju. U tom slučaju je potrebno razmotriti još neko obeležje, kao što je npr. boja perja.

Za prikaz grafika na slici 3.6 je korišćena funkcija `distplot` iz biblioteke Seaborn, koja prikazuje histogram i procenu gustine raspodele verovatnoće. Primeri metoda procene gustine raspodele verovatnoće biće dati u Poglavlju 5.

```
fig , ax = plt.subplots()
seaborn.distplot(o1, color='silver', label='buljina')
seaborn.distplot(o2, color='lightgrey', label='mali čuk')
seaborn.distplot(o3, color='black', label='čuk')
plt.xlabel('visina (cm)'), plt.ylabel('raspodela verovatnoće')
leg = ax.legend(frameon = False)
```



**Slika 3.6:** Prikaz oboležja na primeru visine sove: moguće je razlikovati uzorke iz klase velika ušara od uzoraka iz druge klase, dok na osnovu prikazanog oboležja nije moguće međusobno razlikovati klase mali čuk i čuk.

### 3.4. Zadaci

#### Zadatak 1

Za upoznavanje sa podacima koristiće se skup podataka o prosečnim temperaturama u gradovima širom sveta<sup>1</sup>. Cilj upoznavanja sa podacima je bolje razumevanje oboležja i njihove povezanosti.

1. Prvo treba učitati biblioteke Pandas, Numpy i iz biblioteke Matplotlib kolekciju funkcija pyplot. Zatim treba pomoću funkcije `read_csv` učitati podatke iz datoteke `city_temperature.csv`.
2. Kog formata su podaci? Šta se nalazi po vrstama, a šta po kolonama – šta su uzorci, a šta oboležja? Koliko ima uzoraka, a koliko oboležja? Da li su oboležja numerička ili kategorička i šta to znači? Odgovori na neka od ovih pitanja mogu se dobiti pomoću funkcija `shape` i `dtypes`.
3. Proveriti da li je skup podataka potpun korišćenjem funkcije `isnull`. Kako se može rešiti problem nedostajućih podataka i kog tipa su nedostajući podaci u ovom skupu? Da li je interpolacija pomoću funkcije `fillna` odgovarajuća metoda?

<sup>1</sup>Podaci su dostupni na: <https://www.kaggle.com/sudalairajkumar/daily-temperature-of-major-cities>

### 3 ANALIZA OBELEŽJA

---

4. Izračunati osnovne statističke veličine za svako od obeležja (srednja vrednost, standardna devijacija, minimalna i maksimalna vrednost, medijana, kvartili, interkvartilni opseg). Sve navedene veličine vraća `describe` funkcija pozvana nad izabranim obeležjem. Da li postoje neke nepravilnosti u skupu podataka? Odakle one potiču i kako ih rešiti?
5. Prikazati *boxplot* dijagram i histogram za prosečnu godišnju temperaturu za dva regionala po izboru uz pomoć funkcija `boxplot` i `hist` iz biblioteke `Matplotlib`.
6. Da li raspodela prosečnih dnevних temperatura u Evropi prati normalnu raspodelu? Izračunati koeficijent asimetrije i koeficijent spljoštenosti. Šta se može zaključiti iz njihovih vrednosti?
7. Izračunati prosečne mesečne temperature za svaki region koristeći funkciju `groupby`. Kako se međusobno odnose prosečna temperatura po mesecima u Evropi i Aziji? A Evropi i Africi? Ilustrovati linijskim grafikom i grafikom rasipanja.
8. Šta znači ako je korelacija 0? Koje dodatne informacije pruža grafik rasipanja (funkcija `scatter` iz biblioteke `Matplotlib`) u odnosu na izračunat koeficijent korelacijske?
9. Izračunati korelacionu matricu (funkcija `corr` iz biblioteke `Pandas`) koja pruža uvid u koeficijent korelacijske između svaka dva obeležja. Prikazati je na topotnoj mapi (funkcija `heatmap` iz biblioteke `Seaborn`).

#### Zadatak 2

Za upoznavanje sa podacima koji sadrže pretežno kategorička obeležja koristiće se skup podataka koji sadrži informacije o putnicima koji su plovili „Titanikom“<sup>2</sup>.

1. Učitati bazu i utvrditi broj uzoraka, broj i tip obeležja.
2. Razmisliti koji je najbolji način za rešavanje problema nedostajućih podataka u ovom slučaju. Da li treba dopuniti nedostajuće vrednosti obeležja, ili ima više smisla odbaciti neka obeležja i/ili uzorke?

---

<sup>2</sup>Podaci su dostupni na: [www.kaggle.com/hesh97/titanicdataset-traincsv](http://www.kaggle.com/hesh97/titanicdataset-traincsv)

3. Prikazati uporedno raspodele starosti preživelih i preminulih putnika. Takođe, prikazati raspodelu cene karte za te dve grupe putnika. Da li se može doći do nekih zaključaka na osnovu poređenja ove dve raspodele?
4. Prikazati zavisnost cene karte od starosti putnika i od putničke klase. Da li predstavljanje obe navedene zavisnosti grafikom rasipanja donosi korisne informacije?
5. Prikazati korelacionu matricu koristeći toplotnu mapu. Na šta ukazuje negativna korelacija između putničke klase i cene karte? Da li su u većoj meri korelisana obeležja sa korelacijom -0.7 ili sa korelacijom 0.3?
6. Pri analizi kategoričkih obeležja, korisno je prikazati tzv. pivot tabele, odnosno tabele koje sumarno prikazuju broj uzoraka koji imaju određenu kombinaciju vrednosti dva obeležja. Koristeći funkciju `crosstab` iz Pandas biblioteke prikazati koliko je među putnicima bilo žena, a koliko muškaraca u svakoj od putničkih klasa. Takođe prikazati koliko je putnika preživelo, a koliko nije iz svake od putničkih klasa. Šta se može zaključiti iz dobijenih tabela?
7. Podatke iz pivot tabele iz prethodnog koraka prikazati na grafiku koristeći funkciju `countplot` iz biblioteke Pandas.
8. Za praćenje zavisnosti srednje vrednosti nekog numeričkog obeležja od vrednosti dva kategorička obeležja pogodna je funkcija `barplot` iz biblioteke Seaborn. Prikazati zavisnost prosečne cene karte od pola (M/Ž) putnika, za preživele i stradale putnike.

## 4. Bajesovo pravilo i kvadratni klasifikatori

Prema Bajesovoj teoremi, uslovna verovatnoća događaja  $A$  pod uslovom da se odigrao događaj  $B$  određena je izrazom:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \quad (4.1)$$

Na ovoj matematičkoj vezi zasniva se i konačna formulacija pravila odlučivanja koja minimizuju Bajesov rizik i koja se zajednički nazivaju Bajesovim pravilima odlučivanja. Rizik, kao prosečna cena greške odlučivanja, zavisi od verovatnoće ispravnih i pogrešnih odluka, ali takođe i od njihove cene. Cilj je izbeći katastrofalne odluke, odnosno one sa neprihvatljivo visokom cenom po onoga ko ih donosi. Cilj izbora i transformacije obeležja jeste da se utiče na oblik i karakteristike raspodele vektora obeležja i na taj način smanji rizik odlučivanja, koji je uvek prisutan.

U ovoj vežbi razmatra se najjednostavniji slučaj binarne klasifikacije uzorka. Pravilo odlučivanja koje minimizuje Bajesov rizik često se naziva Bajesovim kriterijumom i za klase  $K_0$  i  $K_1$  dano je izrazom:

$$\Lambda(\mathbf{x}) = \frac{P(\mathbf{x}|K_0)}{P(\mathbf{x}|K_1)} \begin{matrix} > \\[0.2cm] < \end{matrix}_{\substack{K_0 \\ K_1}} \frac{P(K_1)(C_{01} - C_{11})}{P(K_0)(C_{10} - C_{00})}, \quad (4.2)$$

gde su sa  $C_{ij}$  označene cene klasifikacije uzorka klase  $i$  u klasu  $j$ .

Najčešća pretpostavka jeste da sve greške podjednako „koštaju“ ( $C_{10} = C_{01} = 1$ ), a da se ispravne odluke klasifikatora posebno ne nagrađuju ( $C_{00} = C_{11} = 0$ ). Kako se apriorne verovatnoće mogu grupisati sa iste strane nejednakosti na kojoj su i verodostojnosti (klasno uslovljene verovatnoće), kriterijum odlučivanja za ovako definisane cene naziva se kriterijum maksimalne aposteriorne verovatnoće (engl. *maximum a posteriori probability* – MAP), i dat je izrazom:

$$\Lambda(\mathbf{x}) = \frac{P(\mathbf{x}|K_0)P(K_0)}{P(\mathbf{x}|K_1)P(K_1)} = \frac{P(K_0|\mathbf{x})}{P(K_1|\mathbf{x})} \begin{matrix} > \\[0.2cm] < \end{matrix}_{\substack{K_0 \\ K_1}} 1. \quad (4.3)$$

Ako se stvari dodatno pojednostavuje pretpostavkom da su posmatrane klase apriorno podjednako verovatne ( $P(K_0) = P(K_1)$ ), onda se kriterijum odlučivanja uz iste pretpostavke o cenama naziva kriterijum maksimalne vero-

dostojnosti (engl. *maximum likelihood* – ML), i prethodni izraz svodi se na:

$$\Lambda(\mathbf{x}) = \frac{P(\mathbf{x}|K_0)}{P(\mathbf{x}|K_1)} \stackrel{K_0}{>} \stackrel{K_1}{<} 1. \quad (4.4)$$

Pitanje koje se nameće jeste šta raditi u slučaju više od dve klase. MAP pravilo odlučivanja se može tumačiti i kao specijalni slučaj opštijeg pravila koje počiva na principu da se za svaku od klasa definišu posebna preslikavanja  $g_i : \mathbb{R}^d \rightarrow \mathbb{R}$  koja se nazivaju *diskriminantne funkcije*, te da se neobeleženi uzorak  $\mathbf{x} \in \mathbb{R}^d$  pridružuje klasi  $K_i$  čija vrednost diskriminantne funkcije je najveća, odnosno za koju važi da je:

$$g_i(\mathbf{x}) \geq g_j(\mathbf{x}), \forall j, j \neq i, \quad (4.5)$$

što podrazumeva da se u prostoru obeležja opservacija  $\mathbf{x}$  nalazi unutar optimarnog regiona odlučivanja  $R_i$  za klasu  $K_i$  ( $\mathbf{x} \in R_i$ ). Ovakav pristup definisanju pravila odlučivanja otvara mogućnosti i za konstruisanje potpuno novih tipova klasifikatora koji se nazivaju diskriminativnim modelima učenja.

Pojam kvadratnih klasifikatora povezan je sa procesima odlučivanja na osnovu podataka čija se združena raspodela obeležja unutar svake klase opisuje (ili aproksimira) Gausovom raspodelom  $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ , definisanom na sledeći način:

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{D/2} |\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})}, \quad (4.6)$$

gde je  $\mathbf{x}$   $D$ -dimenzionalni vektor obeležja,  $\boldsymbol{\mu}$   $D$ -dimenzionalni vektor srednjih vrednosti obeležja, a  $\boldsymbol{\Sigma}$  kovarijansna matrica data izrazom:

$$\boldsymbol{\Sigma} = \begin{bmatrix} \mathbb{E}[(x_1 - \mu_1)(x_1 - \mu_1)] & \mathbb{E}[(x_1 - \mu_1)(x_2 - \mu_2)] & \cdots & \mathbb{E}[(x_1 - \mu_1)(x_D - \mu_D)] \\ \mathbb{E}[(x_2 - \mu_2)(x_1 - \mu_1)] & \mathbb{E}[(x_2 - \mu_2)(x_2 - \mu_2)] & \cdots & \mathbb{E}[(x_2 - \mu_2)(x_D - \mu_D)] \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{E}[(x_D - \mu_D)(x_1 - \mu_1)] & \mathbb{E}[(x_D - \mu_D)(x_2 - \mu_2)] & \cdots & \mathbb{E}[(x_D - \mu_D)(x_D - \mu_D)] \end{bmatrix} \quad (4.7)$$

u kome simbol  $\mathbb{E}$  označava matematičko očekivanje. Svaka normalna raspodela jedinstveno je određena poznavanjem parametara: kovarijansne matrice  $\boldsymbol{\Sigma}$  i vektora srednjih vrednosti  $\boldsymbol{\mu}$ . Može se uočiti da glavnu ulogu u izrazu 4.6 ima algebarska struktura koja se naziva *kvadratnom formom*:

$$\|\mathbf{x} - \boldsymbol{\mu}\|_{\boldsymbol{\Sigma}^{-1}}^2 = (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}), \quad (4.8)$$

i koju definišu struktura i vrednosti kovarijanske matrice  $\Sigma$  i vektora srednjih vrednosti  $\mu$ .

Kvadratni klasifikatori se mogu opisati kao najjednostavnija vrsta parametarskih modela na bazi normalne raspodele. Ako je poznato da opservacije veličina od interesa koje čine komponente vektora obeležja imaju normalnu raspodelu, tada su kvadratni klasifikatori optimalan izbor pravila odlučivanja jer Gausove raspodele na pravi način opisuju prisutne neodređenosti. Međutim, kada združena raspodela obeležja po klasama ne podleže Gausovoj raspodeli, može se govoriti samo o aproksimaciji stvarnih raspodela koje su najčešće složene i multimodalne. Međutim i takve aproksimacije imaju smisla ukoliko su greške modelovanja prihvatljive, a postoji potreba za oslanjanjem na osobine i analitičku traktabilnost Gausove raspodele.

Kvadratni klasifikatori su u ovoj vežbi uzeti kao pogodan primer za demonstraciju specijalnih kriterijuma odlučivanja koji slede iz opšte teorije minimizacije rizika odlučivanja, odnosno Bajesovog rizika. Za kvadratne klasifikatore uslovne verovatnoće koje se pojavljuju u MAP i ML kriterijumima modeluju se Gausovim raspodelama. Stoga, na osnovu izraza za MAP kriterijum i Gausovu raspodelu i uzimajući u obzir strogu monotonost logaritamske funkcije sledi da je opšti izraz za diskriminantnu funkciju kvadratnog klasifikatora određen sa:

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i) - \frac{1}{2} \ln |\boldsymbol{\Sigma}_i| + \ln P(K_i), \quad (4.9)$$

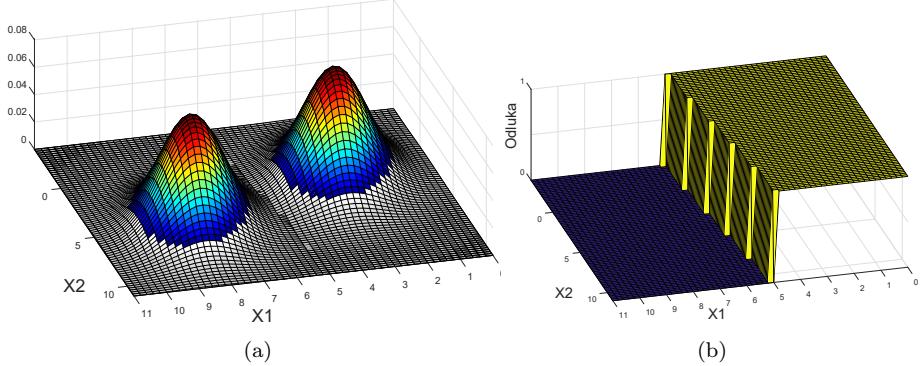
dok su granice odlučivanja između kategorija  $i$  i  $j$  određene kao skupovi tačaka u prostoru obeležja  $\mathbb{R}^d$  koje zadovoljavaju jednakost:  $g_i(\mathbf{x}) = g_j(\mathbf{x})$ .

Kakav oblik će imati granica između regiona odlučivanja  $R_i$  i  $R_j$  uslovljeno je rešenjem sistema jednačina koji proističe iz datog uslova. Ako se pretpostavi da su klasne kovarijanske matrice za sve klase iste, ispostavlja se da granice odlučivanja mogu biti i vrlo jednostavne – linearne, odnosno mogu predstavljati hiperravnji (npr. u 2-D slučaju prave, a u 3-D slučaju ravni).

U opštem slučaju granice odlučivanja kvadratnih klasifikatora su ne-linearne krive drugog reda (u  $\mathbb{R}^2$  to su parabole, hiperbole i elipse), čemu odgovaraju kvadratne diskriminante sa proizvoljnim kovarijanskim matricama. Međutim, ako prilikom modelovanja postoji mogućnost da se opis pojedinačnih klasa pojednostavi, onda bi pod pretpostavkom da jedna ista kovarijansna matrica  $\Sigma$  može da opiše neodređenost svake od klase rezultat bila linearna granica odlučivanja, odnosno klasifikator na bazi minimalnog *Mahalanobisovog rastojanja* definisanog u jednačini (4.8), Slika 4.1. Najčešći cilj ovako „pojednostavljenog“ modelovanja u odnosu na opšte kvadratne klasifikatore jeste da se smanji broj slobodnih parametara koje je potrebno proceniti prilikom obuke modela.

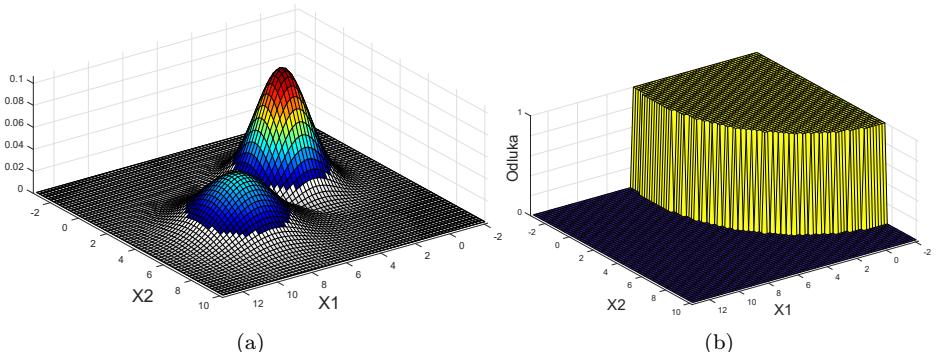
## 4 BAJESOVO PRAVILO I KVADRATNI KLASIFIKATORI

---



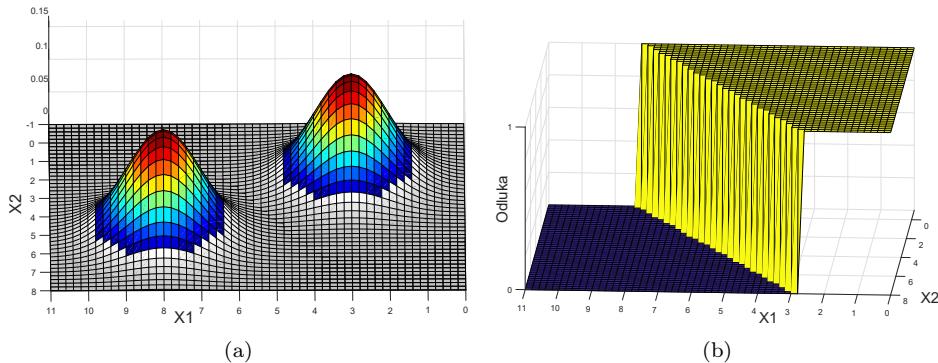
**Slika 4.1:** ML kriterijum odlučivanja za dvodimenzionalne Gausove raspodele sa istim, nedijagonalnim kovarijansnim matricama: (a) gustine raspodele verovatnoće (verodostojnosti); (b) regioni odlučivanja sa linearom granicom između klasa.

Ako bi cilj pak bio da se pored smanjenja broja parametara zadrži i određena diskriminativnost pri opisu različitih klasa, moglo bi da se pretpostavi da su klasne kovarijansne matrice  $\Sigma_1$  i  $\Sigma_0$  međusobno različite, ali da su obe dijagonalne (tj. da su obeležja unutar svake od klasa međusobno nekorelirana, što u slučaju Gausove raspodele znači i statistički nezavisna). Ovom slučaju odgovarale bi nelinearne granice odlučivanja, kao što je prikazano na Slici 4.2(b).



**Slika 4.2:** ML kriterijum odlučivanja između dve klase sa nekorelisanim Gausovim obeležjima: (a) gustine raspodele verovatnoće (verodostojnosti) kojima odgovaraju dijagonalne kovarijansne matrice; (b) regioni odlučivanja sa nelinearnom granicom između klasa.

U slučaju kada su dijagonalne matrice  $\Sigma_1$  i  $\Sigma_0$  iste, opet se dobijaju linearne diskriminantne funkcije  $g_1$  i  $g_0$  i linearne granice odlučivanja u obliku hiperravnji. Međutim, najjednostavniji slučaj jeste ako se prepostavi da su sva obeležja statistički nezavisna i istih varijansi, čime se klasifikatori na bazi minimalnog Mahalanobisovog rastojanja svode na linearne klasifikatore na bazi minimalnog euklidskog rastojanja od centroida posmatranih klasa, što su  $\mu_1$  i  $\mu_0$  za razmatrani problem binarne klasifikacije, kao što je ilustrovano na Slici 4.3.



**Slika 4.3:** Kvadratni klasifikatori na bazi minimalnog euklidskog rastojanja od centroida:  
(a) gustine raspodele verovatnoće (verodostojnosti) kod kojih su obeležja statistički nezavisna i imaju jednake varijanse; (b) regioni odlučivanja sa linearnom granicom između klasa.

## 4.1. Zadaci

### Zadatak 1

Realizovati funkciju `raspodela_1d` koja kao ulazne parametre uzima srednje vrednosti i varijanse, a kao izlaz daje grafike dve jednodimenzionalne normalne raspodele sa datim parametrima. Iz modula `scipy.stats` koristiti funkciju `multivariate_normal.pdf` koja kao ulazne parametre uzima vrednosti nezavisne promenljive i parametre normalne raspodele, a kao izlaz vraća vrednosti zavisne promenljive. Opseg razmatranih vrednosti ulaznog argumenta (nezavisne promenljive) treba da bude takav da obuhvati 99% mogućih opservacija (uzoraka). Prikazati parove funkcija za date vrednosti parametara i identifikovati odgovarajuće regije odlučivanja:

- (a)  $\mu_1 = 0, \sigma_1^2 = 1.5$  i  $\mu_2 = 3, \sigma_2^2 = 4$ ;

- (b)  $\mu_1 = -4, \sigma_1^2 = 2$  i  $\mu_2 = 3, \sigma_2^2 = 1.5$ ;  
 (c)  $\mu_1 = 2, \sigma_1^2 = 2$  i  $\mu_2 = 2, \sigma_2^2 = 1$ .

### Zadatak 2

Realizovati funkciju `raspodela_2d`, sličnu funkciji `raspodela_1d`, koja kao ulazne parametre uzima vektore srednjih vrednosti i varijansi, a kao izlaz daje grafike dve dvodimenzionalne normalne raspodele sa datim parametrima. Prikazati parove funkcija za sledeće vrednosti parametara i analizirati odgovarajuće regije odlučivanja:

- (a)  $\mu_1 = [3 \ 2], \Sigma_1 = \begin{bmatrix} 1.5 & 0 \\ 0 & 1.5 \end{bmatrix}$  i  $\mu_2 = [8 \ 5], \Sigma_2 = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}$ ;
- (b)  $\mu_1 = [3 \ 2], \Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}$  i  $\mu_2 = [8 \ 5], \Sigma_2 = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$ ;
- (c)  $\mu_1 = [3 \ 2], \Sigma_1 = \begin{bmatrix} 1 & 0.6 \\ 0.6 & 3 \end{bmatrix}$  i  $\mu_2 = [8 \ 5], \Sigma_2 = \begin{bmatrix} 2 & -0.6 \\ -0.6 & 1 \end{bmatrix}$ ;
- (d)  $\mu_1 = [3 \ 2], \Sigma_1 = \begin{bmatrix} 1 & 0.8 \cdot 1 \cdot \sqrt{3} \\ 0.8 \cdot 1 \cdot \sqrt{3} & 3 \end{bmatrix}$  i  $\mu_2 = [8 \ 5], \Sigma_2 = \Sigma_1$ .

### Zadatak 3

Analizirati na grafiku rasipanja uzorka kako zavisnost između obeležja utiče na rasipanje uzorka opisanih dvodimenzionalnom Gausovom raspodelom. Generisanje uzorka iz normalne raspodele može se izvršiti korišćenjem funkcije iz NumPy biblioteke pod nazivom `multivariate_normal` prosleđivanjem parametara raspodele na sledeći način:

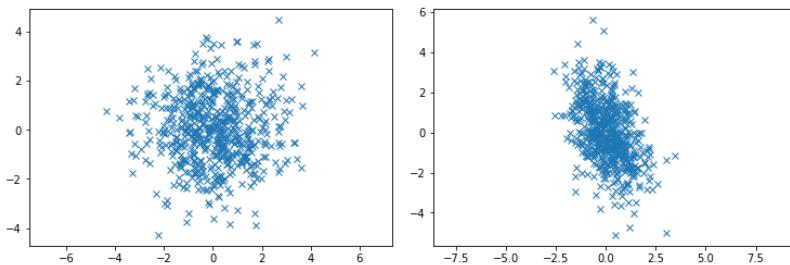
```
x, y = np.random.multivariate_normal(mu, sigma, N).T
plt.plot(x, y, 'x')
plt.axis('equal')
```

Za parametre raspodele koristiti srednju vrednost  $\mu_2 = [0 \ 0]$  i kovariansnu matricu:

- (a)  $\Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$ ,      (b)  $\Sigma = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$ ,      (c)  $\Sigma = \begin{bmatrix} 0.2 & 0 \\ 0 & 2 \end{bmatrix}$ ,

$$(d) \Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 0.2 \end{bmatrix}, \quad (e) \Sigma = \begin{bmatrix} 1 & -0.7 \\ -0.7 & 2.5 \end{bmatrix}, \quad (f) \Sigma = \begin{bmatrix} 2.5 & 0.7 \\ 0.7 & 1 \end{bmatrix}.$$

U kakvoj su vezi vrednosti u kovarijansnoj matrici van glavne dijagonale sa međusobnom zavisnošću odgovarajućih obeležja? Kako se to odslikava na rasipanje uzoraka? Na Slici 4.4 dati su primeri rasipanja uzoraka kada su obeležja međusobno nezavisna (levo) i kada su međusobno zavisna (desno).



**Slika 4.4:** Grafici rasipanja za uzorke iz raspodele opisane dijagonalnom matricom (levo) i nedijagonalnom matricom (desno).

#### Zadatak 4

Data je funkcija `ML_odluka`. Ova funkcija kao ulazne parametre uzima generisane uzorke i parametre dve raspodele iz kojih su uzorci generisani. Računanjem vrednosti gustine raspodele verovatnoće u određenoj tački, ona donosi odluku o klasnoj pripadnosti uzorka na osnovu ML kriterijuma. Potom poredi predviđene klase sa odgovarajućim oznakama stvarne klase i računa grešku po klasi, kao i prosečnu grešku modela.

1. Generisati po 300 uzoraka iz dve klase sa parametrima  $\mu_1 = 0, \sigma_1^2 = 1.5$  i  $\mu_2 = 3, \sigma_2^2 = 1.5$ . Uzorcima pridružiti odgovarajuće oznake (labele) prema pripadnosti klasi (1 za pripadnost prvoj klasi i 2 za pripadnost drugoj). Potom klasifikovati označene uzorke na osnovu ML kriterijuma odlučivanja i prikazati greške modela.
2. Po ugledu na kod iz date funkcije `ML_odluka` implementirati funkciju `MAP_odluka` i potom, korišćenjem MAP kriterijuma, klasifikovati slučajno generisane uzorke prethodno pomenutih raspodela, ako:
  - (a) skup uzoraka sadrži 200 uzoraka iz klase 1 i 400 uzoraka iz klase 2;

## 4 BAJESOVO PRAVILO I KVADRATNI KLASIFIKATORI

---

- (b) skup uzoraka sadrži 120 uzoraka iz klase 1 i 480 izoraka iz klase 2;  
(c) skup uzoraka sadrži 400 uzoraka iz klase 1 i 200 izoraka iz klase 2.

```
def ML_odluka(D, mi1, mi2, sigma1, sigma2):  
    oznake = np.unique(D[:, -1])  
    F1 = scipy.stats.multivariate_normal.pdf(D[:, :-1], mi1,  
                                             sigma1)  
    F2 = scipy.stats.multivariate_normal.pdf(D[:, :-1], mi2,  
                                             sigma2)  
  
    odluke = F1>F2  
    rez_oznake = odluke*oznake[0] + (1-odluke)*oznake[1]  
    greske = (rez_oznake != D[:, -1])  
  
    ver_greske = np.zeros(len(oznake)+1)  
    ver_greske[0] = np.sum(np.logical_and(D[:, -1]==1,  
                                         rez_oznake==2))/np.sum(D[:, -1]==1)  
    ver_greske[1] = np.sum(np.logical_and(D[:, -1]==2,  
                                         rez_oznake==1))/np.sum(D[:, -1]==2)  
    ver_greske[-1] = np.sum(greske)/np.shape(D)[0]  
  
    return rez_oznake, ver_greske
```

## 5. Estimacija GRV: KDE i $k$ NN

Metode za procenu *gustine raspodele verovatnoće* (GRV) mogu da se podele na parametarske i neparametarske, u zavisnosti od toga da li se unapred nameće određena struktura rešenja (određeni oblik nepoznate raspodele). Tako je za parametarske metode karakteristično da se unapred pretpostavlja da bi tražena raspodela mogla da pripada određenoj familiji krivih ili površi, odnosno da se problem procene GRV može svesti na problem tačkastih ocena pojedinačnih parametara koji figurišu u tako pretpostavljenom modelu. Primera radi, uz pretpostavku da uzorci iz određene klase podležu Gausovoj raspodeli, problem se svodi na procenu vrednosti parametara te raspodele.

Alternativa uvođenju pretpostavke o obliku raspodele jeste neparametarsko određivanje GRV direktno na osnovu uzorka, a što u zavisnosti od složenosti oblika raspodele najčešće podrazumeva da je na raspolaganju veći broj uzorka, a u slučaju veće dimenzionalnosti prostora obeležja i veću računsku složenost. Međutim, neparametarski pristupi ne podrazumevaju potpuno odsustvo bilo kakvih parametara, već da su prisutni parametri (kojih je najčešće relativno malo) takvi da omogućavaju upravljanje procesom estimacije i na taj način kao hiperparametri oblikuju karakter rešenja. Na primer, u slučaju odlučivanja metodom  $k$  najbližih suseda, o kojoj će biti više reči u Poglavlju 9,  $k$  jeste vrednost od koje će zavisiti dobijeno rešenje, ali nije parametar raspodele kao što je to npr. kovarijansna matrica kod Gausove raspodele.

Kao najveći nedostatak neparametarskih metoda često se u prvi plan ističe problem „prokletstva dimenzionalnosti“, a koji podrazumeva da sa povećanjem dimenzionalnosti prostora obeležja gustina prikupljenih uzorka brzo opada ka nuli. Nedovoljan broj uzorka po jedinici zapremine mogao bi se nadomestiti prikupljanjem dodatnih uzorka, ali se pokazuje da takva strategija nije održiva, s obzirom na to da je rast zapremine prostora sa povećanjem dimenzionalnosti eksponencijalan. Ako za tim postoji potreba, rešenje jesu robustne neparametarske metode, particija prostora obeležja ili redukcija dimenzionalnosti.

U nastavku će ukratko biti predstavljene dve osnovne metode za neparametarsku procenu gustine raspodele verovatnoće – procena GRV na osnovu kernela (engl. *kernel density estimation* – KDE) i procena GRV na osnovu  $k$  najbližih suseda (engl.  *$k$ -nearest neighbours density estimation* –  $k$ NN). Cilj ove vežbe jeste demonstracija osnovnih principa na kojima se zasnivaju KDE i  $k$ NN i mogućnosti njihove primene u različitim modelima učenja.

### 5.1. Procena GRV na osnovu kernela

Neparametarske metode za procenu GRV na osnovu kernela predstavljaju uopštenja metode *histograma*. U tom smislu, KDE spada u familiju metoda koje kvalitet aproksimacije GRV podižu primenom različitih težinskih funkcija ili *kernela* koji se koriste za ponderisanje uzoraka, odnosno kao „jezgro“ za interpolaciju oblika nepoznate raspodele. U zavisnosti od izbora težinske funkcije postoje različite varijante KDE. Primera radi, KDE metoda najsličnija proceni GRV korišćenjem metode histograma jeste KDE na bazi *Parzenovog kernela* ili *pravougaone prozorske funkcije*.

Uz pretpostavku da je raspodela nepoznate slučajne promenljive diskretna i da je na raspolaganju  $N$  uzoraka, metoda histograma predstavlja aproksimaciju GRV korišćenjem relativnih frekvencija  $P_i$ :

$$P_i \approx \frac{k_i}{N}, \quad (5.1)$$

gde je  $k_i$  frekvencija pojavljivanja događaja  $i$ .

Međutim, veličine koje se razmatraju često nisu diskretnе i procena GRV na bazi metode histograma zahteva kvantizaciju domena nad kojim se izračunava. U skladu sa tim, procena jednodimenzionalne GRV  $\hat{p}_H(x)$  metodom histograma određena je izrazom:

$$\hat{p}_H(x) \approx \frac{1}{h} \frac{k_i}{N}, \quad |x - \hat{x}_i| \leq \frac{h}{2}, \quad (5.2)$$

gde je  $h$  širina kvantizacionog intervala koji reprezentuje diskretna vrednost  $\hat{x}_i$ ,  $x \in \mathbb{R}$  uzorak ili tačka za koju se izračunava aproksimacija, a  $k_i$  broj pojavljivanja uzoraka u datom,  $i$ -tom intervalu, utvrđen na osnovu prethodnih  $N$  opservacija. U (5.2) je prepostavljen da je širina kvantizacionih intervala uniformna, odnosno da je  $h$  konstantno, što je česta pretpostavka. Na osnovu zakona velikih brojeva teorije verovatnoće može se tvrditi da u asimptotskom slučaju,  $N \rightarrow \infty$ , relativne učestanosti događaja konvergiraju stvarnim vrednostima verovatnoće, a da diskretna aproksimacija  $\hat{p}_H(x)$  prikazana u (5.2) konvergira stvarnoj neprekidnoj raspodeli  $p(x)$ .

Po analogiji sa zapreminskom masom (gustinom) homogenih tela, aproksimacija verovatnoće događaja da će se uzorak naći u regionu  $R$  zapremine  $V$  u kome je GRV približno konstantna može se iskazati sa:

$$P = \int_R p(\mathbf{x}) d\mathbf{x} \approx p(\mathbf{x}) \cdot V. \quad (5.3)$$

Odatle sledi da se isti princip koji je usvojen u aproksimaciji jednodimenzionalne GRV na bazi metode histograma u (5.2) može primeniti i na razmatrani višedimenzionalni slučaj  $\mathbf{x} \in \mathbb{R}^D$  u (5.3). Drugim rečima, u regionu nad kojim je  $p(\mathbf{x})$  približno konstantna važi sledeća aproksimacija:

$$\hat{p}(\mathbf{x}) = \frac{P}{V} = \frac{k}{NV}, \quad (5.4)$$

gde je  $k$  broj uzoraka obuhvaćen zapreminom  $V$ , dok je  $N$  ukupan broj uzoraka u prostoru obeležja. Preciznost ili rezolucija aproksimacije se povećava sa smanjenjem zapremine  $V$ , ali istovremeno izaziva i smanjenje vrednosti  $k$ , s obzirom da je ukupan broj uzoraka  $N$ , raspodeljenih po čitavom prostoru obeležja, konačan.

S jedne strane, zapremina  $V$  treba da bude dovoljno mala tako da važi aproksimacija u (5.3). S druge strane, kako bi KDE procena GRV bila dovoljno tačna, zapremina  $V$  treba da bude i dovoljno velika da obuhvati broj uzoraka  $k$  koji je dovoljan da obezbedi konvergenciju relativne učestanosti pojavljivanja ka stvarnoj vrednosti verovatnoće. Stoga je u praksi potrebno naći kompromisno rešenje, koje će istovremeno obezbediti i dovoljno preciznu, ali i dovoljno tačnu procenu originalne raspodele.

Za pronalaženje kompromisnog rešenja u (5.4) postoje dva pristupa. Prvi pristup proceni GRV u okolini tačke  $\mathbf{x}$  sastoji se u tome da se unapred odredi ili fiksira vrednost zapremine  $V$  oko tačke  $\mathbf{x}$  i da se zatim odredi broj uzoraka  $k$  koji se nalaze u njoj, dok drugi pristup podrazumeva da se unapred fiksira potrebna vrednost  $k$  i da se zatim odredi potrebna zapremina prostora  $V$  koja obuhvata traženi broj uzoraka.

U prvom pristupu, ako prilikom određivanja vrednosti  $k$  funkcija za prebrojavanje uzoraka u podjednakoj meri uzima u obzir sve uzorke koji se nalaze u fiksnoj „kockastoј“ zapremini  $V$  koja definiše okolinu neke tačke  $\mathbf{x} \in \mathbb{R}^D$ , onda se radi o KDE metodi na bazi Parzenovog ili pravougaonog kernela, a u suprotnom o KDE metodama koje koriste glatke kernele. Razlog za popularnost Parzenovog kernela jeste i činjenica da su u prošlosti računarski resursi bili manje dostupni, pa su prednost imale kernel metode koje u poređenju sa sofisticiranjim težinskim funkcijama ne iziskuju dodatne računske operacije (u ključnom koraku estimacije Parzenov kernel pridružuje podjednaku težinu svim uzorcima koji se nalaze u domenu nad kojim je definisana kernel funkcija i nema potrebe za dodatnim množenjima sa težinskim koeficijentima). U drugom pristupu, zapremina oko posmatrane tačke  $\mathbf{x} \in \mathbb{R}^D$  se podrazumevano povećava podjednako duž svih  $D$  pravaca prostora, sve dok ne obuhvati traženi broj uzoraka  $k$ , i u tom slučaju se radi o proceni GRV na osnovu  $k$  najbližih suseda, o

čemu će biti više reči u Odeljku 5.2.

U prostoru  $\mathbb{R}^D$  pravougaoni prozor širine  $h$  odgovara hiperkocki zapreminе  $V = h^D$ . Ako se indikatorska funkcija pripadnosti uzorka  $\mathbf{u} \in \mathbb{R}^D$  hiperkocki stranice  $h = 1$ , postavljenoj u koordinatni početak  $\mathbb{R}^D$ , definiše izrazom:

$$K(\mathbf{u}) = \begin{cases} 1, & |u_j| < 1/2, \forall j \in \{1, \dots, D\} \\ 0, & \text{drugde} \end{cases}, \quad (5.5)$$

gde su  $u_j$  odgovarajuće komponente vektora  $\mathbf{u}$ , onda za neki uzorak  $\mathbf{x}^{(n)}$  koji se nađe unutar hiperkocke sa stranicom  $h$  koja je postavljena u koordinatni početak važi:

$$K\left(\frac{\mathbf{0} - \mathbf{x}^{(n)}}{h}\right) = 1. \quad (5.6)$$

U opštem slučaju, ako se opisana hiperkocka stranice  $h$  premesti na proizvoljnu poziciju  $\mathbf{x}$  u  $\mathbb{R}^D$ , vrednost težinske funkcije  $K_P(\mathbf{x}, \cdot)$  u tački  $\mathbf{x}^{(n)}$ :

$$K_P(\mathbf{x}, \mathbf{x}^{(n)}) = K\left(\frac{\mathbf{x} - \mathbf{x}^{(n)}}{h}\right), \quad (5.7)$$

ukazuje na to da li tačka  $\mathbf{x}^{(n)}$  pripada opisanoj hiperkocki, odnosno odgovara težinskoj funkciji koja se naziva *Parzenov kernel*. Vrednost  $K_P(\mathbf{x}, \mathbf{x}^{(n)})$  ne zavisi od konkretnе vrednosti uzorka  $\mathbf{x}^{(n)}$ , već samo od toga da li se on nalazi unutar hiperkocke sa stranicom  $h$ . Pri tome, vrednost pomenute indikatorske funkcije može biti samo 0 ili 1. To znači da je ukupan broj uzoraka  $k$  u tako definisanoj zapremini  $V$  (hiperkocki stranice  $h$ ) zapravo određen sumom vrednosti Parzenovog kernela izračunatog za svaki od  $N$  uzoraka:

$$k = \sum_{i=1}^N K\left(\frac{\mathbf{x} - \mathbf{x}^{(i)}}{h}\right). \quad (5.8)$$

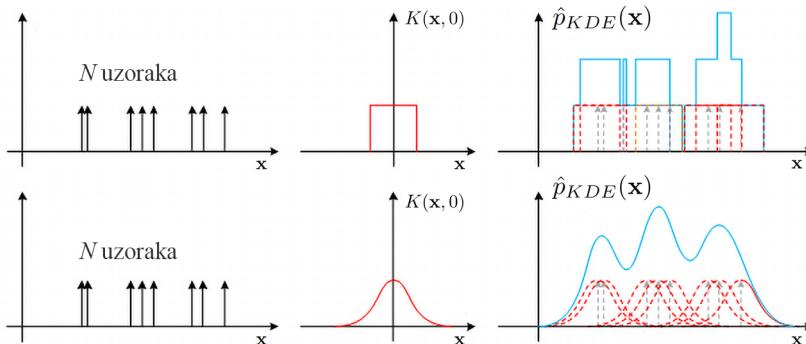
Konačno, na osnovu izraza (5.4) i (5.8) sledi da je u slučaju proizvoljnog kernela KDE aproksimacija GRV u tački  $\mathbf{x} \in \mathbb{R}^D$  određena izrazom:

$$\hat{p}_{KDE}(\mathbf{x}) = \frac{1}{N \cdot h^D} \sum_{i=1}^N K\left(\frac{\mathbf{x} - \mathbf{x}^{(i)}}{h}\right). \quad (5.9)$$

Iz tog razloga se KDE na osnovu Parzenovog kernela (prozora) širine  $h$  smantra uopštenjem histograma, jer je oblik Parzenovog kernela nad domenom koji odgovara fiksnoj zapremini  $V$  konstantan i nema dodatni uticaj na vrednost

KDE procene u (5.9), već se  $K_P$  ponaša isključivo kao indikatorska funkcija koja je korisna za „prebrojavanje“.

Izraz (5.9) može se interpretirati i kao broj hiperkocaka koje se preklapaju u tački  $\mathbf{x}$ , a koje su centrirane oko pojedinačnih uzoraka  $\mathbf{x}^{(i)}$ ,  $i = 1, \dots, N$ . Ovakva interpretacija je bliska i poznatoj činjenici da je konvolucija impulsnog odziva filtra sa povorkom delta impulsa jednaka superpoziciji impulsnih odziva transliranih na pozicije impulsa, pa se izraz (5.9) može tumačiti i tako da niz diskretnih vrednosti  $\mathbf{x}^{(n)}$ ,  $n = 1, \dots, N$ , predstavlja pozicije impulsa u  $\mathbb{R}^D$ , a kernel funkcija odgovara impulsnom odzivu filtra, Slika 5.1.



**Slika 5.1:** KDE procena GRV korišćenjem pravougaone (gornji red) i Gausove (donji red) težinske funkcije (kerneli prikazani u središnjem delu slike). Procene prikazane u desnom delu slike mogu se interpretirati i kao rezultat konvolucije odgovarajućih kernela sa povorkom impulsa koja odgovara uzorcima (prikazanom u levom delu slike).

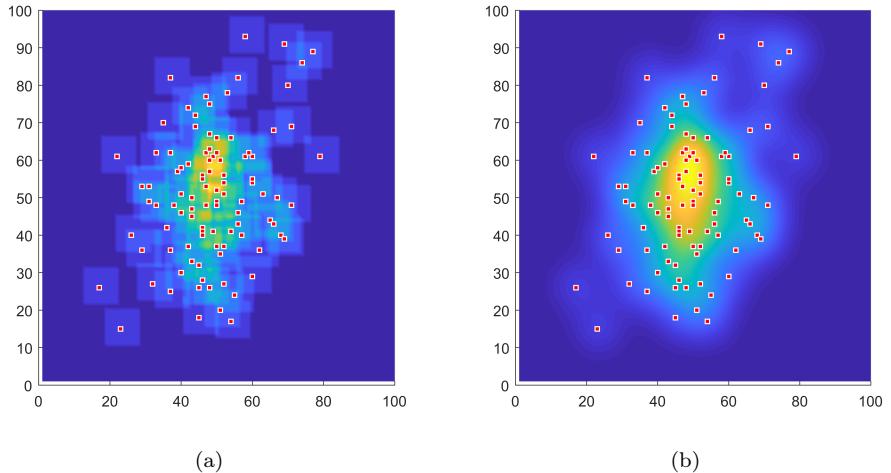
Gausov kernel sa jediničnom varijansom, u tački  $\mathbf{x}^{(n)}$  definisan je izrazom:

$$K_G(\mathbf{x}, \mathbf{x}^{(n)}) = K(\mathbf{x} - \mathbf{x}^{(n)}) = \frac{1}{(2\pi)^{D/2}} e^{-\frac{1}{2}(\mathbf{x} - \mathbf{x}^{(n)})^T (\mathbf{x} - \mathbf{x}^{(n)})}. \quad (5.10)$$

Prepostavka koja se često usvaja jeste da su kerneli izotropni (da se ponašaju isto duž svih pravaca prostora) i da kao kod (5.10) i (5.7) njihove vrednosti zavise samo od rastojanja između argumenata. U skladu sa tim, takvi kerneli se ponekad definišu i kao funkcije samo jednog argumenta:  $|\mathbf{x} - \mathbf{x}^{(n)}|$ .

Parametar  $h$  je srazmeran zapremini  $V$  i predstavlja upravljački ili hiperparametar KDE procene. U tom smislu, ranije diskutovani kompromis između  $k$  i  $V$  se u slučaju KDE metoda svodi na izbor optimalne širine  $h$  kernel funkcije  $K$ .

Uticaj izbora oblika kernel funkcije na kvalitet KDE aproksimacije ilustrovan je na Slikama 5.2 i 5.3.



**Slika 5.2:** Neparametarska procena 2D Gausove GRV na osnovu 99 slučajnih uzoraka iz populacije sa srednjom vrednošću  $\mu_1 = [50 \ 50]$  i kov. matricom  $\Sigma = [150 \ 60; \ 60 \ 300]$ , korišćenjem KDE metode sa: (a) pravougaonom prozorskom funkcijom (Parzenovim kernelom) širine  $h = 10$ ; i (b) glatkim kernelom u obliku Gausove prozorske funkcije sa standardnom devijacijom  $\sigma = h/2$ .

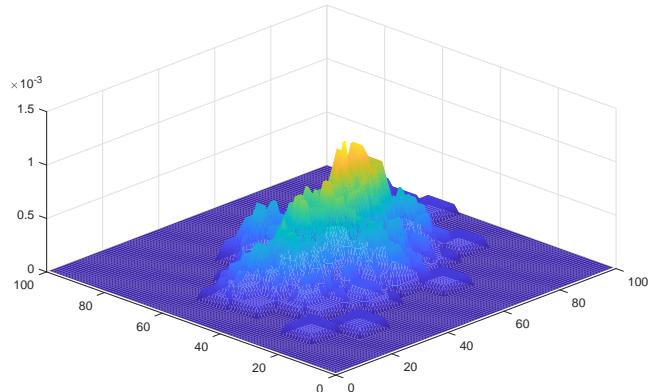
Pitanje koje se dodatno postavlja jeste koliko je KDE procena (5.9) tačna, ili u terminologiji estimatora, *nepristrasna*. Iz ranije diskusije jasno je da kada  $N \rightarrow \infty$ , procena teži stvarnoj raspodeli, ali se postavlja pitanje kako na to utiče izbor oblika težinske funkcije. Odgovor je dat izrazom:

$$\begin{aligned} \mathbb{E}_{p(\mathbf{x})} \left[ \hat{p}_{KDE}(\mathbf{x}) \right] &= \frac{1}{h^D} \int_{\mathbf{y} \in \mathbb{R}^D} K \left( \frac{\mathbf{x} - \mathbf{y}}{h} \right) p(\mathbf{y}) d\mathbf{y} \\ &= \frac{1}{h^D} K \left( \frac{\mathbf{x}}{h} \right) * p(\mathbf{x}). \end{aligned} \quad (5.11)$$

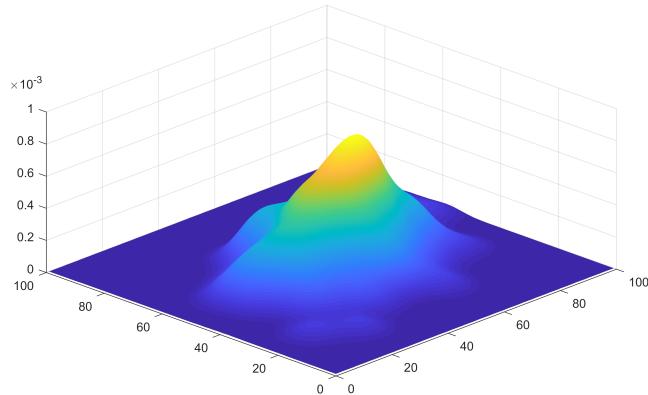
Prema ovom izrazu, očekivana vrednost KDE procene sa glatkim kernelom  $K$  širine  $h$  odgovara vrednosti konvolucije originalne GRV  $p(\mathbf{x})$  sa tim kernelom. To znači da suviše uzak kernel generiše procenu sa lažnim modovima koji ne postoje u originalnoj raspodeli. Sa druge strane, pri velikom  $h$  procena  $\hat{p}_{KDE}(\mathbf{x})$

## 5 ESTIMACIJA GRV: KDE I KNN

---



(a)



(b)

**Slika 5.3:** Ilustracija razlika između neparametarskih KDE procena zasnovanih na pravougaonim i glatkim kernelima, na primeru 99 uzoraka Gausove raspodele sa Slike 5.2: (a) Parzenov prozor širine  $h = 10$ ; (b) glatki Gausov kernel standardne devijacije  $\sigma = h/2$ .

nema onaj nivo detalja koji ima  $p(\mathbf{x})$ , jer je kernel suviše širok i izraženo uprosečavanje sakriva stvarni oblik raspodele. U idealnom slučaju, kada bi važilo

$N \rightarrow \infty$ , mogao bi se usvojiti kernel širine  $h \approx 0$  (što odgovara izuzetno maloj zapremini  $V$ ), očekivana vrednost u (5.11) bi se svela na konvoluciju  $p(\mathbf{x})$  sa delta impulsom, što bi i bila idealna procena. Međutim, za svaku konačnu vrednost  $N$  usvajanje kernela širine bliske nuli nema smisla jer bi dobijena procena odgovarala nizu veoma uskih impulsa u tačkama u kojima se nalaze uzorci.

Optimalna širina kernela se u praksi može birati na više načina, a kod procene jednodimenzionalne GRV vrlo često se za to koristi izraz:

$$h_{opt} = 0,92AN^{-0,2}, \quad A = \min \{\sigma, \text{IQR}/1,32\}, \quad (5.12)$$

gde je  $\sigma$  uzoračka standardna devijacija, a IQR je interkvartilni opseg.

## 5.2. Procena GRV metodom $k$ najbližih suseda

Kao što je već diskutovano u Odeljku 5.1, aproksimacija definisana u (5.4) omogućava i pristup proceni GRV u kome se unapred fiksira potreban broj uzoraka  $k$  u okolini posmatrane tačke  $\mathbf{x} \in \mathbb{R}^D$ , a zapremina  $V$  se posebno određuje za svaku tačku  $\mathbf{x}$  tako što se povećava podjednako duž svih pravaca sve dok ne obuhvati traženi broj uzoraka  $k$  u njenoj okolini. Opisani postupak za izračunavanje  $\hat{p}_{kNN}(\mathbf{x})$  svodi se na traženje  $k$  najbližih suseda tačke  $\mathbf{x}$ , odakle i naziv  $k$ NN (eng. *k-nearest neighbours*).

Trebalo bi istaći da se da se u ovom odeljku  $k$ NN posmatra isključivo kao metoda za *procenu* GRV, iako se tako dobijene klasno uslovljene procene GRV u narednom koraku mogu koristiti i za klasifikaciju, koristeći recimo MAP kriterijum. S druge strane, na osnovu identifikovanih  $k$  najbližih suseda klasifikacija se može izvršiti i direktno, o čemu će biti više reči u Poglavlju 9.

Ako se prilikom  $k$ NN procene GRV izabere prevelika vrednost  $k$ , postiže se efekat usrednjavanja originalne raspodele  $p(\mathbf{x})$  i stapanja pojedinačnih modova (dolazi do gubitka rezolucije), ali je vrednost zapremine  $V$  pri tome uglavnom ujednačena u različitim delovima prostora (pošto je potrebno pronaći velik broj uzoraka i uvek je potrebno pretražiti okolinu sa velikom zapreminom). Sa druge strane, za malu vrednost  $k$  dobija se na rezoluciji procene, ali samo pod uslovom da postoji dovoljan broj uzoraka u svim delovima prostora. U suprotnom, vrednosti  $r_k(\mathbf{x})$  se skokovito menjaju od tačke do tačke, što dovodi do pojave lažnih modova i loše procene GRV. Dakle, kao i kod KDE, izbor vrednosti upravljačkog parametra određen je odgovarajućim kompromisom.

### 5.3. Zadaci

#### Zadatak 1

Date su sledeće funkcije:

1. `raspodela_1D_vise_modova(m, Sigma, P, korak)`

Funkcija za izračunavanje i prikaz multimodalne 1D raspodele sa modovima u obliku 1D Gausove raspodele i ulaznim argumentima:  $m$  – vektor vrednosti koje definišu pozicije modova, odnosno srednje vrednosti pojedinačnih 1D Gausovih komponenti;  $Sigma$  – vektor varijansi 1D Gausovih raspodela koje definišu modove;  $P$  – vektor sa udelima pojedinačnih Gausovih komponenti (modova) u multimodalnoj raspodeli; i  $korak$  – rezolucija sa kojom se diskretizuje interval vrednosti obeležja nad kojim se izračunavaju i skiciraju vrednosti raspodele. Opseg vrednosti za koje se tražena raspodela izračunava i prikazuje određen je intervalom  $3\sigma$  u odnosu na najmanji i najveći mod multimodalne raspodele (najmanju i najveću vrednost vektora  $m$ ), imajući u vidu da se 99% uzoraka Gausove raspodele nalazi u intervalu  $[\mu - 3\sigma, \mu + 3\sigma]$ .

```
def raspodela_1D_vise_modova(m, sigma, P, korak_raspodele):
    M = len(m)
    x_min = np.min(m); x_max = np.max(m)
    sigma_max = np.max(sigma)
    x = np.arange(x_min - 3 * np.sqrt(sigma_max),
                  x_max + 3 * np.sqrt(sigma_max), korak_raspodele)
    F = np.zeros(len(x))
    for i in range(M):
        F += P[i]*scipy.stats.multivariate_normal.pdf(
            x, m[i], sigma[i]).T
    plt.plot(x, F)
```

2. `uzorci_1D_vise_modova(m, Sigma, P, N)`

Funkcija koja generiše  $N$  uzoraka multimodalne raspodele sa modovima u obliku 1D Gausove raspodele. Ulagani argumenti funkcije predstavljaju:  $m$  – vektor vrednosti koje definišu pozicije modova, odnosno srednje vrednosti pojedinačnih 1D Gausovih komponenti;  $Sigma$  – vektor varijansi 1D Gausovih raspodela koje definišu modove;  $P$  – vektor sa udelima pojedinačnih Gausovih komponenti (modova) u multimodalnoj raspodeli; i  $N$  – željeni broj slučajnih uzoraka.

```

def uzorci_1D_vise_modova(m, sigma, P, N):
    M = len(m)
    X = np.zeros(N)
    index = 0
    for i in range(M):
        N_i = int(np.floor(P[i]*N))
        if i == M-1 and index+N_i < N:
            N_i += 1
        X[index:index+N_i] = np.random.normal(
            m[i], np.sqrt(sigma[i]), N_i)
        index += N_i
    return X

```

Za raspodelu sa sledećim karakteristikama modova  $\mu = [0 \ 2.5]$ ,  $\sigma^2 = [0.2 \ 0.2]$ ,  $P = [1/3 \ 2/3]$ , uporediti izgled originalne raspodele sa raspodelom estimiranim pomoću Parzenovog prozora. Varirati širinu kernela  $h$  i broj slučajnih uzoraka  $N$  na osnovu kojih se određuje procena GRV u KDE metodi. Pored toga, uporediti originalne raspodele sa njihovim procenama na osnovu  $k$  najbližih suseda. Varirati broj uzoraka  $N$  i broj najbližih suseda  $k$  na osnovu kojih se estimira GRV u kNN metodi. Zadatak realizovati implementacijom dve funkcije:

- `[p, x_min, x_max] = estimacija_1D_KDE(X, h, korak)` i
- `[p, x_min, x_max, psp] = estimacija_1D_KNN(X, k, korak)`.

Ulagni argumenti su: `X` – vektor slučajnih uzoraka na osnovu kojih se izračunava procena GRV; `korak` – rezolucija sa kojom se diskretizuje domen u kom se nalaze uzorci, čime se implicitno definiše i broj tačaka u kojima se traži procena GRV. Izlazni argumenti su: `p` – vektor sa procenjenim vrednostima GRV; `x_min` i `x_max` – najmanja i najveća vrednost obeležja slučajnih uzoraka prosledenih funkciji kroz ulazni argument `X`; `psp` – vrednost prosečne širine prozora (zapremine prostora obeležja u okolini svake tačke u kojoj se estimira GRV) koja je bila potrebna da bi se prilikom  $k$ NN procene GRV u okolini svih tačaka od interesa u intervalu  $[x_{\min}, x_{\max}]$  obezbedio traženi broj uzoraka  $k$ .

Koraci za implementaciju funkcije `estimacija_1D_KDE`

1. definisati parametre
2. kroz `for` petlju postavljati prozore sa centrom u "i", gde se "i" menja sa zadatim korakom od minimalne do maksimalne vrednosti slučajnih uzoraka (obeležja)
3. prebrojati koliko slučajnih uzoraka upada u prozor

## 5 ESTIMACIJA GRV: KDE I KNN

---

- ```
širine h postavljen u "i"
4. p_procena[k] = broj_uzoraka / (N*h^D) #1D prozor
    k = k +1, natrag na 2. do kraja for petlje po "i".
```

Koraci za implementaciju funkcije `estimacija_1D_KNN`

1. definisati parametre
2. kroz `for` petlju odrediti kNN procenu GRV u svim tačkama "i" koje se nalaze na međusobnom rastojanju korak\_raspodele između minimalne i maksimalne vrednosti obeležja slučajnih uzoraka X, uključujući i granične vrednosti `x_min` i `x_max`
3. izračunati euklidsko rastojanje od trenutne pozicije u kojoj se traži procena GRV do svih uzoraka u X
4. na osnovu rastojanja izabrati k najbližih suseda i izračunati zapreminu prozora koji obuhvata tih k najbližih suseda (slučajnih uzoraka iz X)  
1D zapremina = širina intervala
5.  $p\_procena[j] = k/(N * \text{zapremina\_prozora})$   
 $\text{psp\_sum} = \text{psp\_sum} + \text{sirina\_prozora}$   
 $j = j + 1$ , natrag na 2. do kraja `for` petlje po "i"
6. odrediti prosečnu širinu korišćenih prozora (psp).

U okviru `Scikit-learn` biblioteke, u modulu `neighbors`, postoji klasa `KernelDensity`, koja sadrži implementaciju estimacije raspodele KDE metodom. Može se koristiti za podatke iz prostora bilo koje dimenzionalnosti, za razliku od funkcije koja je implementirana u prethodnom zadatku isključivo za jednodimenzionalne uzorke. Klasa sadrži istoimenu metodu za inicijalizaciju, u kojoj se putem ulaznih parametara `kernel` i `bandwidth` mogu specificirati željeni oblik kernela (`gaussian`, `tophat`, itd.), kao i širina kernela. Raspodela se estimira korišćenjem metode `fit` i dostupnih uzoraka, a konkretne vrednosti estimirane raspodele u zadatim tačkama računaju se pomoću metode `score_samples`.

Primer estimacije GRV na osnovu jednodimenzionalnih uzoraka iz X pomoću KDE sa Gausovim kernelom i Parzenovim prozorom (`tophat` kernel) u tačkama iz `X_plot`:

```
for kernel in ['gaussian', 'tophat']:
    kde = KernelDensity(kernel=kernel, bandwidth=0.5)
    kde.fit(X)
    log_dens = kde.score_samples(X_plot)
    plt.plot(X_plot[:, 0], np.exp(log_dens))
```

## 6. Naivni Bajesov klasifikator

Kao što je već objašnjeno u Poglavlju 4, Bajesov klasifikator definiše pravilo odlučivanja koje minimizuje Bajesov rizik, odnosno uzima u obzir cene odluka i apriorne verovatnoće pojavljivanja pojedinačnih klasa. U ovoj vežbi će biti razmatran specijalan slučaj Bajesovog klasifikatora koji se naziva *naivnim Bajesovim klasifikatorom*, a koji, u cilju povećanja robustnosti procene GRV, polazi od pretpostavke da su obeležja unutar svake klase međusobno statistički nezavisna. Epitet „naivni“ u nazivu klasifikatora ukazuje na to da navedeni uslov u stvarnosti često nije zadovoljen. U cilju pojednostavljenja izlaganja i demonstracije biće uvedena pretpostavka da se Bajesov klasifikator zasniva na MAP kriterijumu odlučivanja i da je u pitanju problem binarne klasifikacije, odnosno, detekcije definisan u (4.3). Prema ovom pravilu odlučivanja neobeleženi uzorak se pridružuje onoj klasi  $K_i$  kojoj odgovara najveća vrednost aposteriorne verovatnoće:

$$P(K_i | \mathbf{x}) = \frac{P(\mathbf{x} | K_i) P(K_i)}{P(\mathbf{x})}. \quad (6.1)$$

Dakle, aposteriorna verovatnoća je određena izglednošću, odnosno, verodostojnošću opservacije,  $P(\mathbf{x} | K_i)$ , i njenom apriornom verovatnoćom,  $P(K_i)$ . Pošto član  $P(\mathbf{x})$  u imeniku (6.1) ne zavisi od klase, isti ne utiče na ishod odluke i svodi se na normalizacioni faktor:

$$P(\mathbf{x}) = \sum_i P(K_i) P(\mathbf{x} | K_i). \quad (6.2)$$

Ključna pretpostavka naivnog Bajesovog klasifikatora jeste da se verovatnoća (6.1) može izračunati na osnovu nezavisne estimacije verodostojnosti pojedinačnih obeležja:

$$P(\mathbf{x} | K_i) = P(x_1, x_2, \dots, x_D | K_i) = \prod_{j=1}^D P(x_j | K_i), \quad (6.3)$$

Što proistiće iz usvojene pretpostavke o statističkoj nezavisnosti obeležja  $x_j$  u okviru jedne klase. Skup za obuku čini  $N$  uzoraka  $\mathbf{x}^{(n)}$ ,  $n = 1, \dots, N$ , koji pripadaju nekoj od  $K$  kategorija ili klasi  $K_i$ , pri čemu se za procenu  $P(x_j | K_i)$ ,  $\forall j$ , koristi samo  $N_i$  uzoraka iz skupa za obuku za koje se zna da pripadaju posmatranoj klasi. Ako nema drugih informacija o apriornim verovatnoćama pojedinih klasa  $P(K_i)$ , za njihove procene obično se usvajaju relativne učestanosti  $N_i/N$ ,  $N = N_1 + N_2 + \dots + N_K$ . Vrednosti  $P(x_j | K_i)$  u (6.3) mogu se procenjivati

## 6 NAIVNI BAJESOV KLASIFIKATOR

---

na razne načine (parametarske ili neparametarske), što zavisi i od toga kakvog tipa su obeležja  $x_j$  (numerička ili kategorička).

U ovoj vežbi model naivnog Bajesovog klasifikatora biće korišćen za implementaciju detektora neželjenih poruka elektronske pošte (engl. *spam*). Takav detektor svaki dokument ili primljenu poruku pre klasifikacije opisuje binarnim vektorom dužine  $D$ , koji sadrži podatak da li se neka od  $D$  vrednosti iz rečnika  $\mathcal{V}$  (skupa unapred definisanih reči) nalazi u poruci ili ne (1 označava da je reč prisutna, a 0 da nije). Obuka detektora podrazumeva prikupljanje  $N$  obeleženih uzoraka, odnosno, poruka za koje se zna da li predstavljaju *spam* ili ne, nakon čega se za svaku reč  $x_j$ ,  $j = 1, \dots, D$  izračunavaju verodostojnosti  $P(x_j | K_i)$  (koliko je izgledno da se određena reč nađe u poruci koja predstavlja *spam*, a koliko je to izgledno u poruci koja nije *spam*). Tako dobijene procene (obučen model) zatim se, zahvaljujući naivnoj pretpostavci (6.3), koriste tokom rada klasifikatora (testiranja) za izračunavanje  $P(\mathbf{x} | K_i)$  i konačno za detekciju *spam* poruka među novim porukama na osnovu vrednosti aposteriornih verovatnoća datih izrazom (6.1).

Ako se uvede notacija u kojoj pojedinačna obeležja  $x_j$  imaju ulogu indikatorskih promenljivih, čemu odgovara kodovanje uzorka (npr. poruka ili dokumenta) vrednostima iz skupa simbola  $\{0, 1\}$ , tako da komponente  $\mathbf{x} \in \{0, 1\}^D$  koje su jednake 1 označavaju prisustvo reči  $w_j$  (elemenata rečnika  $\mathcal{V}$  sa indeksom  $j$ ) u posmatranoj poruci a vrednosti 0 njihovo odsustvo, onda se verodostojnost svakog uzorka u klasi  $K_i$  na osnovu naivne pretpostavke o nezavisnosti obeležja u okviru konkretnе klase može predstaviti izrazom:

$$P(\mathbf{x} | K_i) = \prod_{j=1}^D b_{ij}^{x_j} (1 - b_{ij})^{(1-x_j)}, \quad x_j \in \{0, 1\}, \quad (6.4)$$

gde  $b_{ij}$  označava verovatnoću pojave reči  $w_j$  u poruci iz klase  $K_i$ , odnosno,  $P(x_j = 1 | K_i)$ , što ujedno znači i da važi  $P(x_j = 0 | K_i) = 1 - b_{ij}$ .

U razmatranom primeru sa detekcijom *spam* poruka bira se skup od  $D$  reči reprezentativan za *spam*, koji se naziva rečnikom i ima  $D = |\mathcal{V}|$  elemenata. Takođe važi i da su izglednosti poruka određene proizvodom Bernulijevih raspodela u izrazu (6.4).

Vrednosti parametara  $b_{ij}$  se procenjuju kao relativne frekvencije pojavljivanja *rezervisanih reči*  $w_j$ , reprezentativnih za *spam*, u svakoj od razmatranih klasa  $K_i$  skupa za obuku veličine  $N$ . Ako se sa  $n_i(w_j)$  označi ukupan broj pojavljivanja  $w_j$  u svih  $N_i$  uzoraka klase  $K_i$ , sledi da je:

$$P(x_j = 1 | K_i) \equiv P(w_j \text{ prisutna u poruci} | K_i) = b_{ij} = \frac{n_i(w_j)}{N_i}. \quad (6.5)$$

Treba razmotriti i potencijalni problem koji može da nastane kao posledica činjenice da je tokom obuke modela procenjeno da je neka od verovatnoća  $b_{ij}$  jednaka 0. Tada, ako bi se tokom rada klasifikatora ipak pojavio neki uzorak klase  $K_i$  koji sadrži reč  $w_j$  (što znači da je  $\hat{x}_j = 1$ ), zbog  $b_{ij} = 0$  bi i vrednost  $p(\hat{\mathbf{x}} | K_i)$  data izrazom (6.4), bila jednaka nuli, tako da ovaj uzorak ne bi mogao biti svrstan u klasu  $K_i$ . Primera radi, ako bi jedna od reči iz  $\mathcal{V}$  bila „nagrada“, i ako se ona ne bi javila ni u jednoj *spam* poruci u skupu za obuku, samim tim nijedna nova poruka koja sadrži reč „nagrada“ ne bi mogla biti klasifikovana kao *spam*, što nije opravdano, već je samo posledica nedovoljno reprezentativnog skupa za obuku. Stoga se prilikom implementacije i obuke naivnog Bajesovog klasifikatora unapred pribegava kompenzaciji nepovoljnih ishoda dodavanjem malih vrednosti verovatnoćama uslovnih raspodela za koje je prilikom obuke procenjeno da su jednake nuli.

Za kategorička obeležja kompenzacija se može postići korekcijom  $b_{ij} = 0$  koja se u literaturi naziva Laplasovom estimacijom. Primera radi, kod Bernulijeve raspodele vrednost procena zasnovanih na relativnim frekvencijama u (6.5) koriguje se tako što se u brojilac doda mala pozitivna vrednost  $\alpha$ , pri čemu se odgovarajuća vrednost srazmerna broju mogućih ishoda mora dodati i u imenilac kako bi suma verovatnoća svih događaja i dalje bila jednaka 1:

$$P(x_j = 1 | K_i)_\alpha = \frac{n_i(w_j) + \alpha}{N_i + 2\alpha}. \quad (6.6)$$

S obzirom na to da su vrednosti brojača  $n_i(w_j)$  celobrojne, često se uzima da je  $\alpha = 1$ , što znači da se korigovane vrednosti  $b_{ij_\alpha}$ ,  $\forall i$ , kreću između:  $1/(N_i + 2)$ , ako se tokom obuke modela element  $w_j$  iz rečnika  $\mathcal{V}$  nikada nije pojavio među uzorcima klase  $K_i$ ; i  $n_i(w_j)/N_i$ , što bi bila stvarna verovatnoća kada bi ukupan broj uzoraka određene klase težio beskonačnosti,  $N_i \rightarrow \infty$ . Prvoj vrednosti ili donjoj granici odgovaraju:  $P(x_j = 1 | K_i)_\alpha = 1/(N_i+2) \approx 0$ ;  $P(x_j = 0 | K_i)_\alpha = (N_i + 1)/(N_i + 2) \approx 1$ ;

Umesto binarnih obeležja koja označavaju prisustvo reči, za klasifikaciju poruka (dokumenata) mogu da se koriste i obeležja koja mere učestanost njihove pojave u tekstu. Zadržavajući isti model naivnog Bajesovog klasifikatora može se smatrati da se reči u tekstu pojavljuju nezavisno od drugih reči iz rečnika (što je značajno, ali korisno pojednostavljenje).

Svakom uzorku sa proizvoljnim brojem reči pridružuje se vektor  $\mathbf{x}$  sa  $D$  vrednosti  $x_j$  koje označavaju broj pojavljivanja reči  $w_j$  iz rečnika  $\mathcal{V}$ . Iz usvojene prepostavke o nezavisnosti obeležja sledi da konkretna opservacija  $\mathbf{x}$

## 6 NAIVNI BAJESOV KLASIFIKATOR

---

ima multinomnu raspodelu:

$$P(\mathbf{x}) = \frac{n}{x_1! x_2! \dots x_D!} \prod_{j=1}^{D=|\mathcal{V}|} p_j^{x_j}, \quad \sum_{j=1}^{D=|\mathcal{V}|} x_j = n, \quad (6.7)$$

sa  $n$  pojavljivanja  $D$  reči  $w_j$  iz rečnika  $\mathcal{V}$  u  $\mathbf{x}$ ; i verovatnoćama  $p_j$  pojavljivanja reči  $w_j$  u  $N$  dokumenata skupa za obuku. Slično važi i za uslovnu raspodelu:

$$P(\mathbf{x} | K_i) = \frac{n}{x_1! x_2! \dots x_D!} \prod_{j=1}^{D=|\mathcal{V}|} p_{ij}^{x_j}, \quad \sum_{j=1}^{D=|\mathcal{V}|} x_j = n, \quad (6.8)$$

pri čemu treba obratiti pažnju da verovatnoće  $p_{ij}$  sada zavise od klase  $K_i$  kojoj pripada  $N_i$  obeleženih poruka iz skupa za obuku koje su korištene za njihovu procenu.

Ako za svaku poruku u skupu za obuku sa  $l = 1, \dots, N$  uzoraka postoji odgovarajući opis  $\mathbf{x}^{(l)}$ , po analogiji sa izrazom (6.5) za procenu  $b_{ij}$  sledi da je:

$$P(w_j \text{ prisutna u poruci } | K_i) = p_{ij} = \frac{\sum_{l=1}^N x_j^{(l)} \cdot \mathbb{I}(\mathbf{x}^{(l)} \in K_i)}{\sum_{r=1}^{D=|\mathcal{V}|} \sum_{l=1}^N x_r^{(l)} \cdot \mathbb{I}(\mathbf{x}^{(l)} \in K_i)}, \quad (6.9)$$

gde je  $\mathbb{I}(\cdot)$  indikatorska funkcija, jednaka 1 ako argument sadrži tačan iskaz ili 0 u suprotnom.

Brojilac (6.9) je po vrednosti jednak ukupnom broju pojavljivanja reči  $w_j$  u svim dokumentima (porukama) klase  $K_i$ , kojih u skupu za obuku sa  $N$  obeleženih uzoraka ukupno ima  $N_i$ . Da se  $w_j$  u nekom od  $N_i$  dokumenata pojavljuje samo jednom ili da se broji samo prvo pojavljivanje, vrednost brojilaca u (6.9) bi bila jednak  $n_i(w_j)$ , tj. ista kao i u prethodnom slučaju, u (6.5). Imenilac (6.9) predstavlja normalizacioni faktor koji treba da obezbedi da je:  $\sum_j p_{ij} = 1$ , i predstavlja ukupan broj pojavljivanja svih reči iz rečnika  $\mathcal{V}$  u svih  $N_i$  dokumenata klase  $K_i$ .

Može se uočiti da će u izrazu (6.8) razlomak sa desne strane jednakosti biti različit za svako  $\mathbf{x}$ , ali da taj razlomak pri tome ne zavisi od indeksa  $i$  koji označava klasu uzorka. Odatle sledi da se vrednost razlomka neće menjati za različite vrednosti  $i$  i da samim tim kao konstanta neće uticati na konačnu odluku MAP klasifikatora kada se faktor  $p(\mathbf{x} | K_i)$  iz (6.8) uvrsti u polazni izraz (6.1) za aposteriornu verovatnoću svake od klasa  $i$ . Ovo omogućava da se konačni izraz

za diskriminantnu funkciju zasnovanu na MAP kriterijumu dodatno uprosti i svede na:

$$g_i(\mathbf{x}) = P(K_i) \prod_{j=1}^{|\mathcal{V}|} p_{ij}^{x_j}. \quad (6.10)$$

Može se primetiti i da  $p(x_j | K_i)$  u kontekstu (6.8) označava *raspodelu* frekvencija pojavljivanja reči  $w_j$  za klasu  $K_i$ , ali da se zahvaljujući usvojenoj naivnoj pretpostavci o nezavisnosti pojave reči (iz koje sledi korišćenje multinomne raspodele) njeno izračunavanje svodi na korišćenje izraza (6.9), koji predstavlja ukupnu (agregatnu) meru pojavljivanja  $w_j$  u uzorcima klase  $K_i$ .

## 6.1. Zadaci

### Zadatak 1

Dat je skup podataka<sup>3</sup> koji sadrži SMS poruke. U pitanju je tekstualni fajl koji u svakom redu sadrži po jedan SMS, a pre svakog SMS-a je naznačeno da li je u pitanju koristan SMS (*ham*) ili neželjen (*spam*, koji je najčešće reklamnog karaktera). U bazi se nalaze 5572 poruke, od kojih je oko 85% označeno sa *ham*.

Kod primene klasifikacije u obradi teksta, dokumenti se često posmatraju kao skupovi reči čiji se redosled zanemaruje (engl. *bag of words*), čime se izbegava dublje jezičko modelovanje. Dokumente, u ovom slučaju SMS poruke, moguće je modelovati na dva načina, opisana u teorijskom uvodu. U oba slučaja dokumenti se tretiraju kao uzorci koji se mogu predstaviti vektorom dimenzije  $D=|\mathcal{V}|$ , gde je  $\mathcal{V}$  rečnik koji se koristi, a  $|\mathcal{V}|$  broj različitih reči u rečniku. Kod prvog modela, koji se naziva *Bernulijevim*, svaki dokument predstavlja se vektorom binarnih vrednosti, gde  $i$ -ti element ima vrednost 1 ako  $i$ -ta reč iz rečnika  $\mathcal{V}$  postoji u modelovanom dokumentu, odnosno 0, ako ne postoji. Kod drugog modela, koji se naziva *multinomnim*, dokument je predstavljen vektorom celobrojnih nenegativnih vrednosti, gde  $i$ -ta vrednost govori koliko puta se u modelovanom dokumentu pojavila  $i$ -ta reč iz rečnika  $\mathcal{V}$ . U zavisnosti od korišćenog načina reprezentacije dokumenta, izglednost vrednosti obeležja se računa prema odgovarajućoj raspodeli, Bernulijevoj ili multinomnoj.

U Bernulijevom modelu koriste se procene zasnovane na izrazu (6.5) ili robustnoj varijanti datoј u (6.6), a u multinomnom modelu procene zasnovane na izrazu (6.9). U sledećem zadatku za predstavu dokumenta će biti korišćen Bernulijev model.

---

<sup>3</sup>Podaci su dostupni na: [www.dt.fee.unicamp.br/~tiago/smsspamcollection](http://www.dt.fee.unicamp.br/~tiago/smsspamcollection)

## 6 NAIVNI BAJESOV KLASIFIKATOR

---

1. Učitati skup podataka i podeliti ga na podskup za obuku i podskup za test, uzimajući za obuku prvih 5000 SMS poruka.
2. Originalne SMS poruke prvo treba obraditi tako da se obrišu znakovi interpunkcije, velika slova pretvore u mala i sl. Uklanjanje znakova interpunkcije iz niza karaktera i konverzija velikih slova u mala mogu se obaviti na sledeći način:

```
import string
for ch in sms:
    if ch in string.punctuation:
        sms = sms.replace(ch, " ")
sms=sms.lower()
```

3. Potom treba rastaviti SMS-ove na pojedinačne reči koristeći funkciju `split` koja rastavlja `string` na zadatom karakteru (podrazumevana vrednost je prazno mesto, odnosno `space`). Koliko ukupno ima reči u podskupu za obuku?
4. Formirati rečnik  $\mathcal{V}$  na osnovu liste reči iz skupa za obuku uz izbacivanje duplikata. Koliko reči sadrži rečnik? Koje reči su u rečniku? Postoji li mogućnost dodatnog unapređenja rečnika?
5. Izbaciti iz rečnika sve reči koje se u svim porukama zajedno javljaju manje od  $p$  puta. Neka je  $p = 2$ . Koliko reči sada ima rečnik? Da li treba uzeti veće  $p$ ?
6. Odrediti ukupan broj poruka  $N$  i broj poruka po klasi  $N_i$  u skupu za obuku.
7. Za svaku reč  $w_j$  iz oformljenog rečnika  $\mathcal{V}$ ,  $j = 1, \dots, D$ ,  $D = |\mathcal{V}|$  odrediti  $n_i(w_j)$ , što je broj poruka u klasi  $K_i$  koje je sadrže.
8. Proceniti izglednosti  $P(w_j \text{ prisutna u poruci} | K_i)$  za obe klase, prema izrazu:

$$P(w_j \text{ prisutna u poruci} | K_i) \equiv P(x_j = 1 | K_i) = \frac{n_i(w_j) + \alpha}{N_i + 2\alpha}.$$

Zašto se radi korekcija na osnovu Laplasove estimacije? Koliko treba da bude  $\alpha$ ?

9. Proceniti apriorne verovatnoće klase  $P(K_i)$ ,  $i = 1, \dots, K$ , na osnovu broja uzoraka  $N_i$  u klasi  $K_i$  i ukupnog broja uzoraka  $N$ , koristeći izraz:

$$P(K_i) = \frac{N_i}{N}.$$

10. Učitati, jedan po jedan, sve uzorke iz test skupa i svaki od njih predstaviti vektorom nula i jedinica  $\hat{\mathbf{x}}$  dužine  $D = |\mathcal{V}|$ , označavajući prisustvo reči  $w_j$  u odgovarajućem uzorku sa 1, a odsustvo sa 0.
11. Zatim, koristeći naredni izraz, proceniti aposteriorne verovatnoće za obe klase i na osnovu MAP kriterijuma doneti konačnu odluku kojoj klasi svaki element iz test skupa najverovatnije pripada:

$$\begin{aligned} P(K_i | \hat{S}) &= P(K_i | \hat{\mathbf{x}}) \sim P(\hat{\mathbf{x}} | K_i) P(K_i) \\ &\sim P(K_i) \prod_{j=1}^{|\mathcal{V}|} \left[ x_j P(x_j = 1 | K_i) + (1 - x_j) (1 - P(x_j = 1 | K_i)) \right]. \end{aligned}$$

Implementacija prethodnog izraza data je u sledećem kodu. Funkcija koja se koristi za množenje vektora element po element je `multiply`, a funkcija za množenje svih elemenata vektora koji joj je prosledjen je `prod`.

```
tmp = np.multiply(b_vec, n_i_likelihood) +
      np.multiply((1-b_vec),(1-n_i_likelihood))
p_i=apriori_i*tmp.prod()
```

12. Uporediti predviđene oznake pripadnosti klasama (labele) sa originalnim. Kolika je osetljivost ovog klasifikatora, odnosno, koliki procenat *spam* poruka će biti uspešno detektovan? Kolika je stopa lažnih pozitiva, odnosno, koliki procenat *ham* poruka će greškom biti označen kao *spam*? Kolika je ukupna tačnost klasifikatora na izdvojenom test skupu, ako se ona računa prema izrazu:

$$\text{Tačnost} = \frac{\text{broj prepoznatih } \textit{ham} + \text{broj prepoznatih } \textit{spam}}{\text{ukupan broj SMS}},$$

## 7. Linearna regresija

Linearna regresija predstavlja metodu nadgledanog učenja koja koristi se za predviđanje vrednosti kontinualne izlazne promenljive uz pretpostavku da se ta vrednost može dobiti kao linearna kombinacija vrednosti ulaznih obeležja. Formiranje modela linearne regresije svodi se na određivanje parametara pretpostavljene linearne zavisnosti. Mogućnosti primene linearog modela mogu se proširiti ukoliko se model primeni na neku (nelinearnu) transformaciju obeležja, čime se linearna zavisnost izlaza od obeležja gubi. Ukoliko se predviđa vrednost jedne skalarne izlazne promenljive, u pitanju je univarijantna linearna regresija, a ukoliko se predviđaju vrednosti više izlaznih promenljivih, u pitanju je multivarijantna linearna regresija. U ovom poglavljtu podrazumevaće se da se radi o univarijantnoj linearoj regresiji.

Hipoteza linearne regresije sa jednim obeležjem data je izrazom:

$$h(x) = \theta_0 + \theta_1 x, \quad (7.1)$$

gde  $h(x)$  predstavlja zavisnu promenljivu – onu čiju vrednost treba predvideti,  $x$  nezavisnu promenljivu – onu na osnovu koje se vrši predviđanje, dok su  $\theta_0$  i  $\theta_1$  parametri linearog modela. U slučaju  $D$  obeležja uvodi se vektor obeležja  $\mathbf{x} = [x_0, x_1, \dots, x_D]$ , pri čemu se obeležje  $x_0$  uvodi kao pomoćno radi kompaktnijeg zapisa u vektorskom obliku i njegova je vrednost za svaki uzorak jednaka 1. Uz tu pretpostavku, hipoteza linearne regresije u vektorskem obliku data je izrazom:

$$h(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x}. \quad (7.2)$$

Za dostupan skup za obuku, odnosno odgovarajuće uređene parove  $(\mathbf{x}^{(n)}, y^{(n)})$  zadatak linearne regresije je da odredi vrednosti parametara modela  $\boldsymbol{\theta} = (\theta_0, \theta_1, \dots, \theta_D)$  za koje  $h(\mathbf{x}^{(n)})$  ima vrednost što približniju  $y^{(n)}$ . Iz ovih razloga funkcija cene se definiše kao srednja kvadratna greška za  $N$  uzoraka u skupu za obuku:

$$J(\boldsymbol{\theta}) = \frac{1}{2N} \sum_{n=1}^N \left( h(\mathbf{x}^{(n)}) - y^{(n)} \right)^2, \quad (7.3)$$

i parametri modela određuju se njenom minimizacijom. Jedan od načina minimizacije funkcije cene je iterativni proces koji se naziva metoda opadanja gradijenta (engl. *gradient descent*). Ideja ove metode jeste da se nasumično inicijalizuju parametri modela  $\boldsymbol{\theta}$  i da se potom menjaju u pravcu opadanja gradijenta dok se ne ispuni uslov za prekid iterativnog procesa, što najčešće znači da se

izvrši unapred određeni broj iteracija ažuriranja parametara ili da razlika vrednosti funkcije cene u dva uzastopna koraka opadne ispod unapred utvrđenog praga. Pravilo za ažuriranje parametara modela u vektorskom obliku dato je izrazom:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \frac{1}{N} \mathbf{X}^\top (\mathbf{X}\boldsymbol{\theta} - \mathbf{y}), \quad (7.4)$$

gde je  $\alpha$  brzina učenja, a  $\mathbf{X}$  matrica  $N$  uzoraka opisanih sa po  $D$  obeležja. S obzirom na konveksnu funkciju cilja metoda opadanja gradijenta konvergira ka globalnom minimumu. Premala brzina učenja usporiće konvergenciju, dok povećanjem brzine učenja konvergencija može biti ugrožena jer minimum može biti preskočen. Preporuka je da se obeležja pre primene metode opadanja gradijenta skaliraju tako da imaju sličan opseg, što doprinosi ubrzaju konvergencije.

U okviru `Scikit-learn` biblioteke u modulu `linear_model` postoji implementacija linearne regresije u okviru klase `LinearRegression`. Metoda za inicijalizaciju je `LinearRegression`, metoda za obuku je `fit`, a metoda za testiranje `predict`. Parametri koji se mogu podešavati pri inicijalizaciji su sledeći:

- `fit_intercept`: ako je vrednost `True`, što je i podrazumevana vrednost, model će imati slobodan član, a ako je vrednost `False`, smatraće se da je on jednak 0.
- `normalize`: ako je `fit_intercept` postavljen na `True`, onda se i ovaj parametar može postaviti na vrednost `True` (ako je podrazumevana vrednost `False`) i tada će obeležja biti normalizovana oduzimanjem srednje vrednosti i deljenjem sa  $L_2$ -normom (normalizacija svodenjem na jediničnu normu). Ukoliko obeležja treba standardizovati, ovaj parametar je neophodno postaviti na `False` i izvršiti standardizaciju pre upotrebe klase `LinearRegression`.

Primer obuke i primene obučenog modela linearne regresije na test skupu:

```
regression_model = LinearRegression() #inicijalizacija
regression_model.fit(x_train, y_train) #obuka
y_predicted = regression_model.predict(x_test) #testiranje
```

Korelisanost obeležja i visoka dimenzionalnost otežavaju proces obuke kod linearne regresije, te se preporučuje smanjenje dimenzionalnosti selekcijom ili redukcijom obeležja, kao i primena regularizacije. Formiranje idealnog modela podrazumeva balansiranje između pristrasnosti modela (engl. *bias*) i njegove varijanse. *Pristrasnost* modela obučenog na određenom skupu uzoraka govori o tome koliku će sistematsku grešku on u proseku praviti prilikom

predikcije, odnosno, ukazuje na to u kojoj meri očekivana vrednost procene datim modelom za uzorak  $x$  odstupa od stvarne vrednosti za taj uzorak. Primera radi, nepristrasan regresor će nakon obuke na različitim skupovima podataka grešiti u predikciji na takav način da će prosek predviđenih vrednosti biti blizak tačnoj vrednosti koju treba predvideti (odnosno, u slučaju veoma velikog broja različitih skupova za obuku, prosečna vrednost predikcije biće jednaka tačnoj vrednosti). Relativno velika vrednost pristrasnosti ukazuje na to da je model u nedovoljnoj meri uočio pravilnosti i usvojio znanja koje se nalaze u podacima. S druge strane, *varijansa* modela ukazuje na to u kojoj meri vrednost predikcije za svaki uzorak varira zavisno od skupa podataka korišćenog za obuku. Složeniji modeli neretko imaju problem velike varijanse ukoliko se raspolaže manjim skupom podataka jer je neophodno proceniti veliki broj nepoznatih parametara prilikom obuke. Natprilagođen model imaće malu pristrasnost i veliku varijansu, dok će nedovoljno prilagođen model imati veliku pristrasnost, ali malu varijansu. Cilj je naći kompromis – obučiti model toliko da isprati trend u podacima i umanji sistematsku grešku pri predviđanju, a istovremeno sprečiti variranje vrednosti koje se predviđaju usled promene skupa za obuku. U ovu svrhu koriste se regularizacione tehnike čiji je cilj ograničenje procene koeficijenata i postizanje kompromisa između pristrasnosti i varijanse (malo povećanje pristrasnosti zarad smanjenja varijanse). Jedna od često korišćenih regularizacionih tehnika je *ridge* regresija, koja koristi sledeći oblik funkcije cene:

$$J(\boldsymbol{\theta}) = \frac{1}{2N} \sum_{n=1}^N \left( h(\mathbf{x}^{(n)}) - y^{(n)} \right)^2 + \lambda \sum_{i=1}^D \theta_i^2, \quad (7.5)$$

gde je  $\lambda$  regularizacioni parametar, a  $D$  ukupan broj obeležja. Regularizacioni parametar je nenegativan i najčešće se u praksi određuje unakrsnom validacijom. Novi član funkcije cene (u poređenju sa standardnom funkcijom cene linearne regresije) otežava izbor modela sa velikim vrednostima koeficijenata, koje obično ukazuju na natprilagođenje. Drugim rečima, ako dva modela ostvaruju istu srednju kvadratnu grešku, poželjno je opredeliti se za onaj sa manjim vrednostima koeficijenata. Alternativa *ridge* regresiji je *lasso* regresija, koja za razliku od *ridge* regresije minimizuje  $L_1$  normu umesto  $L_2$  norme. Funkcija cene u slučaju *lasso* regresije data je izrazom:

$$J(\boldsymbol{\theta}) = \frac{1}{2N} \sum_{n=1}^N \left( h(\mathbf{x}^{(n)}) - y^{(n)} \right)^2 + \lambda \sum_{i=1}^D |\theta_i|. \quad (7.6)$$

Prednost *lasso* regresije je u tome što, zahvaljujući korišćenju  $L_1$  norme, često

daje rešenja kod kojih je određen broj koeficijenata jednak nuli, čime se direktno realizuje selekcija obeležja.

U okviru **Scikit-learn** biblioteke u modulu `linear_model` postoje i implementacije *ridge* i *lasso* modela u klasama `Ridge` i `Lasso`. Metode za inicijalizaciju su `Ridge` i `Lasso`, dok su metode za obuku i za testiranje iste kao kod klase `LinearRegression`. Pored dva parametra koji su isti kao kod linearne regresije, parametri koji se još mogu podešavati u ove dve klase pri inicijalizaciji su sledeći:

- `alpha`: vrednost regularizacionog parametra čija podrazumevana vrednost je 1.
- `solver`: algoritam za optimizaciju; podrazumevana vrednost je `auto`, što znači da će se odabratи najpogodniji algoritam prema vrsti podataka za obuku, a jedna od mogućnosti je i `lsqr`, što odgovara primeni minimizacije srednje kvadratne greške.
- `max_iter`: maksimalni broj iteracija algoritma za optimizaciju.

Primer obuke i primene *ridge* i *lasso* modela na test skupu:

```
ridge_model = Ridge(alpha=1.0) #inicijalizacija
ridge_model.fit(x_train, y_train) #obuka
y_predicted = ridge_model.predict(x_test) #testiranje

lasso_model = Ridge(alpha=0.1) #inicijalizacija
lasso_model.fit(x_train, y_train) #obuka
y_predicted = lasso_model.predict(x_test) #testiranje
```

Selekcija obeležja može se vršiti direktno, a postoje i tehnike selekcije obeležja unapred i unazad. Ukoliko se rešava problem sa  $D$  obeležja, selekcija obeležja unapred započinje obukom  $D$  regresora nad skupom za obuku, a svaki od njih koristi samo jedno od dostupnih  $D$  obeležja za predviđanje vrednosti iste izlazne promenljive. Obeležje koje je koristio regresor čija predviđanja najmanje odstupaju u odnosu na stvarne vrednosti, odnosno, čija je suma kvadrata razlike tačnih i predviđenih vrednosti zavisne promenljive (RSS greška) najmanja (Odeljak 2.8), predstavlja prvo selektovano obeležje u redukovanim skupu. Procedura se nastavlja za preostalih  $D - 1$  obeležja, pri čemu svako od njih pored prvog selektovanog obeležja figuriše u jednom od  $D - 1$  novih modela sa dva obeležja. Sledеće obeležje koje ulazi u skup selektovanih je ono čiji model rezultuje najmanjom RSS. Postupak se nastavlja za preostalih  $D - 2$  obeležja. Proces se zaustavlja na osnovu izabranog kriterijuma, na primer, dodavanjem novog

obeležja nema značajnog smanjenja između vrednosti RSS u dva uzastopna koraka ili se vrednost RSS čak povećava.

Druga mogućnost je selekcija obeležja unazad, kada se polazi od modela koji koristi svih  $D$  obeležja i odbacuje jedno po jedno. Kriterijum za eliminaciju obeležja je njegov značaj, koji se određuje na osnovu standardne greške procene i  $t$ -statistike. Naime, prilikom procene svakog parametra  $\theta_i$  utvrđuje se interval poverenja  $[\theta_i - 2SE(\theta_i), \theta_i + 2SE(\theta_i)]$  koji sa verovatnoćom od 95% sadrži stvarnu vrednost parametra, gde je  $SE(\theta_i)$  standardna greška procene vrednosti parametra  $\theta_i$ . Standardna greška definisana je izrazom:

$$SE^2(\theta_i) = \frac{\sigma^2(\theta_i)}{N}, \quad (7.7)$$

gde je  $\sigma^2(\theta_i)$  varijansa estimacije parametra  $\theta_i$  i ukazuje na to kako procena vrednosti parametra varira pri izboru novog skupa za obuku. Za utvrđivanje značaja datog parametra postavlja se hipoteza da ne postoji zavisnost između obeležja uz  $\theta_i$  i promenljive koja se predviđa regresijom, odnosno, hipoteza da je  $\theta_i = 0$ . Za proveru hipoteze koristi se  $t$ -statistika:

$$t = \frac{\theta_i - 0}{SE(\theta_i)}. \quad (7.8)$$

Značaj obeležja, odnosno  $p$ -vrednost, predstavlja verovatnoću da se na slučaj dobije vrednost jednaka  $|t|$  ili veća. Ako je npr.  $p$ -vrednost manja od 0.01, to znači da je verovatnoća da je testirana hipoteza ispravna manja od 1%. Prema tome, mala vrednost  $p$  ukazuje na to da je posmatrano obeležje značajno za procenu izlazne promenljive. S druge strane, veća  $p$ -vrednost za neko obeležje ukazuje da je veća verovatnoća da je hipoteza ispravna, odnosno da to obeležje može da se odbaci. Kod selekcije obeležja unazad redom se odbacuju obeležja sa najvećom  $p$ -vrednošću, sve dok svako od obeležja ne poprimi  $p$ -vrednost manju od nekog definisanog praga. Međutim, ovakvi pristupi selekciji obeležja su dosta zahtevni i nepogodni za primenu kada je broj obeležja velik.

Za računanje  $p$ -vrednosti u `Scikit-learn` biblioteci ne postoji ugradena funkcija, tako da se savetuje kreiranje modela linearne regresije iz biblioteke `statsmodel` korišćenjem klase `OLS`, te računanje i tabelarni prikaz koeficijenata, njihovih standardnih grešaka procene i  $p$ -vrednosti korišćenjem funkcije `summary`.

Primer koda za dobijanje tabele sa  $p$ -vrednostima obeležja:

```
import statsmodels.api as sm
X = sm.add_constant(x_train)
model = sm.OLS(y_train, X).fit()
model.summary()
```

Kada veza između zavisne i nezavisnih promenljivih nije linearna, model linearne regresije u svom osnovnom obliku nije pogodan. U ovom slučaju može da se koristi pomenuto uopštenje linearног modela korišćenjem nelinaernih transformacija obeležja (npr. stepene funkcije) kao i interakcije između njih. Naime, do sada razmatrana osnovna hipoteza linearne regresije podrazumevala je da je efekat jednog od obeležja na promenljivu čija se vrednost predviđa potpuno nezavisan od efekta nekog drugog obeležja. Međutim, kada su obeležja međusobno zavisna, poželjno je uzeti u obzir i njihov zajednički efekat. Primera radi, hipoteza koja uzima u obzir tri obeležja, kao i interakciju između njih, a u kojoj se, radi veće opštosti, javljaju i kvadrati pojedinih obeležja, izgledala bi ovako:

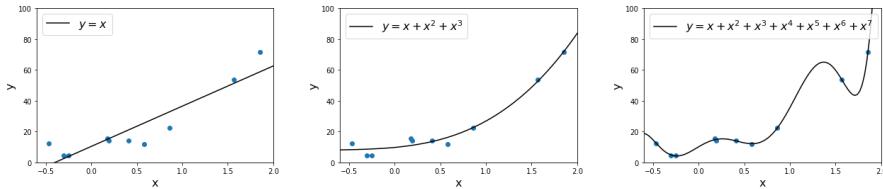
$$\begin{aligned} h(\mathbf{x}) = & \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 \\ & + \theta_4 x_1 x_2 + \theta_5 x_1 x_3 + \theta_6 x_2 x_3 \\ & + \theta_7 x_1^2 + \theta_8 x_2^2 + \theta_9 x_3^2. \end{aligned} \quad (7.9)$$

Uključivanjem interakcija uticaj nekih od obeležja, gledanih posebno, može da se smanji ( $p$ -vrednost postane velika). Ipak te članove hipoteze ne treba tada odbacivati jer bi se time otežala interpretacija modela u kontekstu analize uticaja svakog obeležja posebno i njihove interakcije. Izbor nelinearnih transformacija obeležja je velik i nije jednostavno ispitati koja kombinacija obeležja i transformacije će dati najbolje rezultate. Pored toga, povećanje broja parametara znači i složeniji model zbog čega se korišćenje regularizacije preporučuje da bi se izbeglo natprilagođenje. Na Slici 7.1 su prikazani uzorci i modeli sa različitim hipotezama dobijeni korišćenjem tih uzoraka. S obzirom na rasipanje uzoraka u prostoru obeležja, lako se zaključuje da veza između  $x$  i  $y$  nije linearna (Slika 7.1, primer levo), zbog čega bi se preporučilo uvođenje nelinearnih transformacija datog obeležja  $x$  u hipotezu modela. Međutim, značajno usložnjavanje modela dovodi do natprilagođenja modela, odnosno potpuno prilagođenje uzorcima korišćenim za obuku (Slika 7.1, primer desno).

Kada je potrebno upotrebiti hipotezu sa interakcijama među obeležjima i/ili stepene funkcije obeležja, za pripremu traženih novih obeležja koja će činiti sve kombinacije proizvoda po dva od dostupnih obeležja, kao i stepene pojedinačnih obeležja, koristi se klasa `PolynomialFeatures` iz `preprocessing`

## 7 LINEARNA REGRESIJA

---



Slika 7.1: Ilustracija regresionog modela sa različitim hipotezama.

modula. Ova klasa za inicijalizaciju koristi metodu `PolynomialFeatures`, za računanje ukupnog broja obeležja koji će biti formiran metodu `fit`, a za formiranje novih obeležja od postojećih, metodu `transform`. Pri inicijalizaciji se podešavaju sledeći parametri:

- **degree**: formira polinomijalna obeležja do stepena zadatog vrednošću ovog parametra; dakle ako je vrednost 3, biće formirana obeležja prvog, drugog i trećeg stepena.
- **interactions\_only**: postavljanjem vrednosti na `True` biće podržane samo interakcije među obeležjima, ne i viši stepeni obeležja.
- **include\_bias**: uvodi i konstantu kao obeležje; kada se primenjuje radi dobijanja obeležja za linearnu regresiju, preporučuje se da se vrednost ovog parametra podesi na `False`, te da se po potrebi u inicijalizaciju modela linearne regresije uključi slobodan član .

Primer obuke i primene modela linearne regresije sa hipotezom koja sadrži interakcije među obeležjima na test skup:

```

poly = PolynomialFeatures(interaction_only=True,
                          include_bias=False)
x_inter = poly.fit_transform(X)
x_train, x_test, y_train, y_test = train_test_split(X, y,
  test_size=0.05, random_state=42)
regression_model = LinearRegression() #inicijalizacija
regression_model.fit(x_train, y_train) #obuka
y_predicted = regression_model.predict(x_test) #testiranje

```

## 7.1. Zadaci

### Zadatak 1

Implementirati algoritam linearne regresije sa osnovnom hipotezom nad uzorcima jednodimenzionalnog prostora. Koraci su sledeći:

1. Učitati biblioteke NumPy, Matplotlib i kolekciju funkcija pyplot, kao i biblioteku SciPy.
2. Definisati klasu pod nazivom `LinearRegressionUsingGD`, koja će sadržati metode `__init__` za inicijalizaciju, `fit` za obuku i `predict` za primenu obučenog modela.
3. U okviru metode inicijalizacije definisati maksimalan broj iteracija algoritma opadanja gradijenta i vrednost brzine učenja.
4. U okviru metode za obuku inicijalno postaviti vrednosti parametara hipoteze ( $\theta_0$  i  $\theta_1$ ) na 0, a potom proći unapred zadati broj iteracija. U svakoj iteraciji izvršiti ažuriranje vrednosti parametara hipoteze prema izrazu (7.4) za algoritam opadajućeg gradijenta i izračunati funkciju cene definisanu izrazom (7.3).
5. U metodi za primenu obučenog modela izračunati rezultat regresije na osnovu parametara hipoteze utvrđenih pri obuci.
6. Generisati skup za obuku formiranjem uređenih parova  $(x_i, y_i)$ . Vektor  $\mathbf{x}$ , dužine 100, sadrži slučajne brojeve koji će predstavljati vrednost obeležja za 100 uzoraka, a vektor  $\mathbf{y}$ , koji će predstavljati željene izlaze regresije, dobija se prema izrazu  $\mathbf{y} = 2 + 3 \cdot \mathbf{x}$ .
7. Generisati test skup formiranjem uređenih parova  $(p_i, q_i)$ . Vektor  $\mathbf{p}$  je dužine 20 i sadrži slučajne brojeve, a vektor  $\mathbf{q}$ , koji će predstavljati željene izlaze regresije za uzorce iz  $\mathbf{p}$ , dobija se prema izrazu  $\mathbf{q} = 2 + 3 \cdot \mathbf{p}$ .
8. Prikazati uzorce za obuku na grafiku rasipanja (`scatter plot`).
9. Inicijalizovati model linearne regresije sa željenim parametrima za broj iteracija i brzinu učenja pa obučiti model linearne regresije sa podacima za obuku.
10. Primeniti model na podacima za obuku i na podacima za testiranje.

## 7 LINEARNA REGRESIJA

---

11. Uporediti  $R^2$  grešku dobijenu prilikom primene modela na podacima za obuku i prilikom primene na podacima za testiranje.
12. Proveriti koliki su dobijeni koeficijenti u obučenom modelu linearne regresije. Da li odstupaju od koeficijenata definisanih prilikom generisanja vektora  $\mathbf{y}$ ?
13. Prikazati grafik opadanja vrednosti funkcije cene kroz iteracije. Da li je odabrana brzina učenja odgovarajuća? Da li je odabran broj iteracija dovoljan?
14. Prikazati stvarne i predviđene vrednosti test uzoraka na grafiku rasipanja. Izvesti zaključak koliko je formirani model dobar.
15. U generisani skup za obuku i testiranje dodati varijansu, odnosno na vrednosti  $\mathbf{y}$  i  $\mathbf{p}$  dodati slučajne brojeve iz Gausove raspodele, pa ponoviti obuku i testiranje. Da li je model koji je formiran na osnovu nove baze dobar? Do kakvog zaključka se dolazi za  $R^2$  meru?

### Zadatak 2

Implementirati algoritam za predviđanje vidljivosti u kilometrima na osnovu dostupnih podataka o drugim parametrima vremenske prognoze. Koristiti bazu podataka<sup>4</sup> koja sadrži informacije o parametrima vremenske prognoze za Segedin, grad u Mađarskoj, sakupljene u periodu od 2006. do 2016. godine na svakih sat vremena.

1. Učitati podatke u `DataFrame`. Koliko ima uzoraka? Koliko ima obeležja i kog su tipa?
2. Izvršiti dopunu podataka koji nedostaju i pretvoriti kategorije obeležja koje označava padavine u numeričke vrednosti.
3. Za početak odbaciti preostala kategorička obeležja i podatak o datumu i vremenu.
4. Izvršiti statističku analizu obeležja i prikazati raspodelu vrednosti za obeležje *vidljivost* čije će se vrednosti predviđati.
5. Realizovati funkciju koja računa različite mere uspešnosti regresora, a koja će biti korišćena nakon svake obuke i testiranja.

---

<sup>4</sup>Podaci su dostupni na: [www.kaggle.com/budincsevity/szeged-weather](http://www.kaggle.com/budincsevity/szeged-weather)

6. Podeliti uzorke na dva podskupa, za obuku i testiranje, tako da test podskup sadrži 5% nasumično izabralih uzoraka.
7. Inicijalizovati i obučiti model linearne regresije koristeći osnovnu hipotezu. Testirati i evaluirati uspešnost modela. Kolika je srednja apsolutna greška, a kolika  $R^2$  mera?
8. Izvršiti selekciju obeležja unazad sukcesivno izbacujući obeležja čija je  $p$ -vrednost veća od 0,01. Koliko obeležja je ostalo?
9. Izvršiti standardizaciju obeležja pa ponoviti obuku i testiranje modela linearne regresije sa osnovnom hipotezom. Da li je došlo do poboljšanja modela? Šta se promenilo?
10. Proveriti međusobnu korelisanost obeležja. Postoji li potreba za korišćenjem drugačije hipoteze linearne regresije?
11. Ispitati da li se model može poboljšati korišćenjem drugačijih hipoteza. Da li modele sa interakcijama između obeležja treba formirati koristeći standardizovana obeležja ili originalna?
12. Ispitati da li je moguće poboljšati model korišćenjem regularizacije. Varijirati vrednosti regularizacionog parametra. Kako je regularizacija uticala na vrednosti koeficijenata? Da li regularizacione modele treba primeniti na originalnim ili na standardizovanim obeležjima? Da li se model poboljšao?
13. Koji model regresije se ispostavio najboljim za dati problem? Zašto?
14. Ponoviti proceduru koristeći i podatke o datumu i vremenu kada se vrši merenje. Da li su se ovi podaci ispostavili relevantnim za dati problem?

## 8. Logistička regresija

Logistička regresija definiše statistički model koji se koristi za predviđanje aposteriorne verovatnoće klase za dati uzorak, odnosno, predviđanje vrednosti binarne zavisne promenljive. U slučaju binarne klasifikacije (0/1), model linearne regresije mogao bi biti primenjen na klasifikaciju postavljanjem odgovarajućeg praga odlučivanja, npr.  $T = 0.5$  za izlaznu promenljivu. Međutim, opseg izlazne promenljive nije ograničen na  $[0, 1]$  i sam model je zavisan od uzorka u skupu za obuku, jer dodavanjem novih uzoraka granica odlučivanja može značajno da se promeni. U slučaju više klase potrebno je podeliti opseg izlaznih vrednosti na više podopsega, pri čemu se podrazumeva neki poredak među njima, što ne mora nužno biti slučaj (npr. kod klasifikacije simptoma na blage, umerene i teške postoji prirodan poredak, ali ne i kod klasifikacije životinja na osnovu fotografija). Zaključuje se da linearna regresija nije pogodan model za klasifikaciju, iako bi mogla da se u svom osnovnom obliku primeni na klasifikacione probleme.

Kako bi se uočeni problem prevazišao, umesto osnovnog modela linearne regresije potrebno je naći funkciju čiji kodomen pokriva opseg  $[0, 1]$  i koja omogućava interpretaciju izlaza modela kao verovatnoće pripadnosti odgovarajućoj klasi. Logistička regresija definiše se kao generalizovan linearni regresioni model koji za dati uzorak na ulazu predviđa verovatnoću da uzorak pripada klasi  $y = 1$  koristeći pogodnu nelinearnu funkciju ulaznih promenljivih  $(x_1, x_2, \dots, x_D) = \mathbf{x}$ . Nelinearna funkcija koja se koristi u modelu logističke regresije je sigmoidalna ili logistička funkcija  $g(x)$  (Slika 8.1), definisana izrazom:

$$g(x) = \frac{1}{1 + e^{-x}}. \quad (8.1)$$

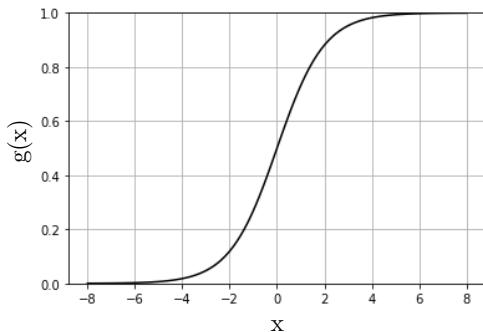
Primera radi, logistička regresija može se iskoristiti za predviđanje verovatnoće da je bankovna transakcija izvršena zloupotrebljenom platnom karticom izračunavanjem vrednosti sigmoidalne funkcije nad linearnom kombinacijom različitih obeležja kao što su datum transakcije, količina novca i mesto sa kog je izvršena. Formalno, hipoteza kod modela logističke regresije može se definisati sledećim izrazom:

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = g(\boldsymbol{\theta}^T \mathbf{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^T \mathbf{x}}} = P(y = 1 | \mathbf{x}, \boldsymbol{\theta}) = \mathbb{E}(y | \mathbf{x}). \quad (8.2)$$

Obuka modela logističke regresije svodi se na pronalaženje optimalnih koeficijenata  $\boldsymbol{\theta}$ , npr. maksimizacijom verodostojnosti skupa za obuku pod pretpostavkom da izlaz modela  $y$  predstavlja verovatnoću pripadnosti datog uzorka

klasi 1. Ovaj problem ekvivalentan je minimizaciji negativne vrednosti logaritma verodostojnosti i može se rešiti metodom opadanja gradijenta. Stoga, izraz za funkciju cene svodi se na:

$$J(\boldsymbol{\theta}) = -\frac{1}{N} \sum_{n=1}^N \left( y^{(n)} \log(h_{\boldsymbol{\theta}}(\mathbf{x}^{(n)})) + (1 - y^{(n)}) \log(1 - h_{\boldsymbol{\theta}}(\mathbf{x}^{(n)})) \right). \quad (8.3)$$



**Slika 8.1:** Sigmoidalna funkcija

Za rešavanje problema sa više klasa može se primeniti više modela logističke regresije, po principu svaki protiv svakog (engl. *one vs. one*) i jedan protiv svih (engl. *one vs. all* ili *one vs. rest*). U prvom slučaju formira se  $\binom{K}{2}$  klasifikatora i uzorak se dodeljuje onoj klasi koja je najveći broj puta bila uspešnija u nadmetanju klasifikatora po parovima. U drugom slučaju konstruiše se  $K$  klasifikatora, a uzorak se dodeljuje klasi za koju je izlaz logističke regresije najveći. Druga mogućnost je primena multinomijalne logističke regresije, ali logistička regresija generalno nije čest izbor za rešavanje problema sa više od dve klase.

Za implementaciju logističke regresije u okviru **Scikit-learn** biblioteke koristi se klasa **LogisticRegression**, koja se nalazi u modulu **linear\_model** i ima sledeće parametre:

- **fit\_intercept**: Određuje da li će model sadržati slobodan član. Može biti **True** ili **False**.
- **class\_weight**: Ukoliko se vrednost ne podesi, uzorci svih klasa imaju isti značaj, odnosno težina im je 1. Ukoliko se odabere vrednost **balanced**,

težine se automatski postavljaju obrnuto proporcionalno zastupljenosti klase. Moguće je dodeliti i konkretne težine svakoj od klasa u formi tzv. rečnika kao `class_label:weight`.

- **solver:** algoritam za rešavanje optimizacionog problema. Moguće vrednosti su: `newton-cg`, `lbfgs`, `liblinear`, `sag`, `saga`.
- **multi\_class:** ako je vrednost `ovr`, problem sa više klasa rešavaće se po principu jedan protiv svih, ako je vrednost `multinomial`, biće korišćena multinomijalna logistička regresija, a ako je vrednost `auto`, izbor će biti `multinomial` osim kada izbor algoritma za rešavanje optimizacionog problema to ne dozvoljava.

Ostali parametri se uglavnom tiču dodatnih podešavanja u zavisnosti od odabranog algoritma za rešavanje optimizacionog problema. Kao i regresioni model, i klasifikacioni modeli imaju fazu inicijalizacije (metoda `LogisticRegression`), fazu primene obučenog modela (metoda `fit`) i fazu testiranja (metoda `predict`). Pri inicijalizaciji algoritma postavljaju se vrednosti hiperparametara. Primer obuke i primene klasifikatora logističke regresije na test podacima dat je u kodu koji sledi.

```
classifier = LogisticRegression(solver='newton-cg')
classifier.fit(x_train, y_train)
y_predicted = classifier.predict(x_test)
```

### 8.1. Zadaci

#### Zadatak 1

Implementirati algoritam za klasifikaciju pacijenata sa tumorom dojke na one sa malignim i one sa benignim oboljenjem. Koristiti bazu podataka<sup>5</sup>, koja sadrži informacije o različitim karakteristikama tumora utvrđenim iglenom biopsijom. Za svakog pacijenta u bazi određena je i klasna labela na osnovu patološkog nalaza.

1. Učitati podatke u `DataFrame`. Koliko ima uzoraka? Koliko ima obeležja i kog su tipa?
2. Izbaciti obeležja koja nisu potrebna. Ima li takvih?

---

<sup>5</sup>Podaci su dostupni na: [www.kaggle.com/uciml/breast-cancer-wisconsin-data](http://www.kaggle.com/uciml/breast-cancer-wisconsin-data)

3. Pretvoriti kategorije klasnih labela u numeričke vrednosti zbog korišćenja ugrađenih funkcija koje to zahtevaju.
4. Proveriti koliko uzoraka ima u kojoj klasi, kao i da li ima nedostajućih vrednosti.
5. Realizovati funkciju koja računa različite mere uspešnosti klasifikatora na osnovu date matrice konfuzije.
6. Podeliti uzorce na tri podskupa (za obuku, validaciju i testiranje) tako da test podskup sadrži 10% nasumično izabranih uzoraka iz baze, a validacioni sadrži 10% od preostalih uzoraka.
7. Inicijalizovati i obučiti model logističke regresije. Koristeći validacioni skup odrediti optimalne hiperparametre, što se svodi na odabir algoritma za rešavanje optimizacionog problema i eventualno podešavanje težina za klase. Na koju meru uspešnosti ima najviše smisla osloniti se u problemu koji se ovde rešava? Da li je podjednako loše proglašiti zdravog pacijenta bolesnim i bolesnog pacijenta zdravim?
8. Prikazati ROC krivu i analizirati rezultate na validacionom skupu za različite pragove odlučivanja. Koji prag je najbolje izabrati?
9. Obučiti konačan model logističke regresije sa utvrđenim parametrima klasifikatora, a potom ga testirati i evaluirati na test skupu. Koliko uzoraka je pogrešno klasifikovano? Koliko je klasifikator u proseku bio siguran kod ispravno donesenih odluka? Koliko je u proseku bio siguran u odluku u slučaju pogrešno klasifikovanih uzoraka? Kakve su mere uspešnosti?
10. Kako bi izgledale mere uspešnosti da je prag odluke umesto podrazumevanih 0,5 bio postavljen na vrednost koja je utvrđena kao najbolja na validacionom skupu?

## 9. Klasifikator metodom $k$ najbližih suseda

Metoda  $k$  najbližih suseda (engl.  $k$  nearest neighbors –  $k$ NN) korišćena je za neparametarsku procenu gustine raspodele u poglavlju 5. Međutim, ova metoda najčešće se koristi kao neparametarska metoda klasifikacije jer na veoma jednostavan način aproksimira Bajesov klasifikator. Metodu  $k$ NN moguće je primeniti i na regresione probleme, ali se u tom kontekstu ređe koristi. Ova metoda spada u grupu metoda kasnog učenja (engl: *lazy learning*), koje podrazumevaju odlaganje obrade uzorka za obuku do trenutka kada treba klasifikovati neobeleženi uzorak. U pitanju je intuitivan algoritam koji klasificuje nepoznati uzorak  $\mathbf{x}$  na osnovu klasne pripadnosti susednih uzoraka  $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(k)}$  iz skupa za obuku. Elementi potrebni za primenu  $k$ NN metode su sledeći:

- Obeleženi skup uzorka za obuku (svaki uzorak  $\mathbf{x}^{(n)}$  ima poznatu klasnu pripadnost, odnosno, dodeljenu klasnu labelu)
- Celobrojni parametar  $k$  (broj najbližih suseda koji se uzima u obzir pri odlučivanju)
- Metrika za utvrđivanje rastojanja između dva uzorka (vektora obeležja)

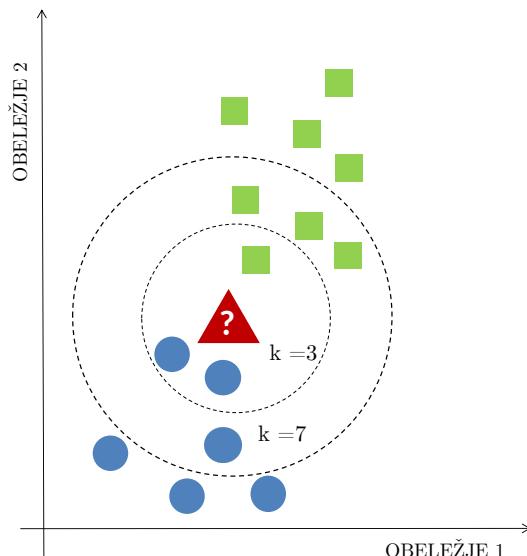
Prilikom klasifikacije neobeleženog uzorka, na osnovu odabrane metrike se meri njegova udaljenost od svakog uzorka iz skupa za obuku. Klasa neobeleženog uzorka predviđa se na osnovu klasne pripadnosti  $k$  uzorka iz skupa za obuku koji su najbliži neobeleženom uzorku, jer se pretpostavlja da je dati uzorak sličniji uzorcima iz skupa za obuku koji su mu bliži u prostoru obeležja. U osnovnoj verziji algoritma, neobeleženi uzorak dodeljuje se klasi koja je u većini među  $k$  najbližih uzoraka (Slika 9.1). Izračunata rastojanja se odbacuju do sledećeg zahteva za klasifikacijom, kada se postupak ponavlja za naredni uzorak.

Izbor metrike prvenstveno zavisi od tipa obeležja. Ukoliko obeležja imaju realne vrednosti, čest izbor su  $p$ -norme, koje su objašnjene u Poglavlju 2. Rastojanje od uzorka  $\mathbf{x}$  do uzorka  $\mathbf{y}$  definisano je kao  $p$ -norma njihove razlike:

$$d_{\mathbf{xy}} = \left( \sum_{i=1}^D |x_i - y_i|^p \right)^{\frac{1}{p}}, \quad (9.1)$$

gde su  $x_i$  i  $y_i$  vrednosti  $i$ -tog obeležja uzorka  $\mathbf{x}$  i  $\mathbf{y}$ , redom. Kada je  $p = 2$  dobija se euklidsko rastojanje. Za obeležja sa celobrojnim vrednostima kao i za binarna obeležja čest izbor je Hemingovo rastojanje, koje predstavlja ideo

broja pozicija (dimenzija) sa različitim vrednostima u ukupnom broju pozicija u vektoru obeležja. Na primer, Hemingovo rastojanje između petodimenzionalnih uzoraka  $[2, 4, 3, 1, 7]$  i  $[3, 4, 6, 1, 7]$  iznosi  $\frac{2}{5}$  jer se dati vektori razlikuju po prvoj i trećoj dimenziji, a ukupna dimenzionalnost je 5. Za binarna obeležja čest izbor je i Žakarova metrika, koja se računa kao ideo broja nejednakih obeležja u broju nenultih obeležja. Na primer, Žakarovo rastojanje između petodimenzionalnih uzoraka  $[1, 0, 0, 1, 0]$  i  $[0, 0, 1, 1, 0]$  iznosi  $\frac{2}{3}$  jer se razlikuju po prvoj i trećoj dimenziji, a na prvoj, trećoj i četvrtoj postoji nenulto obeležje bar kod jednog od uzoraka.



**Slika 9.1:** Primer različitog ishoda klasifikacije neobeleženog uzorka (trougao) za različit izbor parametra  $k$  (za  $k = 3$  uzorak će biti dodeljen klasi „krug“ a za  $k = 7$  klasi „kvadrat“).

Jedno od glavnih pitanja kod primene ove metode klasifikacije jeste kako izabrati parametar  $k$ . Usvajanjem relativno velikih vrednosti parametra  $k$  postiže se da granice regiona odlučivanja budu bliže glatkim, ali se povećava računarska složenost i gubi se lokalni karakter estimacije, pošto na odluku o klasnoj pripadnosti neobeleženog uzorka utiču i uzorci koji su daleko od njega. S druge strane, za male vrednosti parametra  $k$  procena je sklonu šumu, odnosno uticaju faktora koji ne nose nikakvu informaciju. Kada je  $k = 1$ , dobija se poseban slučaj  $k$ NN

## 9 KLASIFIKATOR METODOM $K$ NAJBLIŽIH SUSEDА

---

klasifikatora, tzv. klasifikator metodom najbližeg suseda. Dakle, neobeleženi uzorak smešta se u klasu u kojoj se nalazi njemu najbliži uzorak iz skupa za obuku. Iako je ovo vrlo jednostavan klasifikator, on daje dobre rezultate kada je broj uzoraka u skupu za obuku velik. Međutim, velika količina podataka čini proces određivanja klase nepoznatog uzorka dugotrajnim i zahteva veliki memorijski prostor za pamćenje skupa za obuku. Stoga se pre primene ove metode u visokodimenzionalnim problemima savetuje smanjenje dimenzionalnosti (selekcija ili redukcija obeležja), kao i korišćenje efikasnijih struktura podataka poput  $k$ -D stabala.

Iako ređe,  $k$ NN metoda može se koristiti i u rešavanju regresionih problema. Postavka je identična kao u slučaju klasifikatora na bazi  $k$ NN, a vrednost neobeleženog uzorka se računa kao prosek vrednosti  $k$  najbližih uzoraka. Jedna od ideja za poboljšanje rezultata je i ponderisanje rastojanja – varijanta koja tretira bliže susede kao relevantnije i dodeljuje im veću težinu.

Implementacija  $k$ NN metode u `Scikit-learn` biblioteci za potrebe klasifikacije nalazi se u okviru klase `KNeighborsClassifier`, a za potrebe regresije u okviru klase `KNeighborsRegressor`. Obe klase nalaze se u modulu `neighbors` i imaju iste parametre, koji su objašnjeni u nastavku.

- **`n_neighbors`**: broj suseda koji se razmatraju, tj. parametar  $k$ . Kako bi se u slučaju binarne klasifikacije izbegao nerešen rezultat pri glasanju za dodelu klase, treba birati neparno  $k$ . U slučaju većeg broja klasa, postoji više načina za donošenje odluke u slučaju izjednačenog rezultata, a u ovoj implementaciji odluka zavisi od redosleda pojave klasa u podacima za obuku.
- **`weights`**: težine kojima se ponderišu susedni uzroci. Ovaj parametar služi za određivanje značaja svakog od  $k$  najbližih suseda u odlučivanju. Ukoliko svi susedi treba da imaju jednak uticaj, vrednost parametra se postavlja na `uniform`, što je ujedno i podrazumevana vrednost. Druga mogućnost je `distance`, kada je značaj susednih uzoraka obrnuto proporcionalan njihovoj udaljenosti od neobeleženog uzorka, te bliži uzorci imaju veći uticaj na odluku o klasi ili vrednosti neobeleženog uzorka. Poslednja opcija je da se eksplisitno navedu željene vrednosti za težine.
- **`metric`**: odabrana funkcija rastojanja. Podrazumevana metrika je `minkowski`, koja ima slobodan parametar  $p$ , od kog zavisi selekcija određene funkcije rastojanja iz ove klase. Podrazumevana vrednost za  $p$  je 2, što znači da je podrazumevana metrika euklidsko rastojanje. U

zavisnosti od tipa obeležja (realna numerička, diskretna numerička, binarna, itd), preporučuju se različite metrike (detalji se mogu naći u klasi `neighbors.DistanceMetric`). Ukoliko su vrednosti obeležja realni brojevi, najčešći izbor je euklidsko rastojanje, ali se mogu koristiti i druge  $p$ -norme. U slučaju da su obeležja numerička ali imaju diskretne vrednosti, česta je upotreba Hemingovog rastojanja, a za binarna obeležja prevladava upotreba Žakarovog ili Dajsovog rastojanja. Ako izbor metrike zahteva dodatne parametre, oni se podešavaju u parametrima klase pod nazivom `p` i `metric_params`.

Kao metoda kasnog učenja,  $k$ NN algoritam nema fazu obuke modela, već koristi originalne uzorke za obuku pri klasifikaciji svakog novog uzorka. Međutim, kao i drugi algoritmi mašinskog učenja u okviru `Scikit-learn` biblioteke, primena ovog algoritma prati isti tok: inicijalizacija (metoda `KNeighborsClassifier/KNeighborsRegressor`), obuka (metoda `fit`) i primena obučenog modela (metoda `predict`). Pri inicijalizaciji algoritma postavljaju se vrednosti parametara, dok u fazi obuke, s obzirom da ne postoji obuka modela, algoritam može da formira strukture uzoraka za obuku koje omogućuju da algoritam klasifikacije neobeleženog uzorka radi brže. Primer obuke i primene  $k$ NN klasifikatora na test skup:

```
classifier = KNeighborsClassifier(n_neighbors=3,
                                   metric='euclidean')
classifier.fit(x_train, y_train)
y_predicted = classifier.predict(x_test)
```

## 9.1. Zadaci

### Zadatak 1

Implementirati algoritam za klasifikaciju ispitanika na one koji se izjašnjavaju kao srećni i one koji se ne izjašnjavaju kao srećni. Koristiti bazu podataka<sup>6</sup> koja sadrži rezultate ankete od 143 osobe. U anketi se postavlja 6 pitanja u kojima se proverava koliko je anketirana osoba na skali od 1 do 5 zadovoljna sa: dostupnošću informacija o gradskim službama, visinom troškova stanovanja, kvalitetom državnog školstva, poverenjem u lokalnu policiju, održavanjem ulica i trotoara, i dostupnošću društvenih događaja. Odgovori anketirane osobe su

<sup>6</sup>Podaci su dostupni na: [archive.ics.uci.edu/ml/datasets/Somerville+Happiness+Survey](http://archive.ics.uci.edu/ml/datasets/Somerville+Happiness+Survey)

## 9 KLASIFIKATOR METODOM $K$ NAJBLIŽIH SUSEDА

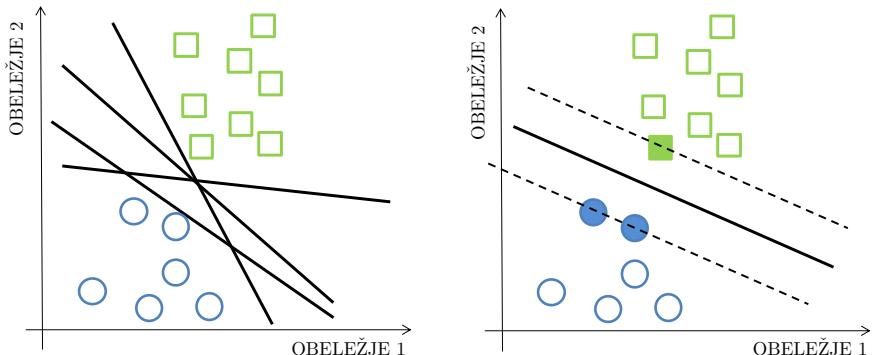
---

obeležja za tu osobu i svaka osoba rekla je da li je srećna ili nije, a taj odgovor predstavlja klasnu labelu.

1. Učitati podatke u `DataFrame`. Koliko ima uzoraka? Koliko ima obeležja i kog su tipa?
2. Proveriti koliko uzoraka ima u kojoj klasi, kao i da li ima nedostajućih vrednosti.
3. Realizovati funkciju koja računa različite mere uspešnosti klasifikatora na osnovu date matrice konfuzije.
4. Podeliti uzorke na pet podskupova i izvršiti unakrsnu validaciju.
5. U svakoj iteraciji unakrsne validacije inicijalizovati, obučiti i testirati model. Koje parametre je moguće varirati? Kako odabrat odgovarajuću metriku?
6. Da li je bitno normalizovati obeležja kod  $k$ NN algoritma? Ima li u ovom konkretnom problemu smisla normalizovati obeležja?
7. Za dve različite metrike, prikazati grafik zavisnosti prosečne tačnosti u 5 iteracija od odabranog broja suseda, a broj suseda varirati od 1 do 10.
8. Za odabране optimalne parametre, konačnu matricu konfuzije izračunati akumuliranjem matrica konfuzije u svakoj iteraciji. Koliko uzoraka je pogrešno klasifikovano? Kakve su mere uspešnosti?
9. Ponoviti računanje mera za klasifikator sa optimalnim parametrima koristeći unakrsnu validaciju sa jednim izdvojenim uzorkom. Da li se rezultati razlikuju? Zašto?

## 10. Mašina na bazi vektora nosača

Klasifikator pod nazivom „mašina na bazi vektora nosača“ (engl. *Support Vector Machine – SVM*) prvenstveno se koristi za rešavanje problema binarne klasifikacije tako što identificuje hipervrš koja treba da razdvoji uzorke u prostoru obeležja tako da između oblasti popunjene uzorcima različitih klasa postoji što širi prostor. Mašina na bazi vektora nosača zasniva se na *klasifikatoru maksimalne marge*, čiji je cilj da odredi hiperravan koja će na optimalan način podeliti prostor na dva dela tako da se u jednom delu nađu uzorci iz jedne klase, a u drugom delu uzorci iz druge. Da bi ovo bilo moguće, klase moraju biti linearne separabilne, a da bi hiperravan razdvajanja bila optimalna, ona mora obezbediti najveću moguću marginu, odnosno, najveće rastojanje od hiperravnog do najbližeg uzorka iz obe klase (Slika 10.1). Ovako definisana hiperravan omogućava bolju generalizaciju, što znači da se očekuje tačnija klasifikacija novih uzoraka u odnosu na bilo koju drugu hiperravan koja bi takođe razdvojila uzorke iz skupa za obuku.



Slika 10.1: Nekoliko mogućih hiperravnih razdvajanja (levo) i optimalna hiperravan razdvajanja (desno).

Ako je  $\mathbf{x}$  proizvoljna tačka u prostoru obeležja, hiperravan se može definisati jednakošću:

$$\theta_0 + \boldsymbol{\theta}^\top \mathbf{x} = 0, \quad (10.1)$$

gde  $\boldsymbol{\theta}$  predstavlja težinske koeficijente uz obeležja, dok je  $\theta_0$  slobodan član i ako je on jednak 0, to znači da hiperravan prolazi kroz koordinatni početak. Vektor  $\boldsymbol{\theta}$  normalan je na tako definisanu hiperravan. Hiperravan razdvajanja biće bilo

koja hiperravan za koju važe relacije:

$$\begin{aligned}\theta_0 + \boldsymbol{\theta}^\top \mathbf{x} &< 0, \quad \mathbf{x} \in K_1, \\ \theta_0 + \boldsymbol{\theta}^\top \mathbf{x} &> 0, \quad \mathbf{x} \in K_2,\end{aligned}\tag{10.2}$$

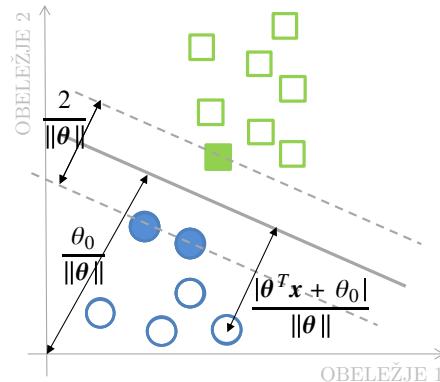
gde su  $K_1$  i  $K_2$  klase. Pošto postoji beskonačno mnogo rešenja za takvu hiperravan (ukoliko ona postoji), biraju se one vrednosti parametara  $\boldsymbol{\theta}$  i  $\theta_0$  za koje važi da je rastojanje uzoraka koji su najbliži hiperravnim odlučivanju  $|\theta_0 + \boldsymbol{\theta}^\top \mathbf{x}| = 1$  (kanonička hiperravan). Uzimajući u obzir i formulu za rastojanje tačke od hiperravni, dobija se izraz za širinu marge naznačen na Slici 10.2. S obzirom da je cilj da se marga razdvajanja maksimizuje, klasifikator maksimalne marge se dobija minimizacijom ciljne funkcije  $J(\boldsymbol{\theta})$ , definisane na osnovu  $L_2$  norme:

$$J(\boldsymbol{\theta}) = \frac{1}{2} \|\boldsymbol{\theta}\|^2,\tag{10.3}$$

uz uslov da važi:

$$y^{(n)}(\theta_0 + \boldsymbol{\theta}^\top \mathbf{x}^{(n)}) \geq 1, \quad \forall n = 1, 2, \dots, N,\tag{10.4}$$

gde je  $N$  broj uzoraka, a  $y^{(n)}$  iznosi 1 ili  $-1$  u zavisnosti od klase kojoj uzorak  $\mathbf{x}^{(n)}$  pripada. Kako je ciljna funkcija kvadratna, ona poseduje jedinstven globalni minimum, odnosno optimalna hiperravan biće jedinstvena. Uzorci koji



**Slika 10.2:** Optimalna hiperravan razdvajanja sa izrazima za rastojanje tačke od prave i širinu marge.

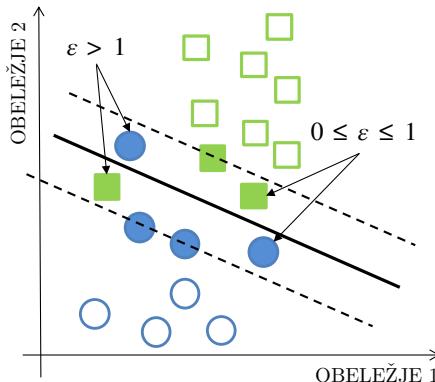
leže na ivicama margine, odnosno uzorci koji su najbliži hiperravnim razdvajanja, nazivaju se *vektori nosači*. Oni su jedini koji utiču na tačan položaj optimalne hiperravni, dok ostali uzorci za obuku nemaju uticaj na nju. Ovako dobijen optimizacioni problem rešava se tehnikom Lagranžovih multiplikatora.

Međutim, kod brojnih klasifikacionih problema uzorci nisu linearno razdvojivi. Tada se klasifikator maksimalne margine ne može upotrebiti jer ne postoji nijedna hiperravan koja ispunjava zadate uslove. Jedno od rešenja je da se dozvoli da pojedini uzorci za obuku budu i sa pogrešne strane bliže ivice margine, pa i sa pogrešne strane hiperravni razdvajanja, čime se dobija tzv. *klasifikator meke margine*. Položaj uzorka u odnosu na ivice margine i hiperravan razdvajanja definiše se uvođenjem tolerancije na grešku po uzorku  $\varepsilon^{(n)}$ , čija vrednost je 0 za uzorce izvan margine, između 0 i 1 ako je uzorak u prostoru margine ali sa ispravne strane hiperravni razdvajanja, odnosno preko 1 ako je uzorak sa pogrešne strane hiperravni (Slika 10.3). U tom slučaju je ciljna funkcija koja se minimizuje data izrazom:

$$J(\boldsymbol{\theta}) = \frac{1}{2} \|\boldsymbol{\theta}\|^2 + C \sum_{n=1}^N \varepsilon^{(n)} \quad (10.5)$$

uz uslov:

$$y^{(n)}(\boldsymbol{\theta}_0 + \boldsymbol{\theta}^\top \mathbf{x}^{(n)}) \geq 1 - \varepsilon^{(n)} \wedge \varepsilon^{(n)} \geq 0, \quad \forall n = 1, 2, \dots, N. \quad (10.6)$$



**Slika 10.3:** Optimalna hiperravan razdvajanja sa tzv. mekom marginom. Uzorci predstavljeni punim oblicima su vektori nosači.

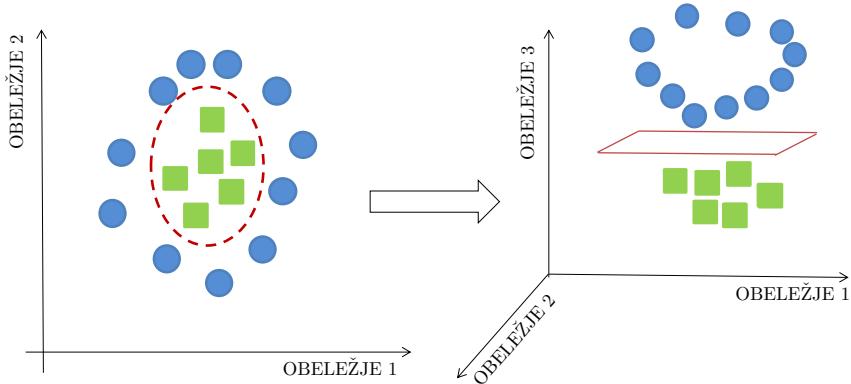
Parametar  $C$  predstavlja regularizacioni parametar i određuje ukupnu toleranciju na grešku klasifikacije. Mala vrednost  $C$  u izrazu za ciljnu funkciju omogućuje porast sume tolerancije na grešku, jer ona, za malo  $C$  ne utiče značajno na funkciju cilja. U tom slučaju se može očekivati šira margina i posledično više uzoraka unutar margine, a neki i sa pogrešne strane hiperravnim (odnosno pogrešno klasifikovani). Veće vrednosti  $C$  podrazumevaju veći doprinos sume tolerancije na grešku po uzorcima u funkciji cilja i stoga se optimalno rešenje pronalazi tako da ova suma bude što manja, odnosno, da se što manji broj uzoraka nalazi sa pogrešne strane hiperravnim i unutar margine. Povećanjem parametra  $C$  očekuje se, dakle, rešenje sa užom marginom, ali treba imati u vidu da striktnija podela uzoraka za obuku može da dovede do smanjene sposobnosti obučenog modela da generalizuje. Vektori nosači u ovom slučaju su, osim onih na ivicama margine, i oni koji se nalaze unutar nje. Ovaj optimizacioni problem nije ništa kompleksniji od prethodnog i takođe se rešava tehnikom Lagranžovih multiplikatora.

Postoje problemi za koje linearne granice nisu odgovarajuće čak ni uz korišćenje meke margine i promenljivog parametra  $C$ . Tada se primenjuje tzv. *kernel trik*: moguće je preslikati uzorce u višedimenzionalni prostor obeležja u kojem su uzorci linearno razdvojivi, čemu bi u originalnom prostoru obeležja odgovarala nelinearna granica odlučivanja (Slika 10.4). Ovako dobijen klasifikator naziva se *mašinom na bazi vektora nosača* (SVM). Problem se rešava u dva koraka, odnosno, uzorci se prvo nelinearno preslikavaju u višedimenzionalni prostor, da bi se zatim u tom prostoru pronašla optimalna hiperravan razdvajanja. Linearni klasifikator na bazi vektora nosača koji čine skup  $S$  može se definisati izrazom:

$$g(\mathbf{x}) = \theta_0 + \boldsymbol{\theta}^\top \mathbf{x} = \theta_0 + \sum_{n \in S} \alpha^{(n)} y^{(n)} (\mathbf{x}^{(n)})^\top \mathbf{x}, \quad (10.7)$$

do kog se dolazi korišćenjem tehnike Lagranžovih multiplikatora  $\alpha^{(n)}$  u minimizaciji funkcije cilja i izražavanjem parametara hiperravnim preko njih. Potrebno je uočiti da u ovom izrazu uzorci učestvuju samo u okviru skalarnog proizvoda sa  $\mathbf{x}$ . Izraz *kernel trick* iskazuje mogućnost za računanje skalarnih proizvoda uzoraka u višedimenzionalnom prostoru obeležja bez potrebe za pronalaženjem pogodne transformacije za prelazak u višedimenzionalni prostor. Skalarni proizvod se zamenjuje odgovarajućom kernel funkcijom  $F$ , pa mašina na bazi vektora nosača može da se formuliše na sličan način:

$$g(\mathbf{x}) = \theta_0 + \sum_{n \in S} \alpha^{(n)} y^{(n)} F(\mathbf{x}, \mathbf{x}^{(n)}). \quad (10.8)$$



**Slika 10.4:** Primer upotrebe SVM sa kernel trikom: problem koji nije linearno separabilan u originalnom prostoru obeležja (levo) postaje linearno separabilan nakon transformacije pomoću kernel funkcije (desno).

Neke od najčešće korišćenih kernel funkcija su:

- radikalni kernel:

$$F(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = e^{-\frac{1}{2\sigma^2} \sum_{k=1}^D (x_k^{(i)} - x_k^{(j)})^2}, \quad (10.9)$$

- polinomijalni kernel stepena  $m$ :

$$F(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \left( 1 + \sum_{k=1}^D x_k^{(i)} x_k^{(j)} \right)^m. \quad (10.10)$$

Kako je kod SVM samo skalarni proizvod zamenjen kernelom, optimizacioni problem i način njegovog rešavanja ostaju isti. Pozitivna strana rešavanja optimizacionog problema Lagranžovim multiplikatorima jeste da se može rešavati kao tzv. dualni problem, koji je srazmeran količini podataka za obuku, ili kao Lagranžov osnovni problem, koji je srazmeran dimenzionalnosti prostora.

Iako je SVM prvenstveno namenjen binarnoj klasifikaciji, primenjuje se i za razdvajanje  $K$  klasi, gde je  $K > 2$ . Moguća su dva pristupa: svaki protiv svakog (engl *one vs. one*) i jedan protiv svih (*one vs. all* ili *one vs. rest*). U prvom slučaju uzorak se dodeljuje onoj klasi koja je najveći broj puta bila

uspešnija u nadmetanju klasifikatora po parovima. U drugom pristupu, konstruiše se  $K$  klasifikatora, a uzorak se dodeljuje klasi za koju  $g(\mathbf{x})$  ima najveću vrednost.

SVM metoda može da se koristi i za rešavanje regresionih problema (engl. *support vector regression* – SVR). U ovom slučaju teži se da predviđene vrednosti što manje odstupaju od ciljnih vrednosti, pri čemu se podrazumeva da je greška predikcije nula ukoliko je razlika predviđene i ciljne vrednosti po apsolutnoj vrednosti manja od praga tolerancije odstupanja  $\varepsilon$ . Vektori nosači su u ovom slučaju oni uzorci za koje je apsolutno odstupanje predviđene vrednosti od ciljne jednako ili veće od  $\varepsilon$ . Na ovaj način se pomoću parametra  $\varepsilon$  definiše širina margine. Prilikom pronašlaska rešenja minimizuje se funkcija cilja:

$$J(\boldsymbol{\theta}) = \frac{1}{2} \|\boldsymbol{\theta}\|^2 + C \sum_{n=1}^N E_\varepsilon(h(\mathbf{x}^{(n)}) - y^{(n)}), \quad (10.11)$$

gde je  $E_\varepsilon$  funkcija greške neosetljiva na ostupanja manja od  $\varepsilon$ ,  $C$  (inverzni) regulizacioni parametar, dok je  $\boldsymbol{\theta}$  vektor koji definiše traženu funkciju predviđanja  $h(\mathbf{x})$ . Parametar  $C$  množi grešku odstupanja, što znači da veće vrednosti ovog parametra u ciljnoj funkciji imaju za posledicu pronalaženje rešenja sa što manje tačaka van  $\varepsilon$ -okoline ciljnih vrednosti, samim tim može da uslovi natprilagođenje. Druga opcija da se zada SVR jeste preko parametra  $\nu$  koji određuje dozvoljen broj uzoraka za koje predviđene vrednosti nisu u  $\varepsilon$ -okolini ciljnih vrednosti. I u ovom slučaju se parametri  $\nu$  i  $C$  određuju postupkom unakrsne validacije. SVR slično klasifikacionom slučaju može da koristi linearan kernel ili neke od nelinearnih kernela.

Prednosti SVM metode ogledaju se u kontroli složenosti, memorijskoj efikasnosti, stabilnosti i ponovljivosti rezultata. Složenost ne zavisi od dimenzionalnosti i dobri rezultati mogu se postići čak i ako je broj obeležja veći od broja uzoraka, a memorijska efikasnost ogleda se u činjenici da položaj hiper-ravnih, odnosno parametri modela, zavise samo od vektora nosača.

U **Scikit-learn** biblioteci postoje sledeće implementirane klase za klasifikator SVM: **SVC**, **NuSVC** i **LinearSVC**. Klase **SVC** i **NuSVC** koriste različit skup ulaznih parametara i malo su drugačije matematički definisane. Klasa **SVC** odgovara teorijskom delu ove vežbe. Ovaj algoritam, pored linearног, može da koristi i druge kernele. Ipak, ako je utvrđeno da treba primeniti linearni kernel, savetuje se korišćenje klase **LinearSVC**, koja predstavlja implementaciju SVM sa linearnim kernelom, ali je dodatno optimizovana te brže konvergira. Osim po brzini, ove dve funkcije razlikuju se i po podrazumevanoj funkciji cene i u slučaju razdvajanja više klase koriste različite pristupe. Sve tri funkcije nalaze

se u **svm** modulu biblioteke **Scikit-learn**. Za rešavanje regresionog problema pomoću ovog algoritma koristi se klasa **SVR** ili **NuSVR**, takođe iz **svm** modula.

Parametri klase **SVC** su sledeći:

- **C**: regularizacioni parametar iz ciljne funkcije;
- **kernel**: podrazumevana vrednost je **rbf**, a može se koristiti **linear**, **poly**, **rbf**, **sigmoid** ili **precomputed**, pri čemu je, u zavisnosti od odabranog kernela, potrebno podesiti njegove parametre;
- **degree**: stepen polinomijalnog kernela;
- **gamma**: parametar **rbf** kernela; podrazumevana vrednost je **scale**, a može se koristiti **scale**, **auto** ili konkretna brojna vrednost;
- **coef0**: nezavisan član za polinomijalni i sigmoid kernel; podrazumevana vrednost je 0;
- **class\_weight**: služi za postavljanje težinskih faktora za uzorke različitih klasa; vrednost **balanced** koristi klasne labele da podesi težine obrnuto proporcionalno zastupljenosti klase, a ako se parametar ne podesi, smatra se da su sve težine 1; može se podesiti i tačna težina za svaku od klasa u okviru tzv. rečnika (**dict**);
- **decision\_function\_shape**: podrazumevana vrednost je **ovr**, a može biti **ovo** (*one versus one*) ili **ovr** (*one versus rest*); koristi se u višeklasnim problemima;
- **break\_ties**: uzima se u obzir samo ako se rešava problem sa više od dve klasе u **ovr** pristupu; ako je vrednost **True**, odluka će se doneti na osnovu toga koliko je klasifikator siguran u svoju odluku, a ako je **False**, uzorak će se dodeliti prvoj navedenoj klasi (podrazumevana vrednost je **False**).

Primer obuke i primene SVM klasifikatora na test skupu:

```
classifier = SVC(C=1, kernel='rbf',
                  decision_function_shape='ovo')
classifier.fit(X_train, y_train)
y_predicted = classifier.predict(X_test)
```

Parametri klase **LinearSVC** su sledeći:

- **penalty**: podrazumevana vrednost je **l2**, a može se koristiti i **l1**, pri čemu se u implementaciji **SVC** koristi **l2**;

- **loss**: ciljna funkcija; podrazumevana vrednost je **squared\_hinge**, a može se koristiti i **hinge**, kao u implementaciji SVC;
- **dual**: podrazumevana vrednost je **True**; određuje koji će se optimizacioni problem rešavati, a savetuje se dualni (vrednost parametra **True**) kada je broj obeležja veći od broja uzoraka.
- **C**: regularizacioni parametar; veća vrednost znači striktniju podelu uzoraka za obuku;
- **multi\_class**: podrazumevana vrednost je **ovr**, a umesto nje se može koristiti i **crammer\_singer** (formira združenu funkciju cene za sve klase, ali se u praksi pokazuje da ne radi ništa bolje od **ovr** pristupa);
- **fit\_intercept**: podrazumevana vrednost je **True**; određuje da li će hiper-ravan razdvajanja sadržati slobodan član;
- **class\_weight**: isto kao u klasi SVC.

Primer obuke i primene linearног SVM klasifikatora na test skupu:

```
classifier = LinearSVC(C=1)
classifier.fit(X_train, y_train)
y_predicted = classifier.predict(X_test)
```

## 10.1. Zadaci

### Zadatak 1

Implementirati algoritam za klasifikaciju aktivnosti čoveka na osnovu senzora sa mobilnog telefona. Koristiti odgovarajuću bazu podataka<sup>7</sup>, koja sadrži 561 očitavanje senzora (po 3 koordinate koje definišu poziciju telefona, očitane sa žiroskopa i akcelerometra). Svakom od 30 ispitanika u bazi dodeljen je identifikacioni broj i za svako očitavanje dodeljena je klasna labela o trenutnoj aktivnosti. Ima ukupno 30 ispitanika, od kojih se 70% nalazi u skupu za obuku, a ostali u skupu za testiranje.

1. Učitati podatke za obuku u jedan **DataFrame**, a podatke za testiranje u drugi **DataFrame**. Koliko ima uzoraka? Koliko ima obeležja i kog su tipa?

<sup>7</sup>Podaci su dostupni na: [www.kaggle.com/uciml/human-activity-recognition-with-smartphones](http://www.kaggle.com/uciml/human-activity-recognition-with-smartphones)

2. Proveriti koliko uzoraka ima u kojoj klasi, kao i da li ima nedostajućih vrednosti.
3. Na osnovu uzoraka za obuku, upotrebom unakrsne validacije sa 3 pod-skupa, pronaći optimalne parametre za  $k$ NN i za SVM klasifikator, oslanjajući se na jednu od mera uspešnosti. Ispisati za svaku kombinaciju parametara odabranu mjeru uspešnosti i zbirnu matricu konfuzije. Koja mera je ovde adekvatna? Koji su parametri optimalni? Kojih je dimenzija dobijena matrica konfuzije?
4. Obučiti konačan model  $k$ NN i konačan model SVM klasifikatora, a potom ih primeniti na test skup i evaluirati performanse. S obzirom da je u pitanju višeklasni problem, računati i mikro i makro mere uspešnosti. Koji klasifikator je dao bolje rezultate? Da li je kod nekog od klasifikatora unakrsna validacija predvidela suviše optimistične rezultate? Zašto? Kako izgleda matrica konfuzije dobijena na test skupu? Koje aktivnosti klasifikatori najčešće mešaju? Zašto?
5. Da li je imalo smisla koristiti identifikacioni broj ispitanika kao jedno od obeležja? Kako bi izgledali rezultati ako bi se to obeležje izostavilo?

## 11. Stabla odluke

Stabla odluke predstavljaju jednu od često korišćenih metoda nadgledanog učenja. Može da se koristi za rešavanje i klasifikacionih i regresionih problema i može da radi i sa numeričkim i sa kategoričkim obeležjima. Koren stabla (engl. *root node*) predstavlja čvor koji sadrži skup svih uzoraka, i od njega se stablo grana, odnosno, vrši se sukcesivna particija skupa uzoraka na dva disjunktna podskupa. Particija se formira na osnovu vrednosti obeležja koje se proceni kao najznačajnije u datom trenutku. Na primer, postavlja se pitanje „*Da li je  $x_1 \leq a$ ?*“, gde je  $x_1$  vrednost jednog od numeričkih obeležja, dok je  $a$  neka numerička vrednost; slično, može se pitati „*Da li je  $x_1 = a$ ?*“, gde je  $x_1$  vrednost jednog od kategoričkih obeležja, dok je  $a$  jedna od mogućih kategorija. Prema odgovoru na postavljeno pitanje (da ili ne), odnosno prema vrednosti datog obeležja za svaki pojedinačni uzorak, skup pridružen čvoru deli se na dva podskupa, čime se ujedno formiraju i dva nova čvora. Algoritam pronalazi optimalno pitanje na osnovu kojeg vrši particiju čvora prolaskom kroz skup mogućih pitanja i procenom u kojoj meri će čvorovi formirani na osnovu svakog od tih pitanja biti „čisti“. Čistoća određenog skupa, odnosno čvora, podrazumeva što izraženiju dominaciju jedne od klase u njemu. Kriterijum za zaustavljanje grananja može biti dobijanje potpuno čistog čvora, a može se zasnovati i na ograničenju u pogledu maksimalnog broja uzoraka u krajnjem čvoru ili maksimalne dubine stabla. Čvorovi koji se dalje ne dele nazivaju se *listovi* ili *krajnji čvorovi* (engl. *leaves/terminal nodes*). Često se primenjuje i ideja formiranja maksimalno razgranatih stabala, nakon čega se primenjuje orezivanje listova (engl. *pruning*) prema određenom kriterijumu orezivanja. Ovaj postupak motivisan je činjenicom da se neretko dešava da loša podela jednog čvora bude propraćena izuzetno dobrom podelom nekog od njegovih potomaka. Formiranje stabla odluke odgovara obuci modela, dok se klasifikacija ili regresija vrši tako što se novi uzorak propusti kroz stablo, počev od korena, i utvrdi se u kom listu će završiti. Vrednost izlazne promenljive kod regresionih problema se najčešće predviđa (procenjuje) kao prosek vrednosti uzorka u listu, dok se klasa test uzorka kod klasifikacionih problema najčešće određuje kao klasa koja prevladava među uzorcima u listu.

Kod klasifikacionih problema kriterijum podele može se zasnivati na vrednosti Đinijevog indeksa (engl. *Gini index*), entropije,  $\chi^2$ -statistike, varijanse i dr. Kod regresionih problema kriterijumi podele zasnivaju se na srednjim kvadratnim ili srednjim apsolutnim greškama između stvarnih i predviđenih vrednosti. Dva najčešće korišćena kriterijuma podele kod klasifikacionih prob-

lema su Đinijev indeks i entropija. Đinijev indeks predstavlja meru ukupne varijanse kroz  $K$  klase, i za čvor  $m$  definisan je izrazom:

$$G_m = \sum_{k=1}^K p_{mk}(1 - p_{mk}), \quad (11.1)$$

gde je  $p_{mk}$  udeo uzoraka klase  $K_k$  iz skupa za obuku u čvoru  $m$ , a  $K$  je ukupan broj klasa. Entropija čvora  $m$  definiše se izrazom:

$$H_m = - \sum_{k=1}^K p_{mk} \log p_{mk}. \quad (11.2)$$

Ispostavlja se da su Đinijev indeks i entropija numerički vrlo bliski te se krajnja stabla ne razlikuju mnogo bez obzira koji od ova dva kriterijuma podele se koristi. Po formuli se vidi da će vrednost Đinijevog indeksa, kao i entropije, biti mala ako je  $p_{mk}$  blisko 0 ili 1, što bi značilo da u čvoru  $m$  uzoraka iz klase  $K_k$  skoro da nema, ili da su u čvoru  $m$  skoro svi uzorci iz klase  $K_k$ , što i jeste odlika čistog skupa. Pri odabiru pitanja kriterijum podele predstavljaće zbir Đinijevih indeksa (ili entropija) za oba čvora-potomka (dobijena podskupa) koja bi odabirom tog pitanja nastala:

$$G = \frac{n_{lev}}{N} G_{lev} + \frac{n_{des}}{N} G_{des}, \quad (11.3)$$

gde je  $G$  ukupna vrednost odabranog kriterijuma podele,  $n_{lev}$  i  $n_{des}$  označavaju broj uzoraka u levom i desnom čvoru-potomku,  $N$  predstavlja broj uzoraka u čvoru koji se deli, a  $G_{lev}$  i  $G_{des}$  su vrednosti odabranog kriterijuma podele za levi i desni čvor-potomak.

Na sledećem primeru ilustrovan je odabir optimalnog pitanja na osnovu Đinijevog indeksa. Neka je cilj klasifikacija studenata na one koji treniraju fudbal i one koji ne treniraju fudbal. Postoji 20 studenata u skupu za obuku stabla, od kojih 10 trenira fudbal, a 10 ga ne trenira, pri čemu su za svakog studenta poznati pol i visina. Razmatra se da li je u datom trenutku bolje postaviti pitanje „*Kog je pola student?*“ ili „*Da li je student viši od 180 cm?*“. Ukoliko se postavi pitanje u vezi sa polom, dobijaju se 2 podskupa, pri čemu u jednom ima 7 devojaka od kojih 14% trenira fudbal, dok u drugom ima 13 momaka od kojih 69% trenira fudbal. S druge strane, podelom prema visini studenata dobija se podskup od 9 studenata nižih od 180 cm od kojih 55% trenira fudbal, kao i podskup od 11 studenata viših od 180 cm od kojih 45% trenira

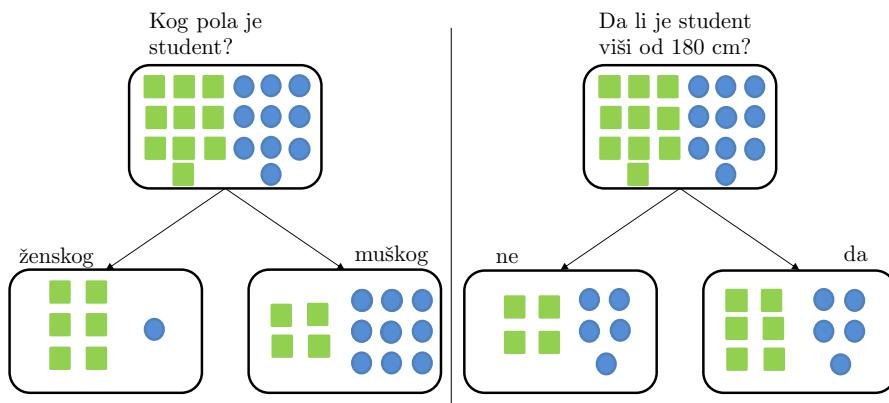
fudbal (Slika 11.1). Računanjem Đinijevog indeksa za podelu prema polu, dobija se

$$\frac{7}{20}(0,14 \cdot 0,86 + 0,86 \cdot 0,14) + \frac{13}{20}(0,69 \cdot 0,31 + 0,31 \cdot 0,69) = 0,36$$

dok je za podelu prema visini studenta ukupan Đinijev indeks:

$$\frac{9}{20}(0,55 \cdot 0,45 + 0,44 \cdot 0,46) + \frac{11}{20}(0,45 \cdot 0,55 + 0,54 \cdot 0,46) = 0,47$$

Sudeći po vrednosti Đinijevog indeksa, bolji izbor je pitanje u vezi sa polom jer je dobijena vrednost manja. Ovo se jasno vidi i sa Slike 11.1 jer su podskupovi dobijeni podelom prema polu čistiji, odnosno u dobijenim podskupovima jedna od klasa jasno dominira.



**Slika 11.1:** Odabir boljeg pitanja prema Đinijevom indeksu s ciljem podele studenata u dve grupe, oni koji igraju fudbal (kružići) i oni koji ne igraju fudbal (kvadratići).

Prednosti ovog algoritma ogledaju se u mogućnosti jednostavne interpretacije i vizualizacije, kao i u relativno maloj osetljivosti na pitanja koja nisu od značaja, odnosno, ne doprinose tačnosti klasifikacije. Algoritam je pogodan za utvrđivanje najvažnijih obeležja i otkrivanje odnosa među njima. Međutim, stabla odluke imaju i niz mana. Naime, prilikom obuke vrlo lako dolazi do natprilagođenja (engl. *overfitting*), a uz to male varijacije u podacima mogu dovesti do formiranja značajno drugačijeg stabla. Pored toga, stabla odluke vrše i fragmentaciju podataka, odnosno, kad je odgovor na neko pitanje negativan, postaju irelevantni svi uzorci za koje je odgovor bio pozitivan, i čak i

ako među nekim drugim obeležjima ovih uzoraka postoje isti odnosi, ta znanja se ne koriste. Generalno, stablo odluke nije čest izbor jer postoje znatno bolji algoritmi i za klasifikaciju i za regresiju, ali se istovremenim korišćenjem većeg broja stabala odluke, odnosno, formiranjem ansambla stabala odluke, njihovi nedostaci mogu u znatnoj meri prevazići.

U **Scikit-learn** biblioteci u vezi sa stablima odluke postoje implementirane klase `DecisionTreeClassifier` i `DecisionTreeRegressor`, koje se nalaze u modulu `tree` (trenutna implementacija ne dozvoljava eksplicitnu upotrebu kategoričkih obeležja). Prva je namenjena za klasifikacione probleme, druga za regresione, a parametri su im isti.

Parametri klase `DecisionTreeClassifier` i `DecisionTreeRegressor` su sledeći:

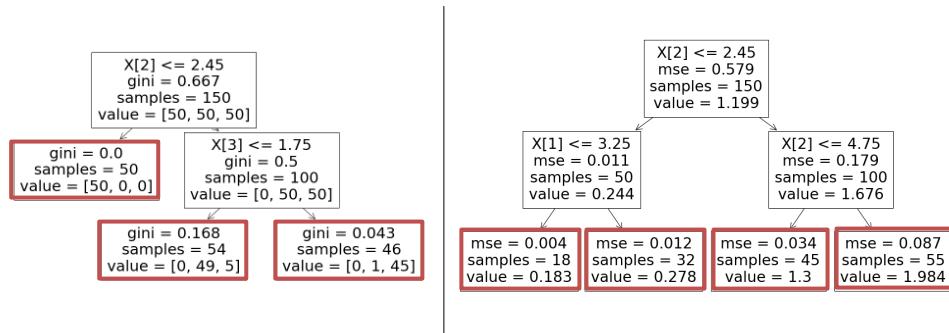
- **criterion:** kriterijum podele; podrazumevana vrednost kod klasifikacionog stabla je `gini`, a može se odabrat i `entropy`; kod regresionog stabla podrazumevana vrednost je `mse` i predstavlja srednju kvadratnu grešku, a moguće je odabrat i `friedman_mse` ili `mae` – srednju apsolutnu grešku.
- **max\_depth:** ako se ovaj parametar ne podesi, stablo se deli dok listovi ne postanu potpuno čisti ili dok ne sadrže `min_samples_split` uzoraka.
- **min\_samples\_split:** podrazumevana vrednost je 2; predstavlja minimalan broj uzoraka u čvoru da bi bilo dozvoljeno njegovo deljenje; ako je vrednost celobrojna, predstavljaće broj uzoraka, a ako se specificira necelobrojna vrednost između 0 i 1, ona će predstavljati udeo broja uzoraka.
- **min\_samples\_leaf:** podrazumevana vrednost je 1; predstavlja minimalan broj uzoraka koji list mora sadržati;
- **max\_features:** predstavlja broj obeležja koja će stablo razmatrati pri izboru najboljeg pitanja za podelu čvora; vrednost parametra može biti `None`, `log2`, `sqrt`, `auto` ili konkretna numerička vrednost;
- **max\_leaf\_node:** predstavlja najveći dozvoljen broj listova;
- **min\_impurity\_decrease:** podela će biti izvršena ukoliko smanjuje nečistoću skupa bar za vrednost postavljenu ovim parametrom;
- **class\_weight:** služi za postavljanje težinskih faktora za uzorce različitih klasa; vrednost `balanced` koristi klasne labele da podesi težine obrnuto

proporcionalno zastupljenosti klase, a ako se parametar ne podesi, smatra se da su sve težine 1; može se podesiti i tačna težina za svaku od klasa u okviru tzv. rečnika (`dict`).

Vizuelizacija stabla može biti veoma korisna, i u okviru `Scikit-learn` biblioteke ostvaruje se pozivom `tree.plot_tree(trained_tree)`. U svakom čvoru na generisanoj slici piše koje pitanje se postavlja za podelu, kolika je vrednost odabranog kriterijuma podele, koliko ima uzoraka u datom čvoru i koja je raspodela tih uzoraka po klasama u slučaju klasifikacionog stabla (Slika 11.2 levo) ili kolika je prosečna vrednost ciljnog obeležja kod tih uzoraka u slučaju regresionog stabla (Slika 11.2 desno).

Primer obuke i testiranja klasifikatora na bazi stabla odluke:

```
classifier = DecisionTreeClassifier(max_depth=5,
                                     max_features=2)
classifier.fit(X_train, y_train)
y_predicted = classifier.predict(X_test)
```



Slika 11.2: Primer klasifikacionog stabla (levo) i regresionog stabla (desno).

## 11.1. Metoda slučajne šume

Ansambalsko učenje podrazumeva obuku više jednostavnih modela sa ciljem unapređenja performansi pri rešavanju kompleksnijih problema. Jedna od metoda ansambalskog učenja koja koristi stabla odluke kao jednostavne klasifikatore ili regresore naziva se metoda slučajne šume (engl. *random forest*).

Metoda slučajne šume se pokazuje kao veoma uspešan algoritam i za klasifikacione i za regresione probleme ako je skup podataka za obuku dovoljno velik. Ideja je da se obuci mnoštvo stabala odluke, a donošenje krajnje odluke o klasi ili vrednosti nepoznatog uzorka vrši se glasanjem u slučaju klasifikacije, odnosno usrednjavanjem dobijenih rezultata u slučaju regresije. Na početku se, primenom *bootstrap* metode ponovnog uzorkovanja iz skupa za obuku, formira  $M$  novih skupova. *Bootstrap* metoda ponovnog uzorkovanja podrazumeva kreiranje novih skupova za obuku nasumičnim izvlačenjem uzoraka sa vraćanjem iz originalnog skupa. Kada je svaki od novodobijenih skupova za obuku iste veličine kao originalni skup, on tipično sadrži oko  $\frac{2}{3}$  jedinstvenih uzoraka. Svako od stabala odluke u okviru metode slučajne šume zatim se obučava, odnosno, formira na osnovu jednog od  $M$  novodobijenih skupova za obuku. Pored toga, uvodi se i ograničenje da svako od stabala u trenutku donošenja odluke o najboljem pitanju za podelu čvora sme da razmatra samo  $p$  nasumično odabranih obeležja od mogućih  $D$  obeležja. Ovo ograničenje uvodi se radi dekorelacije stabala, pri čemu  $p$  predstavlja parametar algoritma koji se obično bira po pravilu  $p \approx \sqrt{D}$ . Kod ovog algoritma greška se procenjuje na osnovu tačnosti predviđanja za uzorke koji u postupku ponovnog uzorkovanja *bootstrap* metodom nisu nijednom izvučeni iz polaznog skupa. Naime, kako je svako od  $M$  stabala obučeno nad jednim od *bootstrap* skupova, odnosno, na približno  $\frac{2}{3}$  uzoraka iz skupa za obuku, to znači da je za testiranje svakog stabla preostalo oko  $\frac{1}{3}$  ukupnog broja uzoraka (engl. *out-of-bag* uzorci). Prema tome, za svaki uzorak iz skupa za obuku greška se može proceniti na osnovu predikcije stabala za koja je taj uzorak predstavljao *out-of-bag* uzorak. Procena greške dobijena na ovaj način naziva se *out-of-bag* greškom. Najveći problem ovog algoritma je nemogućnost jednostavne interpretacije.

Analogno funkcijama za klasifikaciono i regresiono stablo, u biblioteci **Scikit-learn** postoje i implementacije za algoritam slučajne šume za klasifikaciju – `RandomForestClassifier` i za regresiju – `RandomForestRegressor`, i nalaze se u modulu `ensemble`. Ove klase pored parametara koji se javljaju i kod obuke pojedinačnog stabla odluke, sadrže i sledeće parametre:

- `n_estimators`: predstavlja broj stabala koji će se obučiti, a podrazumevana vrednost je 100;
- `bootstrap`: podrazumevana vrednost je `True`; ako je vrednost `False` onda se ceo skup podataka za obuku koristi za obuku svakog od pojedinačnih stabala;
- `oob_score`: podrazumevana vrednost je `False`; estimira uspešnost gener-

alizacije modela na osnovu *out-of-bag* greške;

- **max\_samples:** podrazumevana vrednost je `None`, a može biti i konkretna numerička vrednost; ako je parametar `bootstrap` podešen na `True`, ovim parametrom se određuje koliko će se uzoraka koristiti pri obuci svakog pojedinačnog stabla (vrednost `None` podrazumeva sve uzorke, celobrojna vrednost predstavlja broj uzoraka, a necelobrojna vrednost između 0 i 1 predstavlja udeo u ukupnom broju uzoraka iz skupa za obuku).

Primer odabira broja stabala kod primene metode slučajne šume na osnovu *out-of-bag* greške:

```
error_rate = []
num_estimators = range(10, 100) #isprobati 10 do 100 stabala
for i in num_estimators:
    classifier = RandomForestClassifier(n_estimators=i,
   max_depth=5, max_samples=100, oob_score=True)
    classifier.fit(X, Y)
    oob_error = 1 - classifier.oob_score_
    error_rate.append(oob_error)
tmp = error_rate.index(min(error_rate))
fin_num_estimators = num_estimators[tmp]
```

## 11.2. Zadaci

### Zadatak 1

Implementirati algoritam za klasifikaciju ispitanika na one koji zarađuju više i one koji zarađuju manje od 50 hiljada dolara godišnje. Koristiti bazu podataka<sup>8</sup> koja sadrži podatke o preko 30 hiljada ispitanika, a koji se tiču njihovog obrazovanja, zanimanja, bračnog statusa i dr.

1. Učitati podatke u `DataFrame`. Koliko ima uzoraka? Koliko ima obeležja i kog su tipa?
2. Proveriti koliko uzoraka ima u kojoj klasi, kao i da li ima nedostajućih vrednosti.
3. Na proizvoljan način pretvoriti vrednosti kategoričkih obeležja u numeričke vrednosti zbog implementacije algoritma.

---

<sup>8</sup>Podaci su dostupni na: [archive.ics.uci.edu/ml/datasets/Adult](http://archive.ics.uci.edu/ml/datasets/Adult)

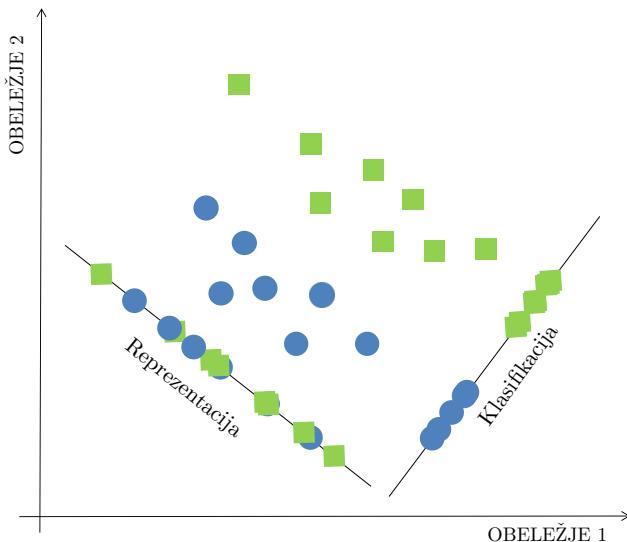
4. Nasumično izdvojiti 15% podataka koji će služiti za testiranje krajnjih klasifikatora. Ostatak podataka koristiti za obuku i odabir optimalnih parametara.
5. Koristeći implemetaciju za stablo odluke, izvršiti unakrsnu validaciju sa 5 podskupova. Koje parametre je moguće menjati?
6. Prikazati obučena stabla. Prikazati konačnu matricu konfuzije i izračunati tačnost klasifikatora.
7. Da li je u ovom slučaju bitna normalizacija obeležja? Koja obeležja su se pokazala veoma bitnim? Da li je došlo do preobučavanja? Šta je potrebno uraditi kako bi se sprečilo preobučavanje?
8. Automatizaciju unakrsne validacije uz pretragu optimalne kombinacije parametara moguće je izvršiti korišćenjem funkcije `gridsearchCV`.
9. Obučiti stablo sa konačno odabranim parametrima i prikazati ga. Kolika je tačnost klasifikatora pri unakrsnoj validaciji sa optimalnim parametrima? Da li se rezultat poboljšao u odnosu na prethodno dobijeni?
10. Testirati klasifikator na izdvojenom test skupu koristeći optimalne parametre. Prikazati konačnu matricu konfuzije i izračunati tačnost klasifikatora. Kakav je rezultat u odnosu na onaj dobijen unakrsnom validacijom?
11. Izvršiti unakrsnu validaciju i odabir parametara za algoritam slučajne šume. Koje parametre je moguće menjati?
12. Prikazati grafik zavisnosti *out-of-bag* greške klasifikacije od broja korišćenih stabala.
13. Obučiti klasifikator sa optimalnim parametrima, testirati ga na izdvojenom test skupu i prikazati matricu konfuzije. Uporediti dobijeni rezultat sa najboljim stablom odluke. Koji algoritam postiže veću tačnost klasifikacije?

## 12. Metode za smanjenje dimenzionalnosti: PCA i LDA

Najveći problem svih algoritama mašinskog učenja jeste ograničen broj uzoraka za obuku. Problemi su često visokodimenzionalni, te prostor gotovo nikada nije dovoljno popunjen uzorcima za obuku. Za postojeći broj uzoraka za obuku postoji maksimalan broj obeležja koji, kada se prekorači, rezultuje opadanjem performansi algoritma mašinskog učenja. Smanjenje dimenzionalnosti prostora je zato veoma korisno, a uglavnom dovodi i do ubrzanja algoritma i pojednostavljenja modela. Dva su osnovna pristupa smanjenju dimenzionalnosti: odabir (selekција) obeležja i izdvajanje (redukcija) obeležja. Selekcija podrazumeva biranje bitnijih obeležja iz skupa svih raspoloživih obeležja, dok redukcija podrazumeva formiranje manjeg skupa novih obeležja kombinovanjem postojećih. Rešavanje problema redukcije obeležja svodi se na pronalaženje preslikavanja  $\mathbf{y} = f(\mathbf{x}) : \mathbb{R}^D \rightarrow \mathbb{R}^M$  gde je  $M < D$ , tako da transformisani vektor obeležja  $\mathbf{y}$  u najvećoj meri očuva informacije iz originalnog prostora obeležja. Izbor preslikavanja određen je funkcijom cilja, prema čemu tehnikе za redukciju dimenzionalnosti delimo u dve grupe: tehnikе reprezentacije i tehnikе klasifikacije (Slika 12.1). Cilj tehnikе reprezentacije jestе da preslikavanjem predstave skup uzoraka što vernije u prostoru sa manjim brojem dimenzija, dok je cilj tehnikе klasifikacije da preslikavanjem omoguće što bolje razdvajanje klasa u prostoru sa manjim brojem dimenzija. Među metodama linearne redukcije obeležja najčešće se koriste tehnikе razlaganja na glavne komponente (engl. *Principal Component Analysis* – PCA), koja koristi kriterijum reprezentacije, kao i tehnikе linearne diskriminantne analize (engl. *Linear Discriminant Analysis* – LDA), koja koristi kriterijum klasifikacije.

### 12.1. Razlaganje na glavne komponente - PCA

Cilj PCA je da predstavi uzorce iz visokodimenzionalnog prostora u prostoru sa manjim brojem dimenzija što vernije, odnosno, zadržavajući u što većoj meri varijansu sadržanu u podacima. PCA predstavlja metodu nenadgledanog učenja, jer ne koristi informaciju o vrednosti izlaza za uzorce iz skupa za obuku. Polazeći od pretpostavke da su raspoloživa obeležja korelisana i da postoji redundantnost u višedimenzionalnoj reprezentaciji podataka, cilj PCA je određivanje novih nekorelisanih obeležja (PCA komponente) linearnim kombinacijama postojećih obeležja uz očuvanje varijanse u podacima. Izborom najinformativnijih PCA komponenata obrazuje se novi prostor sa manjim brojem



**Slika 12.1:** Redukcija prostora sa dve dimenziije na jednu dimenziiju prema različitim kriterijumima za redukciju dimenzionalnosti.

dimenzija i uzorci se projektuju na njega. Najinformativnije PCA komponente biće one koje odgovaraju pravcima najvećeg rasipanja uzorka. Sledi formalni koraci PCA algoritma.

1. Standardizacija obeležja. Kako je za PCA bitno samo rasipanje uzorka, potrebno je centrirati ih, odnosno svesti srednju vrednost svakog obeležja na 0. Međutim, kada obeležja imaju različite opsege vrednosti, neophodno je svesti standardnu devijaciju svih obeležja na 1. Ovim se sprečava prisitasnost PCA prema obeležjima koja imaju veći opseg vrednosti.
2. Formiranje kovarijanske matrice i određivanje njenih karakterističnih vektora i vrednosti. Kovarijansna matrica  $\Sigma$  predstavlja matricu  $D \times D$  čiji svaki element predstavlja kovarijansu između dva obeležja. Karakteristični vektori kovarijanske matrice predstavljaju glavne komponente, odnosno određivače glavne pravce novog prostora obeležja. Varijansa sadržana u podacima jednim delom se odnosi na „korisnu“ informaciju, odnosno, informaciju koja je u određenoj vezi sa posmatranim obeležjima. Značaj svake glavne komponente, PCA komponente, određen je varijansom uzorka

duž njenog pravca i opisuje se odgovarajućom karakterističnom vrednošću. Imajući u vidu da suma svih karakterističnih vrednosti kovarijanske matrice opisuje ukupnu varijansu u podacima, udio svake pojedinačne glavne komponente u ukupnoj varijansi ukazuje na njen značaj i određuje se kao:

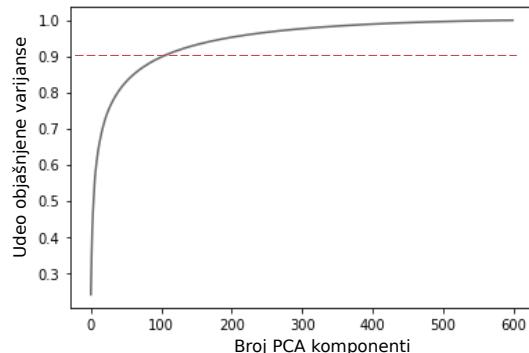
$$\text{udio\_komponente\_}i = \frac{\lambda_i}{\sum_{i=1}^d \lambda_i}, \quad (12.1)$$

gde je  $\lambda_i$  karakteristična vrednost koja odgovara  $i$ -toj komponenti. Prema tome, svaka PCA komponenta predstavlja određen pravac i rasipanje uzoraka duž njega i time „objašnjava“ deo varijanse u podacima.

3. Izdvajanje podskupa od  $M$  karakterističnih vektora. Svakom karakterističnom vektoru odgovara po jedna karakteristična vrednost. Vektor kojem odgovara najveća karakteristična vrednost biće proglašen za prvu PCA komponentu jer je to pravac duž kog je varijansa podataka najveća. Sledeća PCA komponenta odgovaraće sledećoj najvećoj karakterističnoj vrednosti. Pravac te komponente poklapaće se sa pravcem drugog po redu najvećeg rasipanja uzoraka u originalnom prostoru, koji je pritom normalan na pravac prve PCA komponente. Postupak se nastavlja na isti način, da bi se na kraju dobilo ukupno najviše  $D$  PCA komponenata, odnosno onoliko koliko ima dimenzija u originalnom prostoru (osim ako je broj uzoraka manji od broja dimenzija, kada je maksimalan broj PCA komponenata jednak broju uzoraka). Kako je cilj smanjenje dimenzionalnosti uz očuvanje varijanse, za kreiranje novog prostora obeležja bira se prvih  $M$  PCA komponenata, kojima odgovara  $M$  najvećih karakterističnih vrednosti matrice  $\Sigma$ .
4. Projekcija uzoraka na  $M$ -dimenzionalni prostor. Novi prostor opisan je sa novih  $M$  obeležja, koja u opštem slučaju nemaju fizičku interpretaciju preko originalnih obeležja, iako se može utvrditi sa kojim obeležjima polaznog prostora imaju najveću korelaciju. PCA komponente su međusobno ortogonalne, odnosno, dobijena obeležja nisu korelisana.

Izbor broja osnovnih komponenata  $M$ , odnosno dimenzionalnosti novog prostora zavisi u velikoj meri od problema koji se rešava. Jedna od primene PCA je vizuelizacija podataka, i u tom slučaju  $M$  očigledno ne sme biti veće od 3 jer u prostoru sa više od 3 dimenzije nije moguće izvršiti vizuelizaciju. Za sve ostale primene teško je unapred odgovoriti koliko PCA komponenata treba zadržati. Često se posmatra kako broj zadržanih komponenata  $M$  utiče na odnos

varijanse koja se zadržava u prostoru obeležja dimenzionalnosti  $M$  (objašnjen deo varijanse) i ukupne varijanse koja je postojala u polaznom prostoru obeležja dimenzionalnosti  $D$ . Traži se takav broj  $M$ , da dodavanje preostalih PCA komponenata ima veoma mali doprinos porastu objašnjene varijanse (primera radi, na Slici 12.2 se vidi da je dovoljno zadržati 100 komponenata, pa čak i manje, jer poslednjih 500 komponenata doprinose ukupnoj varijansi manje od 10%). Ukoliko se PCA komponente koriste kao obeležja za klasifikaciju ili regresiju, preporuka je da se optimalan broj komponenata, odnosno, broj komponenata koji će obezbediti najbolje performanse algoritma, odredi unakrsnom validacijom. Redukcija dimenzionalnosti korišćenjem PCA metode ne garantuje ostvarivanje boljih performansi algoritma u poređenju sa performansama postignutim nad originalnim skupom obeležja, pa se iz tih razloga preporučuje da se uporede performanse algoritama sa redukcijom dimenzionalnosti korišćenjem PCA i bez nje. Na primer, smanjenje dimenzionalnosti može doprineti boljoj klasifikaciji, ali to ne mora biti slučaj jer PCA pri smanjenju dimenzionalnosti ne uzima u obzir klasne labele.



**Slika 12.2:** Grafik zavisnosti objašnjenog dela varijanze od broja PCA komponenata uzetih u obzir.

U **Scikit-learn** biblioteci za PCA metodu postoji implementirana klasa **PCA** i nalazi se u modulu **decomposition**. S obzirom da služi za redukciju dimenzionalnosti, implementirane metode su PCA za inicijalizaciju, **fit** za obuku (pronalaženje PCA komponenata) i **transform** za projekciju uzorka na dimenzijske novog prostora. Glavni parametar klase je **n\_components**. Ako se vrednost ne precizira, zadržavaju se sve komponente, tj. **min(broj\_obeležja, broj\_uzoraka)**. Vrednost ovog parametra može biti ili broj komponenata koje

se zadržavaju (celobrojna vrednost) ili udeo varijanse (vrednost između 0 i 1), što podrazumeva da se zadrži onaj deo komponenata, počev od prve, čiji je sumirani udeo u objašnjenu varijanse jednak traženom. Vrednost takođe može biti mle, što će znaciti automatski odabir broja PCA komponenata prema Minkinom MLE algoritmu.

Primer primene PCA algoritma uz prethodnu standardizaciju podataka s ciljem da se zadrže komponente koje obuhvataju 95% ukupne varijanse:

```
s = StandardScaler()
s.fit(X_train)
X_train_std = s.transform(X_train)
X_test_std = s.transform(X_test)

pca = PCA(n_components=0.95)
pca.fit(X_train_std)
X_train_r = pca.transform(X_train_std)
X_test_r = pca.transform(X_test_std)
```

U praksi je često potrebno ispitati zavisnost objašnjeno delu varijanse, odnosno, od broja zadržanih PCA komponenata. Da bi se ta zavisnost grafički prikazala, potrebno je zadržati sve komponente. Udeo objašnjene varijanse po komponenti može se dobiti pomoću izlaznog parametra `explained_variance_ratio`, što je pokazano u sledećem kodu:

```
pca = PCA()
pca.fit(X_train_std)
plt.plot(np.cumsum(pca.explained_variance_ratio_))
```

## 12.2. Linearna diskriminantna analiza - LDA

Cilj linearne diskriminantne analize jeste da se dimenzionalnost prostora smanji, a da se pritom očuva što više diskriminatornih informacija. Formiranjem linearne kombinacije postojećih obeležja LDA pronalazi pravac  $w$  takav da se prilikom projekcije uzorka na njega maksimizuje separabilnost između klasa (Slika 12.3). Uzimanje u obzir klasnih labela svrstava ovu metodu među metode nadgledanog učenja.

Poboljšanje razdvojivosti između klase postiže se maksimizacijom rastojanja između srednjih vrednosti LDA projekcija uzorka iz različitih klasa i istovremenom minimizacijom varijanse projekcije uzorka unutar svake klase

pojedinačno. Kod binarne klasifikacije rezultat primene LDA na uzorak  $\mathbf{x}$  predstavlja skalar  $y = \mathbf{w}^\top \mathbf{x}$  (Slika 12.3). Ovaj skalar predstavlja projekciju uzorka  $\mathbf{x}$  na pravac  $\mathbf{w}$ , koji se pronalazi maksimizacijom funkcije cilja  $J(\mathbf{w})$ . Funkcija cilja je definisana kao kvadrat apsolutne vrednosti razlike srednjih vrednosti projekcija unutar klasa  $\mu_k$ , normalizovan merom rasipanja projektovanih uzoraka unutar klasa:

$$J(\mathbf{w}) = \frac{|\mu_1 - \mu_2|^2}{s_1^2 + s_2^2}, \quad (12.2)$$

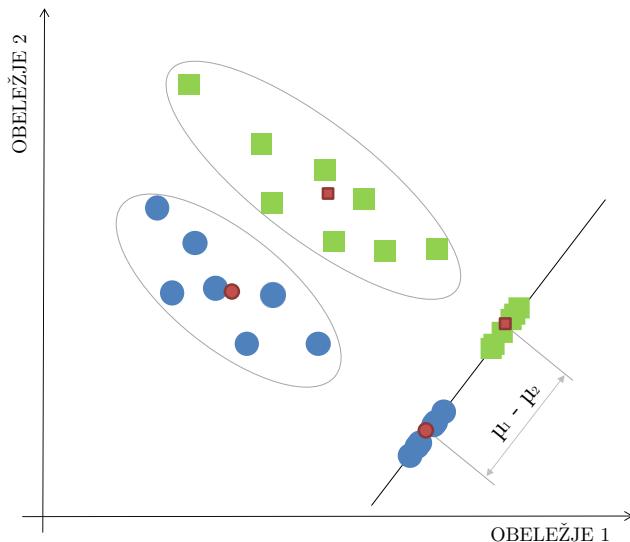
gde je  $s_k^2$  rasipanje projekcija uzoraka iz klase  $K_k$  na pravac  $\mathbf{w}$  i predstavlja ekvivalent varijansi, a definisano je izrazom:

$$s_k^2 = \sum_{n=1}^{N_k} (y^{(n)} - \mu_k)^2, \quad (12.3)$$

gde je  $N_k$  broj uzoraka klase  $K_k$ . Rezultat LDA u slučaju binarne klasifikacije predstavlja, dakle, projekciju uzorka na pravac odabran tako da su uzorci iz iste klase projektovani što bliže jedni drugima, dok su centroidi klasa istovremeno što više udaljeni jedan od drugog. Višestrukom primenom binarne klasifikacije ili direktnom generalizacijom metode na slučaj više dimenzija, LDA se može primeniti i na probleme klasifikacije u  $K$  klasa, gde je  $K > 2$ .

LDA se može koristiti i kao klasifikator i kao algoritam za redukciju dimenzionalnosti. Ukoliko je raspodela obeležja po klasama multivarijabilna Gausova raspodela, a kovarijansne matrice su iste u obe klase, LDA se može povezati sa Bajesovom klasifikacijom (Poglavlje 4.1, klasifikacija na osnovu minimalnog rastojanja), pri čemu se klasifikacija vrši na osnovu rastojanja projekcija uzorka od projekcija odgovarajućih centroida. Iz tog razloga, ako se LDA koristi kao klasifikator, a raspodele obeležja po klasama odstupaju od Gausove multivarijabilne raspodele, preporučljivo je pored standardizacije obeležja primeniti i tehnike predobrade koje će raspodelu što više približiti Gausovoj (na primer, logaritmovati ili korenovati vrednosti obeležja). S druge strane, cilj LDA može biti i redukcija dimenzionalnosti pre primene nekog drugog algoritma za klasifikaciju. Primenom LDA svakako će se maksimizovati rastojanje između projekcija centroida uz istovremenu minimizaciju varijansi unutar pojedinih klasa, ali se u opštem slučaju ne može izbeći određeno preklapanje između klasa. To samo znači da ni klasifikacija u novom prostoru neće biti idealna, ali je ipak moguće da bude tačnija u odnosu na direktnu klasifikaciju bez primene LDA.

Za LDA metodu postoji implementirana klasa `LDA` i nalazi se u modulu `lida`. S obzirom da služi za redukciju dimenzionalnosti i za klasifikaciju,



**Slika 12.3:** Prikaz smanjenja dimenzionalnosti prostora sa dve dimenziye na jednu primenom LDA algoritma.

implementirane metode su LDA za inicijalizaciju, `fit` za obuku (pronalaženje LDA komponenata i obuku klasifikacionog modela), `transform` za projekciju uzorka na dimenziye novog prostora i `predict` za klasifikaciju. Glavni ulazni parametar klase je `n_components`. Može se postaviti na konkretnu celobrojnu vrednost, koja će odrediti koliko komponenata da se zadrži. Maksimalno je moguće zadržati `min(broj_klase-1, broj_uzoraka)` komponenata, što je ujedno i podrazumevana vrednost parametra. Primer primene LDA algoritma sa standardizovanim podacima s ciljem da se zadrže dve LDA komponente:

```
s = StandardScaler()
s.fit(X_train)
X_train_std = s.transform(X_train)
X_test_std = s.transform(X_test)

lda = LDA(n_components=2)
lda.fit(X_train, y_train)
X_train_r = lda.transform(X_train_std)
X_test_r = lda.transform(X_test_std)
```

### 12.3. Zadaci

#### Zadatak 1

Implementirati funkciju `PCA_decomp` ( $\mathbf{X}$ ,  $M$ ) koja kao izlazne vrednosti vraća sledeće: `sopstvene_vrednosti`, `sopstveni_vektori`, `udeo_komponente`, `X_nakon_redukcije`. Koraci algoritma su:

1. Centriranje uzoraka, odnosno svođenje srednjih vrednosti obeležja na 0 (može se izvršiti korišćenjem klase `StandardScaler` ili po definiciji).
2. Računanje kovarijanske matrice.
3. Računanje karakterističnih vektora i karakterističnih vrednosti kovarijanske matrice. U `numpy.linalg` modulu postoji funkcija `eig`, koja za prosledenu kovarijansnu matricu vraća vektor karakterističnih vrednosti i matricu čije kolone predstavljaju karakteristične vektore. Redosled karakterističnih vektora odgovara redosledu karakterističnih vrednosti.
4. Sortiranje karakterističnih vrednosti i izračunanje udela određene komponente u ukupnoj varijansi prema obrascu:

$$\text{udeo\_komponente\_}i = \frac{\lambda_i}{\sum_{i=1}^d \lambda_i},$$

gde je  $\lambda_i$  karakteristična vrednost koja odgovara  $i$ -toj komponenti.

5. Sortiranje karakterističnih vektora u skladu sa sortiranim karakterističnim vrednostima.
6. Projekcija uzoraka na prvih  $M$  karakterističnih vektora prema obrascu:

$$\mathbf{X}_{\text{redukovano}} = (\mathbf{V}^\top \mathbf{X}^\top)^\top,$$

gde je  $\mathbf{V}$  matrica koja sadrži prvih  $M$  karakterističnih vektora, a  $\mathbf{X}$  matrica koja sadrži uzorke iz  $D$ -dimenzionalnog prostora. Konačno  $\mathbf{X}_{\text{redukovano}}$  sadržaće uzorke projektovane na  $M$ -dimenzionalni prostor.

Generisati na slučaj 200 uzoraka iz dvodimenzionalne raspodele  $\mathcal{N}(\mathbf{0}, \mathbf{1})$ . Ilustrovati podatke grafikom rasipanja. Potom primeniti funkciju `PCA_decomp` i ilustrovati originalne podatke i karakteristične vektore, a na drugom grafiku rasipanja dobijene podatke u redukovanim prostoru. Zadatak uraditi za  $M = 1$  i za  $M = 2$ .

**Zadatak 2**

Implementirati algoritam za klasifikaciju cifara pisanih rukom. Koristiti bazu podataka<sup>9</sup> koja sadrži slike rukom pisanih cifara veličine 28x28, konvertovane u formu vektora 1x784 i smeštene u matricu po vrstama. Koristiti samo podatke za koje je navedeno da su namenjeni za obuku jer podaci namenjeni za testiranje ne sadrže klasne labele.

1. Učitati podatke u `DataFrame`. Koliko ima uzoraka? Šta predstavljaju obeležja u ovom skupu podataka? Da li su klase jednakozastupljene?
2. Korišćenjem ugrađenih funkcija iz `Scikit-learn` biblioteke za PCA i LDA redukovati dimenzionalnost skupa za obuku na 2 i vizuelizovati podatke na grafiku rasipanja prikazujući uzorke iz različitih klasa različitim bojama. Koji algoritam je postigao bolje rezdvajanje klasa? Da li je u ovom slučaju 2 ipak premala dimenzionalnost za rešavanje problema klasifikacije?
3. Iz skupa podataka nasumično izdvojiti 30% uzoraka koji će biti korišćeni isključivo za testiranje.
4. Korišćenjem ugrađene funkcije PCA i 1NN klasifikatora, izvršiti klasifikaciju koristeći originalne podatke i koristeći podatke nakon primene PCA. Uporediti rezultate na osnovu udela ispravno klasifikovanih uzoraka iz test skupa. Na koliko dimenzija je redukovani prostor? Kako broj dimenzija redukovanih prostora utiče na tačnost klasifikacije?
5. Korišćenjem ugrađene funkcije LDA i 1NN klasifikatora izvršiti klasifikaciju koristeći podatke nakon primene LDA. Uporediti rezultate na osnovu udela ispravno klasifikovanih uzoraka iz test skupa sa rezultatom dobijenim pre redukcije dimenzionalnosti. Na koliko dimenzija je prostor redukovani? Koliki je maksimalan broj dimenzija prostora redukovanih primenom LDA? Kako broj dimenzija redukovanih prostora utiče na tačnost klasifikacije?
6. Da li je pri rešavanju ovog problema korišćenje LDA algoritma bilo opravданo? Koji algoritam za redukciju dimenzionalnosti je dao bolji rezultat pri rešavanju problema prepoznavanja rukom pisanih cifara 1NN metodom?

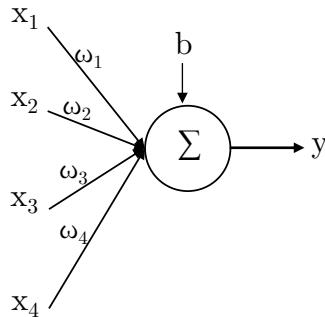
---

<sup>9</sup>Podaci su dostupni na: [www.kaggle.com/c/digit-recognizer/data](http://www.kaggle.com/c/digit-recognizer/data)

## 13. Uvod u neuronske mreže

Veštačke neuronske mreže predstavljaju složene računarske sisteme nastale povezivanjem veštačkih neurona, inspirisane načinom funkcionisanja neurona u ljudskom mozgu. Iako svoju primenu pronalaze i u nenadgledanom učenju, u ovom poglavlju razmatraju se isključivo kao algoritam nadgledanog učenja, i kao takve imaju vrlo izraženu sposobnost generalizacije.

Osnova svake neuronske mreže jeste veštački neuron. Najjednostavniji veštački neuron naziva se perceptron. Ulaz  $\mathbf{x}$  perceptrona predstavljen je u obliku realnog vektora podataka, a njegov izlaz  $y$  je jedan binarni podatak, 0 ili 1. Svaki od ulaza množi se težinskim faktorom  $w_i$ , koji opisuje doprinos ulaza pri računanju izlaza (Slika 13.1). Izlaz perceptrona zavisi od toga da li je



**Slika 13.1:** Dijagram perceptrona.

odgovarajuća linearna kombinacija ulaza veća od vrednosti unapred određenog praga  $p$ :

$$y = \begin{cases} 1, & \mathbf{w}^\top \mathbf{x} \geq p \\ 0, & \mathbf{w}^\top \mathbf{x} < p \end{cases}, \quad (13.1)$$

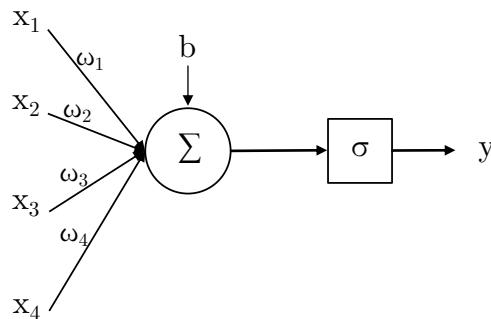
gde je  $\mathbf{x}$  vektor ulaza,  $\mathbf{w}$  vektor njima odgovarajućih težina, a  $y$  predstavlja izlaz neurona. Umesto praga  $p$  koristi se slobodan član  $b = -p$  koji se naziva pomerajem (engl. *bias*), pa se vrednost izlaza neurona može predstaviti na sledeći način:

$$y = \begin{cases} 1, & \mathbf{w}^\top \mathbf{x} + b \geq 0 \\ 0, & \mathbf{w}^\top \mathbf{x} + b < 0 \end{cases}. \quad (13.2)$$

Problem kod korišćenja perceptrona je činjenica da male promene težina mogu da dovedu do značajnih promena izlaza (0 na 1 ili obrnuto), a poželjno je da

male promene težina malo utiču na rezultat, te da se uz taj uslov mreži omogući da samostalno pronađe optimalne težine. Rešenje ovog problema je u uvođenju nelinearne transformacije (aktivacione funkcije) linearne kombinacije  $\mathbf{w}^\top \mathbf{x} + b$  na izlazu neurona (Slika 13.2). Ako je aktivaciona funkcija  $\sigma$ , izlaz neurona definisan je izrazom:

$$y = \sigma(\mathbf{w}^\top \mathbf{x} + b) \quad (13.3)$$



Slika 13.2: Šema veštačkog neurona.

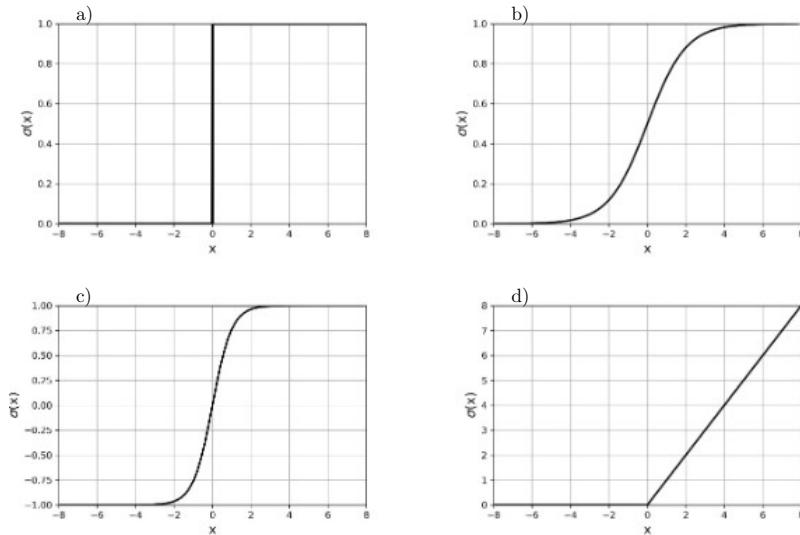
U praksi se mogu koristiti različiti tipovi aktivacionih funkcija. U slučaju da je aktivaciona funkcija Hevisajdova funkcija (Slika 13.3a), neuron se svodi na perceptron. Mnogo češće se, međutim, koriste neprekidne aktivacione funkcije kao što su sigmoidalna  $\sigma(x)$  (Slika 13.3b), tangens hiperbolički  $\sigma_{\tanh}(x)$  (Slika 13.3c) i ispravljačka linearna jedinica (engl. *Rectified Linear Unit – ReLU*)  $\sigma_{\text{ReLU}}(x)$  (Slika 13.3d) definisane sledećim izrazima:

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \quad (13.4)$$

$$\sigma_{\tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \quad (13.5)$$

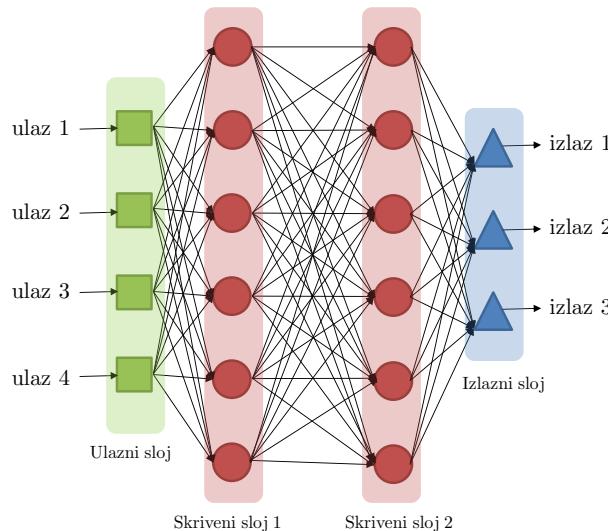
$$\sigma_{\text{ReLU}}(x) = \max(0, x). \quad (13.6)$$

Za složenije probleme efikasnim se pokazalo kombinovanje neurona u slojeve neuronske mreže (Slika 13.4). Prvi sloj neuronske mreže naziva se ulaznim slojem i prima ulazne podatke (vrednosti ulaznih obeležja), čime je definisana i njegova veličina, odnosno broj neurona u njemu. Svaki naredni sloj kao ulaze



**Slika 13.3:** Grafici aktivacionih funkcija: a) Hevisajdova funkcija, b) sigmoidalna funkcija, c) tangens hiperbolički i d) ispravljajuća linearna jedinicna.

prima izlaze iz prethodnog sloja. Poslednji (izlazni) sloj izračunava vrednosti izlaznih obeležja, što određuje i koliko neurona on treba da sadrži. Primera radi, ako se posmatra problem klasifikacije slika dimenzije  $10 \times 10$  piksela u 4 klase, gde svaki piksel predstavlja po jedno obeležje, dimenzija ulaznog sloja neuronske mreže jednaka je 100, a dimenzija izlaznog sloja, koji izračunava verovatnoću pripadnosti određenoj klasi, jednaka je 4. Slojevi između ulaznog i izlaznog nazivaju se skrivenim slojevima i mogu biti proizvoljne veličine. U pomenutom primeru skup za obuku sadrži slike i odgovarajuće klasne labele svih slika, čime se za svaki ulaz eksplisitno definiše željeni izlaz mreže, ali ne i izlazi svih skrivenih slojeva. Upravo zato što izlazi skrivenih slojeva nisu definisani skupom za obuku, niti direktno dostupni, nazvani su skrivenim. Ukoliko izlaz svakog neurona iz određenog sloja ujedno predstavlja ulaz svakog neurona narednog sloja, slojevi se smatraju potpuno povezanim (engl. *fully connected layers*). Broj slojeva određuje dubinu mreže, zbog čega se mreže sa više slojeva i nazivaju dubokim neuronskim mrežama (engl. *deep neural networks* – DNN).



**Slika 13.4:** Dijagram potpuno povezane neuronske mreže sa dva skrivena sloja.

Postoje različite arhitekture neuronskih mreža, a izbor arhitekture često zavisi od oblasti primene. Mreže kod koje se informacije prostiru samo od ulaza ka izlazu nazivaju se nerekurzivnim mrežama (engl. *feedforward*), a u literaturi često i višeslojnim perceptronima (što ne znači nužno da su aktivacione funkcije neurona Hevisajdove funkcije). Mreže kod kojih postoji petlje nazivaju se rekurzivnim neuronskim mrežama (engl. *recurrent neural networks* – RNN), i one su naročito korisne kod problema kod kojih ulazni podaci predstavljaju vremenski niz, kao što je, primera radi, slučaj u govornim tehnologijama. Drugu veoma značajnu grupu mreža, specifičnih po tome što koriste konvoluciju kao operaciju nad ulaznim podacima, čine konvolucione neuronske mreže (engl. *convolutional neural networks* – CNN). Ove mreže pretežno se koriste u obradi slike, kako za klasifikaciju tako i za segmentaciju.

Kao i u drugim algoritmima mašinskog učenja, određivanje vrednosti parametara (u ovom slučaju težina i pomeraja) vrši se u odnosu na funkciju cene, kao meru odstupanja predviđenih vrednosti izlaza od željenih (ciljnih). Kada se mreža koristi za rešavanje regresionog problema, najčešći izbor funkcije cene je srednja kvadratna greška (Izraz 13.7), dok je u slučaju klasifikacionog

problema najčešći izbor međuentropijska funkcija (Izraz 13.8):

$$J_{mse} = \frac{1}{2} \sum_{i=1}^N \sum_{m=1}^{k_L} (\hat{y}_m^{(i)} - y_m^{(i)})^2 , \quad (13.7)$$

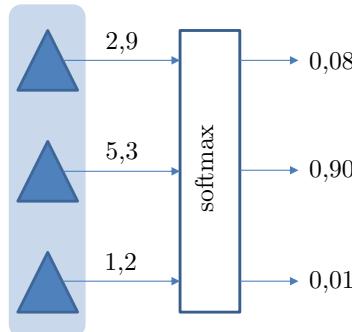
$$J_{ce} = -\frac{1}{2} \sum_{i=1}^N \sum_{m=1}^{k_L} y_m^{(i)} \ln(\hat{y}_m^{(i)}) + (1 - y_m^{(i)}) \ln(1 - \hat{y}_m^{(i)}) , \quad (13.8)$$

gde je  $y_m^{(i)}$  željena izlazna vrednost  $m$ -tog neurona izlaznog sloja mreže za  $i$ -ti uzorak,  $\hat{y}_m^{(i)}$  je izlazna vrednost  $m$ -tog neurona koju mreža predviđa za  $i$ -ti uzorak,  $N$  je ukupan broj uzoraka, a  $k_L$  broj neurona u izlaznom sloju mreže. Zbog nelinearnosti aktivacione funkcije i velikog broja parametara mreže funkcija cene nije konveksna, pa se za minimizaciju funkcije cene koriste iterativni postupci, uglavnom zasnovani na algoritmu opadanja gradijenta. Ishod ovakvog postupka zavisi od inicijalnih parametara mreže, tako da se ne može garantovati dostizanje globalnog minimuma. Međutim, ovo ne predstavlja problem ukoliko je razlika između globalnog minimuma i postignutog lokalnog minimuma mala. Proces obuke mreže započinje inicijalizacijom parametara mreže slučajnim malim vrednostima. Potom se propagacijom ulaznih podataka unapred (engl. *forward propagation*) izračunavaju izlazi mreže na osnovu vrednosti trenutnih parametara mreže. Na osnovu dobijenih izlaza mreže i željenih izlaza mreže (datih u skupu za obuku) izračunava se vrednost funkcije cene. Sledeći korak je algoritam propagacije unazad (engl. *backpropagation algorithm*). Ovaj algoritam omogućava izračunavanje vrednosti parcijalnih izvoda funkcije cene po parametrima mreže. Time se određuje kako parametri izlaznog sloja utiču na vrednost funkcije cene, zatim kako ulazi tog sloja utiču na izlaze, a potom se prethodni korak ponavlja za svaki prethodni sloj, redom do ulaznog sloja. Kada se odrede vrednosti svih izvoda, svi parametri se ažuriraju korišćenjem neke od optimizacionih metoda u pravcu opadanja gradijenta. Veličina koraka u pravcu opadanja gradijenta određena je brzinom učenja. Premala brzina učenja može da uspori proces konvergencije ka minimumu, dok prevelika brzina može da dovede do previše grube pretrage i preskakanja globalnog minimuma funkcije cene. Putanja ka minimumu može značajno da osciluje tokom učenja, a različiti optimizacioni algoritmi za ažuriranje parametara mreže teže da priguše te oscilacije i na taj način ubrzaju konvergenciju. Neki od češće korišćenih algoritama su stohastični algoritam opadanja gradijenta sa momentom (engl. *stochastic gradient descent with momentum*) i Adam (engl. *adaptive moment estimation*). Čest izbor aktivacione funkcije u neuronskoj mreži je sigmoidalna. Međutim,

neretko se dešava da moduo gradijenta postane veoma mali (problem nestajućeg gradijenta, engl. *vanishing gradient*) što dovodi do neznatne promene parametara (težina i pomeraja) pa mreža prestaje da uči. Prednost ReLU aktivacione funkcije u odnosu na sigmoidalnu aktivacionu funkciju ogleda se u tome što zbog načina na koji je definisana za velike vrednosti ulaza ne dovodi do problema nestajućeg gradijenta. Međutim, javlja se problem što u potpunosti filtrira, tj. ne propušta negativne vrednosti. Zbog toga neuroni sa negativnim izlazima postaju neaktivni i ne doprinose učenju, što smanjuje teorijski maksimum mogućih performansi algoritma. Postoje i razne modifikacije ReLU funkcije koje prevazilaze pomenuti problem. Izbor aktivacione funkcije zavisi od problema koji se rešava. Još jedna aktivaciona funkcija, koja se uglavnom koristi samo za izlazni sloj mreže pri rešavanju klasifikacionih problema, jeste *softmax* funkcija. Ova normalizovana eksponencijalna funkcija data je izrazom:

$$S(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{j=1}^K e^{x_j}}, \quad (13.9)$$

gde je  $\mathbf{x}$  vektor izlaza iz mreže,  $x_i$  je  $i$ -ti izlaz mreže, a  $K$  broj klasa (odnosno broj izlaza iz mreže). Ova funkcija pretvara vektor izlaznih vrednosti mreže u raspodelu koja određuje kolika je verovatnoća pripadnosti odgovarajućeg ulaza u mrežu svakoj od klasa. Na Slici 13.5 dat je primer primene *softmax* funkcije na izlaze mreže u problemu sa tri klase.



**Slika 13.5:** Ilustracija primene *softmax* aktivacione funkcije u izlaznom sloju neuronske mreže.

Jedna *epocha* obuke neuronske mreže predstavlja jedan prolazak svih uzoraka za obuku kroz mrežu. Mreža se tipično obučava kroz veći broj epoha, a

obuka se prekida kada prođe unapred određen broj epoha ili kada se vrednost funkcije cene u unapred određenom broju uzastopnih epoha ne smanjuje (ili se ne menja više od zadatog praga). Vrednosti parametara mreže mogu se ažurirati nakon čitave epohe, ali je to u praksi najčešće neizvodljivo zbog ograničenja radne memorije. Drugu krajnost predstavlja ažuriranje vrednosti parametara nakon svakog uzorka, ali ono najčešće dovodi do stohastičnih promena i spore konvergencije. Zato se vrednosti parametara u praksi najčešće ažuriraju nakon prolaska jednog podskupa uzoraka zadate veličine (eng. *batch*) kroz mrežu.

Radi dodatnog ubrzanja učenja po pravilu se vrši normalizacija ulaza mreže. Bržu konvergenciju smanjenjem zavisnosti između slojeva omogućuje i tzv. *batch* normalizacija. Ovaj postupak normalizuje izlaze skrivenih slojeva, čime se takođe vrši i regularizacija, jer se srednja vrednost i varijansa računaju na nivou *batch-a*, a ne celog skupa za obuku. Postoje i metode koje su namenjene isključivo za regularizaciju, a standardno se koriste  $L_2$  regularizacija i *dropout* metoda.  $L_2$  regularizacija sprečava pojavu veoma velikih vrednosti parametara koji sugerisu na natprilagođenje. *Dropout* metoda tokom obuke zanemaruje izlaze pojedinih neurona sa unapred određenom verovatnoćom, izjednačavajući ih sa nulom. Na taj način eliminise međusobnu zavisnost neurona i sprečava natprilagođenje modela. Dobra praksa je i promena redosleda prosleđivanja uzorka mreži u svakoj epohi. Praćenjem greške na skupu za obuku i validaciju, može se uočiti natprilagođenje, odnosno porast greške na validacionom skupu i istovremeno smanjenje greške na skupu za obuku. Obuka se u tom slučaju može zaustaviti pre ispunjenja unapred postavljenih kriterijuma za zaustavljanje, što se naziva ranim zaustavljanjem (engl. *early stopping*). Rano zaustavljanje primenjuje se kada u nekoliko uzastopnih epoha ne dolazi do smanjenja vrednosti funkcije cene na skupu uzoraka koji je unapred izdvojen za validaciju.

Za implementaciju neuronskih mreža uglavnom se koriste specijalizovane biblioteke i okruženja poput **Tensorflow**, **PyTorch**, **Keras** i **CNTK**. Za potrebe ove vežbe koristiće se implementacija u okviru biblioteke **Scikit-learn** u modulu **neural\_network** kroz klase **MLPClassifier** i **MLPRegressor**. Ove klase predstavljaju implementaciju samo najjednostavnijih nerekurzivnih potpuno povezanih neuronskih mreža sa ograničenom mogućnošću podešavanja vrednosti hiperparametara kao što su broj slojeva ili broj neurona u svakom sloju mreže. Prva klasa je namenjena za primenu u klasifikacionim problemima, i koristi međuentropiju kao funkciju cene i *softmax* aktivacionu funkciju u izlaznom sloju za višeklasne probleme. Druga klasa je namenjena za rešavanje regresionih problema, i koristi srednju kvadratnu grešku kao funkciju cene bez aktivacione funkcije u izlaznom sloju. Parametri ove dve klase su isti i omogućuju podešavanje hiperparametara modela:

- **hidden\_layer\_sizes**: definiše arhitekturu mreže kroz broj i veličinu skrivenih slojeva; npr. (100, 250) označava dva skrivena sloja od kojih prvi ima 100 neurona a drugi 250; podrazumevana vrednost je (100), tj. jedan sloj sa 100 neurona.
- **activation**: definiše izbor aktivacione funkcije u neuronima skrivenih slojeva; može se izabrati `identity`, što je praktično isto kao i da nema aktivacione funkcije, `logistic` za sigmoidalnu funkciju, `tanh` za tangens hiperbolički i `relu` za ispravljačku linearu jedinicu; podrazumevana vrednost je `relu`.
- **solver**: definiše izbor optimizacionog algoritma za ažuriranje parametara mreže; može se odabratи `sgd` (stohastično opadanje gradijenta), `adam`, koji je ujedno i podrazumevani algoritam ili `lbfgs` što je još jedan od često korišćenih optimizacionih algoritama.
- **alpha**: parametar  $L_2$  regularizacije; podrazumevana vrednost je 0,0001.
- **batch\_size**: broj uzoraka koji čine jedan *batch*; podrazumevana vrednost je `auto`, kada se veličina definiše kao  $\min(200, N)$  gde je  $N$  broj uzoraka. Kada je algoritam za optimizaciju `lbfgs`, ovaj parametar se ne uzima u obzir.
- **learning\_rate**: određuje način promene brzine učenja tokom obuke; može biti `constant`, što znači da se brzina učenja tokom obuke ne menja, `adaptive`, što znači da se brzina učenja smanjuje 5 puta kada se desi da u dve uzastopne epohe nije došlo do smanjenja funkcije cene bar za prag `tol` (definisan posebnim parametrom) ili nije došlo do smanjenja funkcije cene na validacionom skupu za bar `tol`, a uključen je `early_stopping`. Poslednja moguća vrednost je `invscaling`, i ona obezbeđuje postepeno smanjenje brzine učenja. Ovaj parametar koristi se samo u slučaju da se koristi stohastično opadanje gradijenta, tj. kada je za optimizacioni algoritam izabran `sgd`.
- **learning\_rate\_init**: inicijalna vrednost funkcije cene; podrazumevana vrednost je 0,001.
- **power\_t**: parametar koji se koristi za promenu brzine učenja kroz epohe, u slučaju da je način promene podešen na `invscaling`; podrazumevana vrednost je 0,5.

- **max\_iter**: maksimalan broj epoha za obuku; podrazumevana vrednost je 200.
- **shuffle**: određuje da li treba menjati redosled prosleđivanja uzoraka za obuku u svakoj epohi; podrazumevana vrednost je **True**.
- **random\_state**: celobrojna vrednost koja se koristi radi ponovljivosti rezultata za sve postupke koji se vrše na slučaj (inicijalizacija parametara mreže, izbor validacionog skupa za **early\_stopping**, izbor *batch-a*).
- **tol**: prag za smanjenje funkcije cene, čija upotreba je objašnjena kod parametra **learning\_rate**; podrazumevana vrednost je  $10^{-4}$ .
- **early\_stopping**: određuje da li će se koristiti mehanizam ranog zaustavljanja obuke; podrazumevana vrednost je **False**.
- **n\_iter\_no\_change**: definiše maksimalan broj uzastopnih epoha koji će biti uzet u obzir kod mehanizma ranog zaustavljanja obuke modela; podrazumevana vrednost je 10.
- **validation\_fraction**: određuje udeo skupa za obuku koji će biti korišćen kao validacioni skup kod mehanizma ranog zaustavljanja; podrazumevana vrednost je 0,1.

Primer obuke i testiranja klasifikatora neuronske mreže sa 3 skrivena sloja od po 25 neurona:

```
classifier = MLPClassifier(hidden_layer_sizes=(25,25,25),
                           max_iter=500, solver='sgd')
classifier.fit(X_train, y_train)
y_predicted = classifier.predict(X_test)
```

### 13.1. Zadaci

#### Zadatak 1

Implementirati algoritam za klasifikaciju na bazi neuronske mreže. Koristiti bazu podataka koja sadrži akustička obeležja (mel-frekvencijeske kepstralne koeficijente – MFCC) izdvojena iz audio snimaka<sup>10</sup>. Svaki snimak predstavlja

<sup>10</sup>Podaci su dostupni na: [archive.ics.uci.edu/ml/datasets/Speaker+Accent+Recognition](http://archive.ics.uci.edu/ml/datasets/Speaker+Accent+Recognition)

jednu izgovorenu reč na engleskom jeziku, gde su govornici poreklom iz različitih država: Sjedinjenih Američkih Država, Velike Britanije, Španije, Italije, Nemačke i Francuske. U govoru se jasno mogu čuti različiti akcenti (dato je nekoliko primera audio snimaka). Zadatak je obućiti neuronsku mrežu koja na osnovu izdvojenih akustičkih obeležja kao izlaz vraća iz koje države potiče govornik.

1. Učitati podatke u `DataFrame` strukturu. Koliko ukupno ima uzoraka? Koliko ima obeležja i kog su tipa?
2. Proveriti da li ima nedostajućih vrednosti i kako su definisane labele.
3. Proveriti koliko uzoraka ima u svakoj od klase. Da li su klase jednakozastupljene?
4. Utvrditi na osnovu statističkih parametara da li postoje razlike između klasa za data obeležja.
5. Definisati funkciju koja na osnovu matrice konfuzije računa tačnost klasifikatora po klasi i prosečnu tačnost.
6. Definisati funkciju koja na osnovu matrice konfuzije računa osetljivost klasifikatora po klasi, kao i prosečnu, odnosno makro osetljivost.
7. Na osnovu dostupnih uzoraka, upotrebom unakrsne validacije sa 3 podskupa, pronaći optimalne parametre za MLP klasifikator, oslanjajući se na jednu od mera uspešnosti. Koristiti mehanizam ranog zaustavljanja algoritma. Ispisati za svaku od iteracija unakrsne validacije odabranu meru uspešnosti i nacrtati krivu udela tačno klasifikovanih uzoraka validacionog skupa kroz epohe. Prikazati zbirnu matricu konfuzije i izračunati na osnovu nje odabranu meru uspešnosti. Koja mera je ovde adekvatna? Koji su parametri optimalni? Kojih je dimenzija dobijena matrica konfuzije?
8. Koristeći implementirane funkcije, proveriti tačnost i osetljivost po klasi i prosečno. Zašto je tačnost klasifikatora visoka, a osetljivost niska? Da li je osetljivost niska za sve klase? Zašto? Na koji način problem može biti prevaziđen?
9. Nebalansirani skupovi podataka često dovode do loših performansi klasifikatora. Dve najjednostavnije ideje za prevaziđenje ovog problema zasnivaju se na smanjenju broja uzoraka najzastupljenije klase i povećanju broja uzoraka manje zastupljenih klasa. Ove metode treba primenjivati oprezno. Zašto?

10. Napraviti novi skup podataka izbacujući svaki treći uzorak iz klase 'US'. Koliko sada ima uzoraka? Kakva je raspodela uzoraka po klasama?
11. Ponoviti unakrsnu validaciju koristeći novi skup podataka. Proveriti tačnost i osetljivost po klasi i u proseku. Da li se rezultat poboljšao? Kakve promene su najznačajnije?
12. Napraviti treći skup podataka dupliranjem svakog od uzoraka koji nije iz klase 'US'. Koliko sada ima uzoraka? Kakva je raspodela uzoraka po klasama?
13. Ponoviti unakrsnu validaciju koristeći novi skup podataka. Proveriti tačnost i osetljivost po klasi i u proseku. Uporediti rezultate na sva tri skupa. Na koji način je postignut najbolji rezultat? Zašto?
14. Koristeći funkciju `mfcc` iz `python_speech_features` biblioteke i funkciju `wav` iz `scipy.io.wavfile` učitati 10 audio snimaka datih u bazi i izdvojiti po 12 MFCC. Sve smestiti u novi `DataFrame`.
15. Obučiti klasifikatore na celokupnim dostupnim bazama i testirati na premljenim podacima. Kakvi su rezultati?

## Literatura

- [1] Christopher M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.
- [2] Vladimir Crnojević. *Prepoznavanje oblika za inženjere*. Tehnička edicija FTN, 2014.
- [3] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern classification*. John Wiley & Sons, 2012.
- [4] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Springer series in statistics New York, 2001.
- [5] Laura Igual and Santi Segui. *Introduction to Data Science*. Springer, 2017.
- [6] Gareth James, Daniela Witten, Trevor Hastie, et al. *An introduction to statistical learning*. Springer, 2013.
- [7] Wes McKinney. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. O'Reilly Media, Inc., 2012.
- [8] Sebastian Raschka and Vahid Mirjalili. *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing Ltd, 2019.
- [9] Joseph L. Schafer and John W. Graham. “Missing data: our view of the state of the art”. In: *Psychological methods* (2002).
- [10] Hiroshi Shimodaira. “Text classification using naive Bayes”. In: *Learning and Data Note* (2014).
- [11] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition, Fourth Edition*. 4th. USA: Academic Press, Inc., 2008.
- [12] John W. Tukey. *Exploratory data analysis*. Reading, Mass., 1977.

## Skupovi podataka

- [1] Tiago A. Almeida, Jose Maria G. Hidalgo, and Akebo Yamakami. “Contributions to the study of SMS spam filtering: new collection and results”. In: *Proceedings of the 11th ACM symposium on Document engineering*. 2011, pp. 259–262.
- [2] Tiago A. Almeida, Nick Street, and Olvi Mangasarian. *UCI Machine Learning Repository*. 2012. URL: <https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection>.
- [3] Davide Anguita. *UCI Machine Learning Repository*. 2012. URL: <https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones>.
- [4] Davide Anguita, Alessandro Ghio, Luca Oneto, et al. “A public domain dataset for human activity recognition using smartphones.” In: *Esann*. Vol. 3. 2013, p. 3.
- [5] University of Dayton. *Environmental Protection Agency Average Daily Temperature Archive*. 2020. URL: <http://academic.udayton.edu/kissock/http/Weather/default.htm>.
- [6] Ernest Fokoue. *UCI Machine Learning Repository*. 2020. URL: <https://archive.ics.uci.edu/ml/datasets/Speaker+Accent+Recognition>.
- [7] Waldemar W. Koczkodaj. *UCI Machine Learning Repository*. 2018. URL: <https://archive.ics.uci.edu/ml/datasets/Somerville+Happiness+Survey>.
- [8] Waldemar W. Koczkodaj, Tamar Kakiashvili, Agnieszka Szymańska, et al. “How to reduce the number of rating scale items without predictability loss?” In: *Scientometrics* 111.2 (2017), pp. 581–593.
- [9] Ronny Kohavi and Barry Becker. *UCI Machine Learning Repository*. 1996. URL: <https://archive.ics.uci.edu/ml/datasets/Adult>.
- [10] Wolberg William, Nick Street, and Olvi Mangasarian. *UCI Machine Learning Repository*. 1995. URL: [https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+\(Diagnostic\)](https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)).