Mašinsko učenje

Klasifikacija

Nemanja Maček

Sadržaj

Uvodne napomene

Da li ste do sada koristili klasifikaciju kao vid mašinskog učenja?

- Najverovatnije jeste, čak iako niste znali za to.
- Primer: sistem elektronske pošte ima mogućnost automatskog otkrivanja neželjene pošte.
  - To znači da će sistem analizirati sve dolazne poruke i označiti ih kao spam ili ne-spam.
  - Često, vi, kao krajnji korisnik, imate mogućnost da ručno označavate poruke kao neželjene, kako biste poboljšali sposobnost otkrivanja spama.
  - Ovo je oblik mašinskog učenja gde sistem uzima primere dve vrste poruka: spam i tzv. šunka (tipičan izraz za „ne-spam e-poštu“) i koristeći ove primere automatski klasifikuje dolazne e-poruke.

Uvodne napomene

Šta je klasifikacija?

- Korišćenjem primera iz istog domenskog problema koji pripadaju različitim klasama obučavamo model, odnosno „generišemo pravila“ koja se mogu primeniti na nove (prethodno nepoznate) primere.

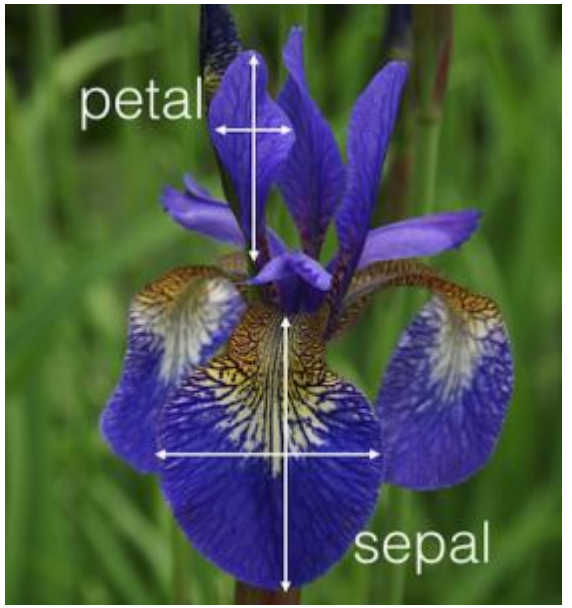
Iris skup podataka

O skupu podataka.

- Skup podataka Iris je klasičan skup podataka iz 1930-ih godina; to je jedan od prvih primera statističke klasifikacije.
- Skup podataka je kolekcija morfoloških merenja izvesne količine cvetova perunike.
- Ova merenja nam omogućavaju da identifikujemo različite vrste cveća.
- Danas se vrste identifikuju pomoću DNK, ali u 30-tim godinama uloga DNK u genetici još nije zabeležena.

Iris skup podataka

O skupu podataka.



- Četiri obeležja izdvojena su za svaku biljku (sl. desno):
  - *sepal length* (dužina čašičnog listića),
  - *sepal width* (širina čašičnog listića),
  - *petal length* (dužina latice) i
  - *petal width* (širina latice)
- U skupu postoje tri klase koje identifikuju biljke: Iris Setosa (sl. dole levo), Iris Versicolor (sl. dole u sredini) i Iris Virginica (sl. dole desno).

Iris skup podataka

Formulacija problema.

- Ovaj skup podataka ima četiri obeležja.
- Pored toga, za svaku biljku zabeležena je vrsta, odnosno vrednost klasnog obeležja.
- Problem koji želimo da rešimo je sledeći: imajući u vidu ove primere, da li možemo predvideti vrstu novog cveta na terenu na osnovu merenja?
- Ovo je problem klasifikacije, odnosno nadgledanog učenja: da li na osnovu označenih podataka možemo „generisati pravila“ koja se kasnije mogu primeniti na druge primere?
- Primeri poznatiji čitaocima koji ne izučavaju botaniku su:
  - filtriranje neželjene e-pošte,
  - detekcija upada u računarske sisteme i mreže,
  - detekcija prevare kreditnim karticama, itd.

Iris skup podataka

Vizualizacija podataka.

- Predstavićemo vrstu Setosa trouglom, vrstu Versicolor krugom i vrstu Virginica oznakom x.
- Najpre čitamo podatke, a zatim crtamo grafikone.

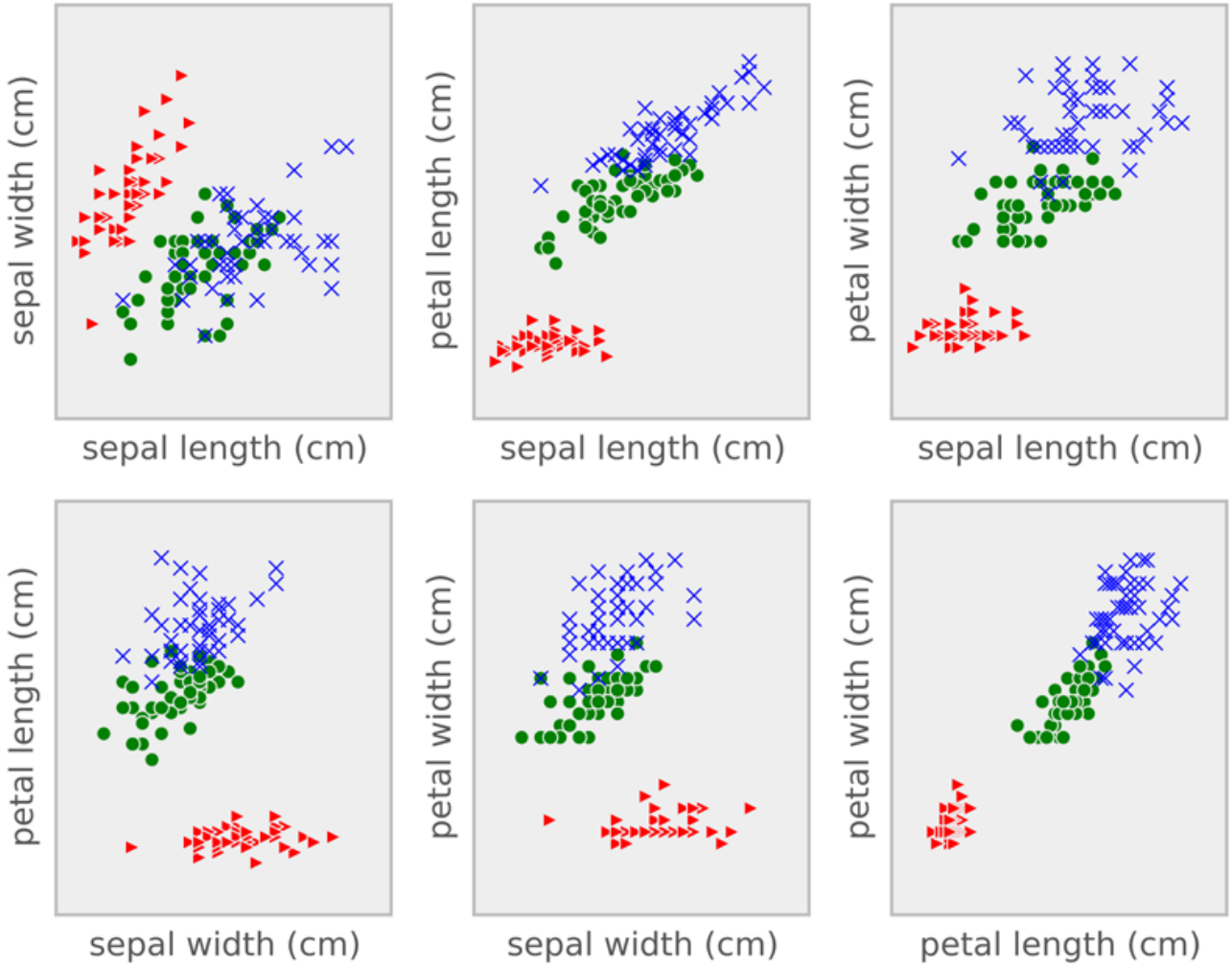
```
from matplotlib import pyplot as plt
import numpy as np
from sklearn.datasets import load_iris # čitamo podatke korišćenjem load_iris iz sklearn
data = load_iris() # load_iris vraća objekat koji sadrži vrednosti obeležja i klasa
features = data.data
feature_names = data.feature_names
target = data.target
target_names = data.target_names
```

Iris skup podataka

Vizualizacija podataka.

```
fig,axes = plt.subplots(2, 3) # po tri vizualicije u dva reda
pairs = [(0, 1), (0, 2), (0, 3), (1, 2), (1, 3), (2, 3)] # uparivanje obeležja
color_markers = [ ('r', '>'), ('g', 'o'), ('b', 'x'), ] # postavljanje boja i markera
for i, (p0, p1) in enumerate(pairs):
  ax = axes.flat[i]
  for t in range(3):
    c,marker = color_markers[t] # koristi različiti par boja/marker za svaku klasu
    ax.scatter(features[target == t, p0], features[target == t, p1], marker=marker, c=c)
  ax.set_xlabel(feature_names[p0])
  ax.set_ylabel(feature_names[p1])
  ax.set_xticks([])
  ax.set_yticks([])
fig.tight_layout()
fig.savefig('figurure1.png')
```

Iris skup podataka



Iris skup podataka

Jednostavan klasifikacioni model.

- Cilj je, kao što je već rečeno da razdvojimo tri vrste cveta.
- Analizom slike na str. 10 vidi se da obeležje *petal length* razdvaja vrstu Setosa od ostale dve klase.
- Šta činimo dalje? Tražimo granicu razdvajanja.

Iris skup podataka

Vizualizacija podataka.

```
# Iskoristićemo NumPy da kreiramo niz stringova:
>>> labels = target_names[target]
# Petal length je obeležje na poziciji 2
>>> plength = features[:, 2]
>>> is_setosa = (labels == 'setosa')
# Ovaj korak je bitan:
>>> max_setosa = plength[is_setosa].max()
>>> min_non_setosa = plength[~is_setosa].min()
>>> print('Maximum of setosa: {0}.'.format(max_setosa))
Maximum of setosa: 1.9.
>>> print('Minimum of others: {0}.'.format(min_non_setosa))
Minimum of others: 3.0.
```

Iris skup podataka

Jednostavan klasifikacioni model.

- Ovim smo napravili jednostavan model: ako je dužina latice manja od 2, onda je ovo cvet Irisa Setosa; inače je cvet biljke Iris Virginica ili Iris Versicolor.
- Ovo je naš prvi klasifikacioni model koji veoma dobro razdvaja cvetove Iris Setosa od druge dve vrste (bez ikakvih grešaka).
- Međutim, u ovom slučaju mi se zapravo nismo bavili mašinskim učenjem.
- Umesto toga, sami smo pogledali podatke, tražeći razdvajanje između klasa.
- Mašinskim učenjem se bavimo kada pišemo kod koji automatski traži ovo razdvajanje.

Iris skup podataka

Razdvajanje sa najmanjom greškom.

- Problem razdvajanja Irisa Setosa u odnosu na druge dve vrste bio je veoma lak.
- Međutim, ne možemo odmah videti šta je najbolji prag za razlikovanje cveta Iris Virginica od cveta Iris Versicolor.
- Čak možemo i da vidimo da nikada nećemo postići savršeno odvajanje sa ovim obeležjima.
- Možemo, međutim, tražiti najbolje moguće razdvajanje, odnosno razdvajanje koje čini najmanju grešku.

Iris skup podataka

Razdvajanje sa najmanjom greškom.

- Najpre odabiramo instance skupa koje ne pripadaju klasi Setosa.

```
>>> features = features[~is_setosa] # ~ je bulovski operator negacije
>>> labels = labels[~is_setosa]
>>> is_virginica = (labels == 'virginica') # kreira novu ciljnu promenljivu, is_virginica
```

- Nakon toga u petlji prolazimo sva moguća obeležja i pragove kako bi utvrdili koji će rezultovati najboljom tačnošću.
- Tačnost se grubo definiše kao odnos primera (instanci) koji model pravilno klasifikuje i ukupnog broja primera.

Iris skup podataka

Razdvajanje sa najmanjom greškom.

```
# postavi vrednost naveće tačnosti na najmanju moguću vrednost
best_acc = -1.0
# Prođi u petlji kroz sva obeležja
for fi in range(features.shape[1]):
  # Testiraj sve moguće vrednosti praga za obeležje fi
  thresh = features[:,fi].copy()
  thresh.sort()
  for t in thresh:
    # uzmi vektor obeležja fi
    feature_i = features[:, fi]
    # generiši predviđanje koristeći t kao prag
    pred = (feature_i > t)
    # tačnost je odnos ispravnih predviđanja
    acc = (pred == is_virginica).mean()
```

Iris skup podataka

Razdvajanje sa najmanjom greškom.

```
# Isprobaćemo da li poređenje tipa „veće“ ili „manje“ od praga daje bolji rezultat
rev_acc = (pred == ~is_virginica).mean()
if rev_acc > acc:
  reverse = True
  acc = rev_acc
else:
  reverse = False
# Ako je tačnost veća od prethodna, zapamti obeležje i prag.
if acc > best_acc:
  best_acc = acc
  best_fi = fi
  best_t = t
  best_reverse = reverse
```

Iris skup podataka

Razdvajanje sa najmanjom greškom.

- Promenljive `best_fi`, `best_t`, i `best_reverse` čine naš model.
- Te informacije su nam neophodne da bi klasifikovali nov, nepoznat objekat, odnosno da bi smo mu dodelili vrednost klasnog obeležja na sledeći način:

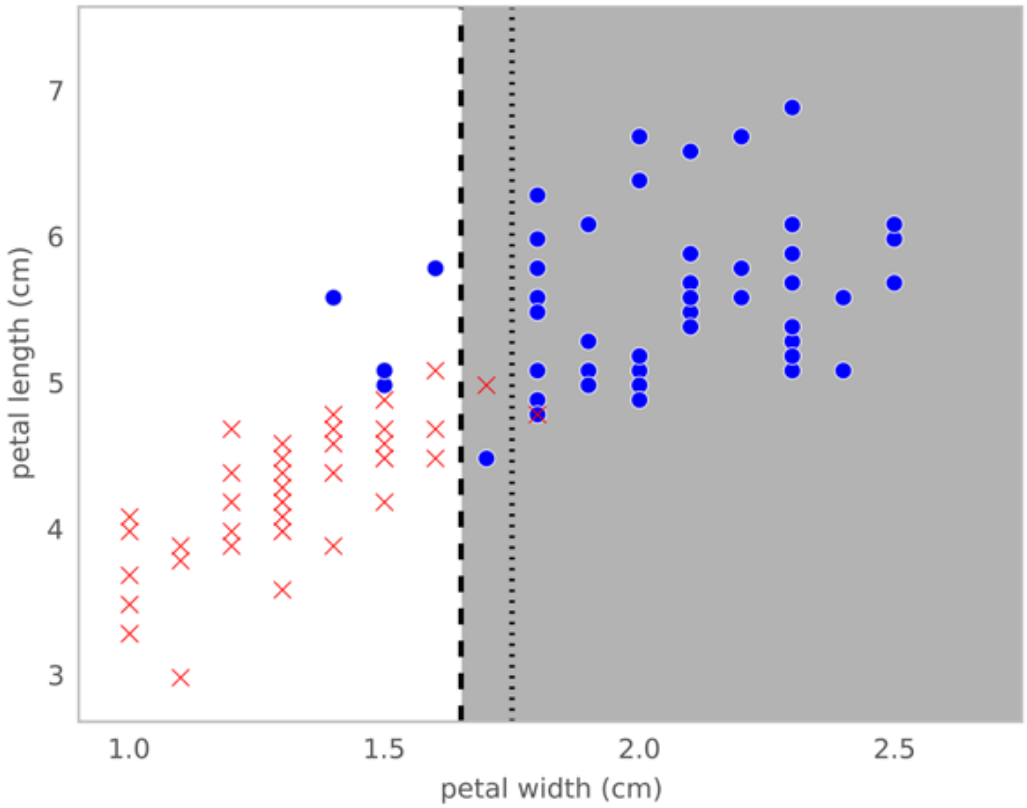
```
def is_virginica_test(fi, t, reverse, example):
  "Apply threshold model to a new example"
  test = example[fi] > t
  if reverse:
    test = not test
  return test
```

Iris skup podataka

Razdvajanje sa najmanjom greškom.

- Kako izgleda ovaj model?
- Ako pokrenemo kod na celokupnim podacima, model koji je identifikovan kao najbolji donosi odluke na osnovu širinu latice.
- Jedan način da se stekne uvid u funkcionisanje modela je vizualizacija granice odluke.
- Na str 20. ekranu vidimo dva regiona: beo i siv (osenčen).
- Bilo koja tačka koja pripadne belom regionu biće klasifikovana kao Iris Virginica, dok će svaka tačka koja pripadne osenčenoj strani biti klasifikovana kao Iris Versicolor.
- Kod koji iscrtava model dat je u dodatku knjige, datoteka `ch02/figure2.py`.

Iris skup podataka



Iris skup podataka

Ocena modela.

- Model koji je do sada razmatran je jednostavan model koji postiže 94% tačnosti na celom skupu podataka.
- Međutim, ova procena može biti previše optimistična: podatke koje smo koristili da definišemo šta će biti prag smo zatim koristili da ocenimo model.
- Ono što stvarno želimo da uradimo je procena sposobnosti generalizacije modela.
- Drugim rečima, treba izmeriti performanse modela u slučajevima da algoritam klasifikuje podatke kojima nije obučavan.
- Napravićemo strožiju procenu i koristiti „odložene“ (engl. *held-out*) podatke.
- Podatke ćemo razdvojiti tako što ćemo jedan deo podataka iskoristiti da obučimo model, a drugi deo da ga testiramo, odnosno ocenimo.
- Kompletan kod dostupan na str. 22 i 23 i u dodatku knjige, datoteka `ch02/heldout.py`.

Iris skup podataka

Ocena modela.

```
import numpy as np
from sklearn.datasets import load_iris
from threshold import fit_model, accuracy
data = load_iris()
features = data['data']
labels = data['target_names'][data['target']]
# Uklanjamo setosa primere:
is_setosa = (labels == 'setosa')
features = features[~is_setosa]
labels = labels[~is_setosa]
# Klasifikujemo - Virginica ili Versicolor
is_virginica = (labels == 'virginica')
```

Iris skup podataka

Ocena modela.

```
# Podeli podatke na trening i test podatke
testing = np.tile([True, False], 50)  # testing = [True,False,True,False,True,False...]
training = ~testing # trening skup sadrži ono što nije u test skupu
model = fit_model(features[training], is_virginica[training])
train_accuracy = accuracy(features[training], is_virginica[training], model)
test_accuracy = accuracy(features[testing], is_virginica[testing], model)
print('''\
Training accuracy was {0:.1%}.
Testing accuracy was {1:.1%} (N = {2}).
'''.format(train_accuracy, test_accuracy, testing.sum()))
```

Iris skup podataka

Ocena modela.

- Šta dobijamo ako pokrenemo prethodni kod?

```
Training accuracy was 96.0%.
Testing accuracy was 90.0% (N = 50).
```

Iris skup podataka

Ocena modela.

- Šta se zapravo dogodilo?
- Tačnost u slučaju skupa obučavajućih podataka (koji je podskup svih podataka) očigledno je veća nego ranije.
- Međutim, tačnost nad test podacima je niža!
- Iako ovo može iznenaditi neiskusnu osobu koja se bavi mašinskim učenjem, očekuje se da će tačnost testiranja biti niža od tačnosti treninga.
- Osvrnućemo se na grafik koji je pokazao granicu odluke (str. 20).
- Razmotrite šta bi se desilo sa granicom odluke ukoliko neki od primera blizu granice nisu bili tamo prilikom obuke? Lako je zaključiti da će se granica pomeriti malo udesno ili ulevo.

Iris skup podataka

Ocena modela.

- NAPOMENE:
  - U ovom primeru razlika između tačnosti merenih na podacima za obuku i testiranje nije velika. Kada koristite složeni model, moguće je dobiti 100% tačnost u obuci i vrlo nisku tačnost pri testiranju!
  - Drugim rečima, tačnost na obučavajućem skupu je isuviše optimistična procena koliko je Vaš algoritam dobar. U eksperimentima se uvek meri i prijavljuje tačnost testiranja, odnosno tačnost na na skupu primera koji se nisu koristili za obuku!

Iris skup podataka

Ocena modela.

- Mogući problem sa *hold-out* validacijom je u tome da smo samo polovinu podataka koristili za obuku. Međutim, ukoliko isuviše podataka iskoristimo za obuku, procena greške testiranja se vrši na vrlo malom broju primera.
- U idealnom slučaju, mi bi koristili sve podatke i za obuku i za testiranje, ali je to nemoguće.
- Dobra aproksimacija ovog nemogućeg ideala je metoda koja se zove unakrsna validacija (engl. *cross-validation*).
- Najjednostavnija forma unakrsne validacije je izostavi-jedan unakrsna validacija (engl. *leave-one out cross-validation*).
  - Iz trening podataka se iskjljučuje jedan primer, obučava se model i proverava da li je pravilno klasifikovao isključeni primer.
  - Ovaj proces se zatim ponavlja za sve elemente u skupu podataka.

Iris skup podataka

Ocena modela.

```
>>> correct = 0.0
>>> for ei in range(len(features)):
      # odaberi sve sem primera na poziciji `ei`:
      training = np.ones(len(features), bool)
      training[ei] = False
      testing = ~training
      model = fit_model(features[training], is_virginica[training])
      predictions = predict(model, features[testing])
      correct += np.sum(predictions == is_virginica[testing])
>>> acc = correct/float(len(features))
>>> print('Accuracy: {0:.1%}'.format(acc))
Accuracy: 87.0%
```

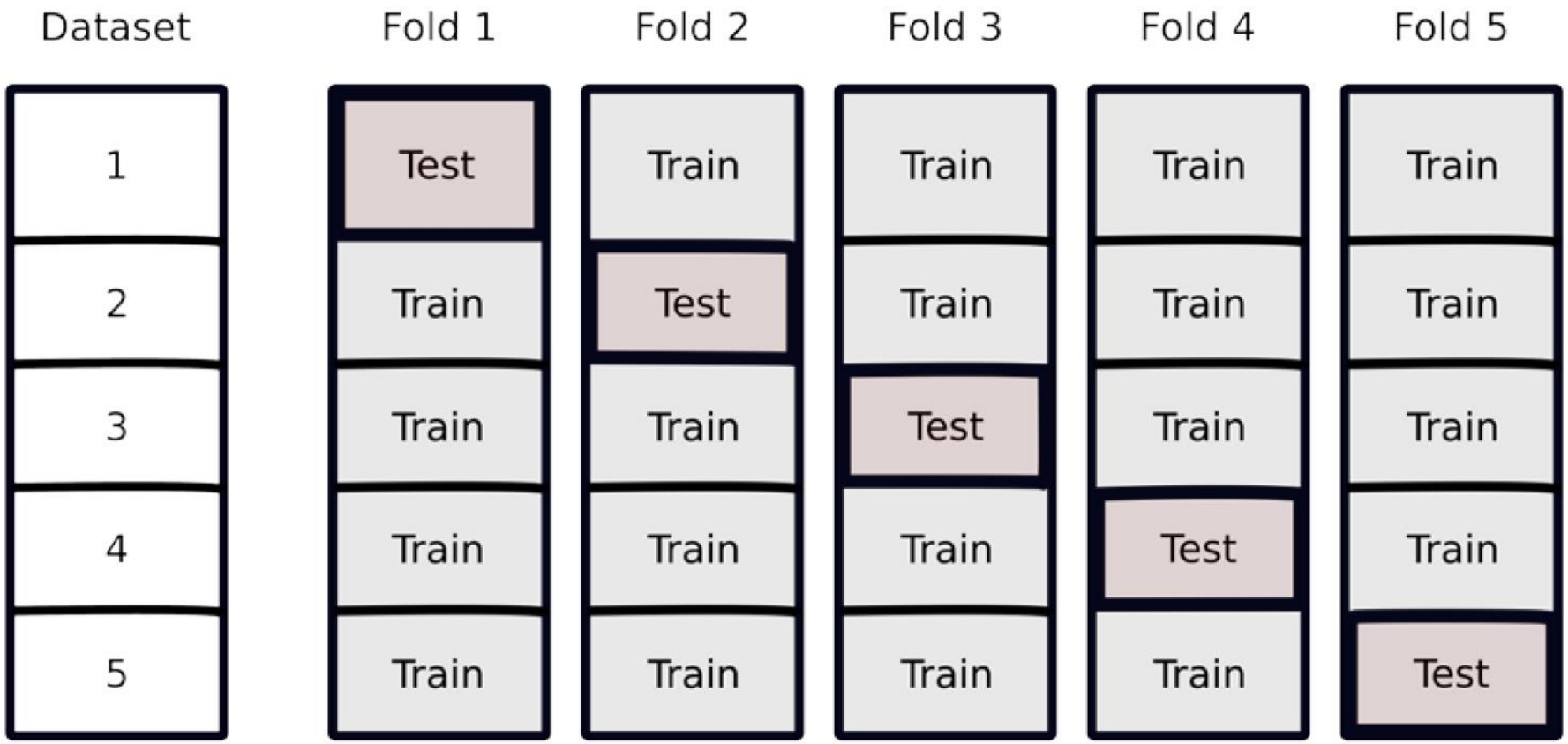
Iris skup podataka

Ocena modela.

- Kada se koristi izostavi-jedan unakrsna validacija, svaki primer je testiran na modelu obučenom potpunim skupom iz kog je izostavljen taj primer.
- Dakle, unakrsna validacija je pouzdana procena mogućnosti generalizacije modela.
- Osnovni problem sa prethodnom metodom validacije je potreba za velikim brojem obučavanja (broj raste sa veličinom skupa).
- Zato se koristi takozvana k-tostruka (engl. *k-fold*) unakrsa validacija.
- Ukoliko, na primer, koristimo 5-struku unakrsnu validaciju, podatke delimo na pet delova, od kojih u svakoj iteraciji 4 dela koristimo za obuku, a jedan za testiranje (v. str. 30).
- Broj delova na koje početni skup deli zavisi od veličine skupa, vremena potrebnog za obuku modela itd.
- Prilikom generisanja delova, jako je bitno da budu balansirani.

Iris skup podataka

Ocena modela.



Iris skup podataka

Šta su lažno negativni i lažno pozitivni rezultati?

- Primer iz medicine:
  - Lažno negativno (engl. *false negative*) – rezultat nekog testa je negativan, ali to zapravo nije tačno; to može dovesti do toga da pacijent ne primi terapiju za ozbiljnu bolest.
  - Lažno pozitivno (engl. *false positive*) – rezultat testa je pozitivan iako pacijent zapravo nema tu bolest; to može dovesti do dodatnih testova za potvrđivanje bolesti ili nepotrebnog lečenja (što može za posledicu imati dodatne troškove ili neželjene efekte lečenja).
- U praksi se ponekad mora pravi kompromis između učestalosti lažno pozitivnih i lažno negativnih rezultata.
- Šta su primeri lažno pozitivnih i lažno negativnih rezultata za:
  - filtre neželjene elektronske pošte,
  - sisteme za detekciju upada u računarske mreže, i
  - sisteme za detekciju prevare kreditnim karticama?

Iris skup podataka

Ka složenijim klasifikatorima.

- Model koji klasifikuje na osnovu vrednosti jednog obeležja i praga je jednostavan.
- Postoji veliki broj složenijih modela.
- Kako odabrati pravi? Potrebno je obratiti pažnju na:
  - strukturu modela – kako model donosi odluke,
  - proceduru pretrage – kako ćemo odrediti parametre modela,
  - funkciju gubitka ili dobiti – optimizacija modela na takav način da pravi manje grešaka tipa lažno pozitivnih ili lažno negativnih rezultata.

Seeds skup podataka i metoda najbližeg suseda

O skupu podataka.

- Analiziraćemo još jedan poljoprivredni skup podataka – seeds – koji je i dalje mali, ali ipak prevelik za vizualizaciju podataka, odnosno iscrtavanje koje smo učinili sa Irisom.
- Ovaj skup podataka sastoji se od merenja semena pšenice.
- Skup podataka sadrži sedam kontinualnih obeležja koja ne sadrže nepostojeće vrednosti:
  - površina A,
  - perimetar P,
  - kompaktnost $C = 4\pi A/P^2$,
  - dužinu jezgra,
  - širinu jezgra,
  - koeficijent asimetrije i
  - dužinu žlebova jezgra.

Seeds skup podataka i metoda najbližeg suseda

O skupu podataka.

- Postoje tri klase, koje odgovaraju sortama pšenice: Canadian, Koma i Rosa.
- Kao i u prethodnom primeru, cilj je da se klasifikuje vrsta biljke na osnovu ovih morfoloških merenja.
- Za razliku od skupa podataka cveta perunike, koji je prikupljen u 1930-tim, ovo je noviji skup podataka i njegova obeležja su automatski izračunata na osnovu digitalnih slika.

- Napomena: *The University of California at Irvine* (UCI) održava on-line repozitorijum skupova podataka za mašinsko učenje. Oba skupa (iris i seeds) korišćena u ovim belškama su preuzeta odatle. Repozitorijum je dostupan na adresi: http://archive.ics.uci.edu/ml/.

Seeds skup podataka i metoda najbližeg suseda

Inženjering obeležja.

- Ukoliko malo bolje pogledate obeležja, primetićete da kompaktnost nije nezavisno obeležje, već funkcija površine i perimetra.
- U praksi je često korisno generisati nova kombinovana obeležja.
- Pokušaj stvaranja novih obeležja se naziva inženjeringom obeležja (engl. *feature engineering*).
- Ova aktivnost utiče na performanse klasifikatora – jednostavan algoritam će postići bolje performanse na dobro definisanim obeležjima od fantastičnog algoritma na ne-tako dobro definisanim obeležjima.
- Da bi ste izgenerisli dobra obeležja, potrebno je znanje o konkretnom problemskom domenu koji rešavate. Drugim rečima, čak i pre nego što započnete prikupljanje podataka morate da znate koje podatke je vredno sakupljati a koji su manje korisni ili beskorisni.
- Na sreću, za veliki broj problemskih domena postoji dovoljno literature u kojoj su opisana obeležja i tipovi obeležja pogodni za rešavanje datog problema.

Seeds skup podataka i metoda najbližeg suseda

Izbor obeležja.

- Prirodno je postaviti pitanje da li možemo automatski odabrati dobra obeležja.
- Ovaj problem je poznat kao izbor obeležja (engl. *feature selection*).
- Postoje mnoge metode koje su predložene za rešenje ovog problema, ali u praksi vrlo jednostavne ideje najbolje funkcionišu.
- Za jednostavne probleme koje trenutno rešavamo nema potrebe odbacivati neka obeležja.
- U slučajevima kada imamo hiljade obeležja, odbacivanje većine njih koje imaju mali ili neznačajan utisak na klasifikaciju može učiniti ostatak procesa mnogo bržim.

Seeds skup podataka i metoda najbližeg suseda

Metoda najbližeg suseda.

- Ovaj metod se naziva još i metod zasnovan na memoriji ili instancama.
- Pripada porodici algoritama učenja koji umesto izvođenja eksplicitne generalizacije upoređuje nove instance sa instancama koje se nakon trening faze nalaze u memoriji.
- Učenje zasnovano na instancama je vrsta lenjog učenja.
  - Lenjo učenje u veštačkoj inteligenciji je poželjna osobina učenja ako postoji zahtev za stalnim menjanjem baze znanja, gde svaka takva promena ne povlači ponavljanje celog postupka učenja već samo efikasno inkrementalno dodavanje znanja.
- Zajednička karakteristika ovih metoda je da sve one smeštaju trening instance u memorijske strukture i koriste ih direkrno za klasifikaciju.
  - Najednostavniji oblik memorijske strukture je višedimenzionalni prostor definisan obeležjima.
  - Svaka trening instanca je reprezentovana kao tačka u prostoru.

Seeds skup podataka i metoda najbližeg suseda

Metoda najbližeg suseda.

- Klasifikacija novih instanci se obavlja prema principu najbližeg suseda, gde se nova instanca upoređuje sa instancama iz skupa za učenje korišćenjem definisane metrike.
- Metrika definiše rastojanje instanci na osnovu vrednosti njihovih obeležja, a odgovara intuitivnom shvatanju sličnosti instanci tako da ako su instance sličnije rastojanje je manje.
- Nova instanca se klasifikuje na osnovu pretraživanja skupa za učenje sa ciljem pronalaženja instance koja mu je u smislu rastojanja najbliža.

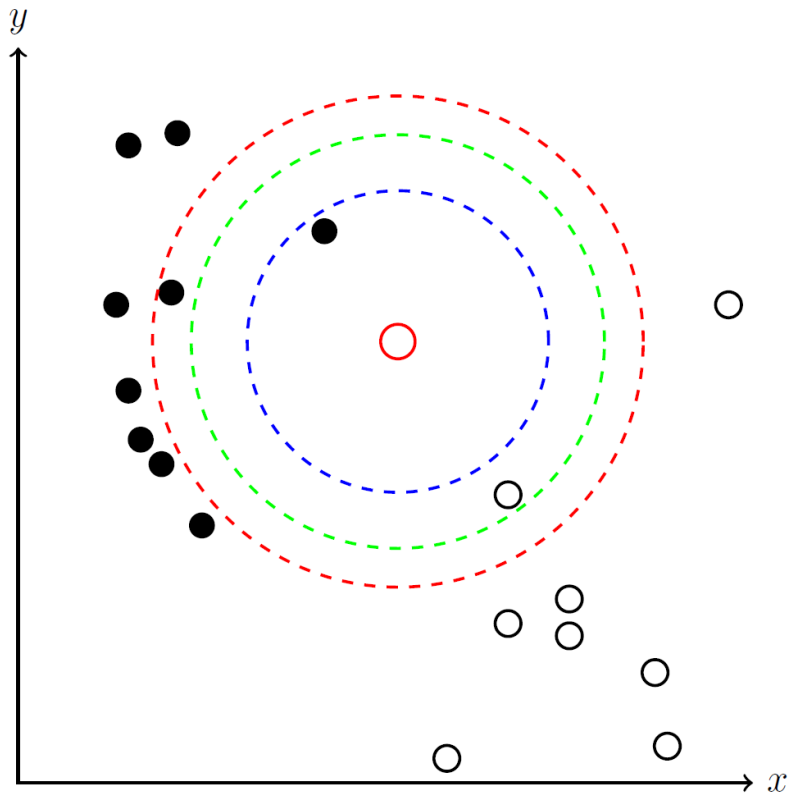
Seeds skup podataka i metoda najbližeg suseda

Metoda najbližeg suseda.

- **Trening**:
  - Smeštanje trening instanci.
- **Klasifikacija**:
  - Konkretna instanca x sadrži k najbližih suseda među instancama u trening skupu.
  - Najveći broj suseda iste klase određuje klasu instance.
- Kao što se može primetiti, trening faza ne postoji u običajenom smislu.
- Cena ovoga je sporija procedura odlučivanja jer u cilju klasifikovanja jedne poruke moramo računati rastojanja do svih trening poruka i pronaći k najbližih suseda.

Seeds skup podataka i metoda najbližeg suseda

Metoda najbližeg suseda. Ilustracija algoritma za *k*=1, *k*=2 i *k*=3.



** Izvor slike: A. Nedeljković (2015): Implementacija i evaluacija algoritama mašinskog učenja za filtriranje neželjene elektronske pošte. Matematički fakultet, Univerzitet u Beogradu.*

.

Seeds skup podataka i metoda najbližeg suseda

Metoda najbližeg suseda.

- Kriterijumi prihvatanja i eleminisanja instanci uglavnom pretstavljaju nepovratne strategije pohlepnog (engl. *greedy*) karaktera.
- Najjednostavniji kriterijum je da se ispituje instanca prema rezultatu klasifikacije korišćenjem do tada izdvojenih reprezentativnih instanci.
  - Ako je u pitanju netačna klasifikacija instance ona se pridodaje u skup reprezentativnih instanci jer je evidentno da menja granice klasa.
  - Ukoliko je klasifikacija instance tačna tada se ona proglašava suvišnom jer informacija koju nosi je već sadržana u skupu pomoću kojeg je klasifikovana.
- Nedostaci ovog kriterijuma za izbor instanci:
  - U početnoj fazi procesa pretraživanja postoji nezanemariva verovatnoća odbacivanja instanci koje se mogu pokazati važnim za tačnost klasifikacije rezultujućeg modela.
  - Izabrani podskup reprezentativnih instanci ne zavisi samo od polaznog skupa, već i od redosleda evaluacije instanci.

Seeds skup podataka i metoda najbližeg suseda

Metoda najbližeg suseda.

- Pored izbora instanci za pamćenje, na oblik klasifikacionog modela se može uticati i modifikacijom funkcije rastojanja.
- Jednak uticaj svih obeležja u instanci na konačan rezultat je jedno od svojstava euklidskog rastojanja, ali su u praksi retki problemi kod kojih sva obeležja imaju jednak značaj za proces klasifikacije.
- Modifikacija euklidskog rastojanja podrazumeva uvođenje težinskih vrednosti obeležja w_i.
- Ako sa w_i označimo težinsku vrednost pridruženu obeležju A_i, onda modifikovano Euklidsko rastojanje instanci x i y možemo predstaviti na sledeći način:

$$d_w(x, y) = \sqrt{\sum_{i=1}^{n} w_i^2 (x_i - y_i)^2}$$

Seeds skup podataka i metoda najbližeg suseda

Upotreba scikit-learn.

- Do sada smo koristili ručno rađeni kod za klasifikaciju.
- Međutim, zbog svojih biblioteka, Python je odgovarajući jezik za mašinsko učenje.
- Biblioteka scikit-learn postala je standardna biblioteka za mnoge zadatke mašinskog učenja, uključujući klasifikaciju.
- U nastavku izlaganja ćemo koristiti primenu metode najbližeg susjeda.

Seeds skup podataka i metoda najbližeg suseda

Upotreba scikit-learn.

- Dve osnove metode scikit-learn za klasifikaciju su:
  - `fit (features, labels)` – ovo je korak učenja koji kreira parametre modela.
  - `predict (features)` – ova metoda se može pozvati samo nakon učenja i vraća predviđanje za jedan ili više ulaza.
- Primenu najbližih suseda za naše podatke počinjemo uvozom objekta `KneighborsClassifier` iz podmodula sklearn.neighbors:

```
>>> from sklearn.neighbours import KNeighborsClassifier
```

- Modul scikit-learn je uvezen kao sklearn (ponekada ćete otkriti da se scikit-learn pominje pomoću ovog kratkog imena umesto punog imena). Sve funkcionalnosti sklearn modula su u podmodulima, kao što su sklearn.neighbours.

Seeds skup podataka i metoda najbližeg suseda

Upotreba scikit-learn.

- Sada možemo da napravimo instancu objekta klasifikatora.
- U konstruktoru određujemo broj suseda koje treba razmotriti:

```
>>> classifier = KNeighborsClassifier (n_neighbors = 1)
```

- Ako ne odredimo broj suseda, podrazumeva se 5, što ponekad može da bude dobar izbor za klasifikaciju.
- Želimo da koristimo unakrsnu validaciju da pogledamo naše podatke (v. str 46.)

Seeds skup podataka i metoda najbližeg suseda

Upotreba scikit-learn.

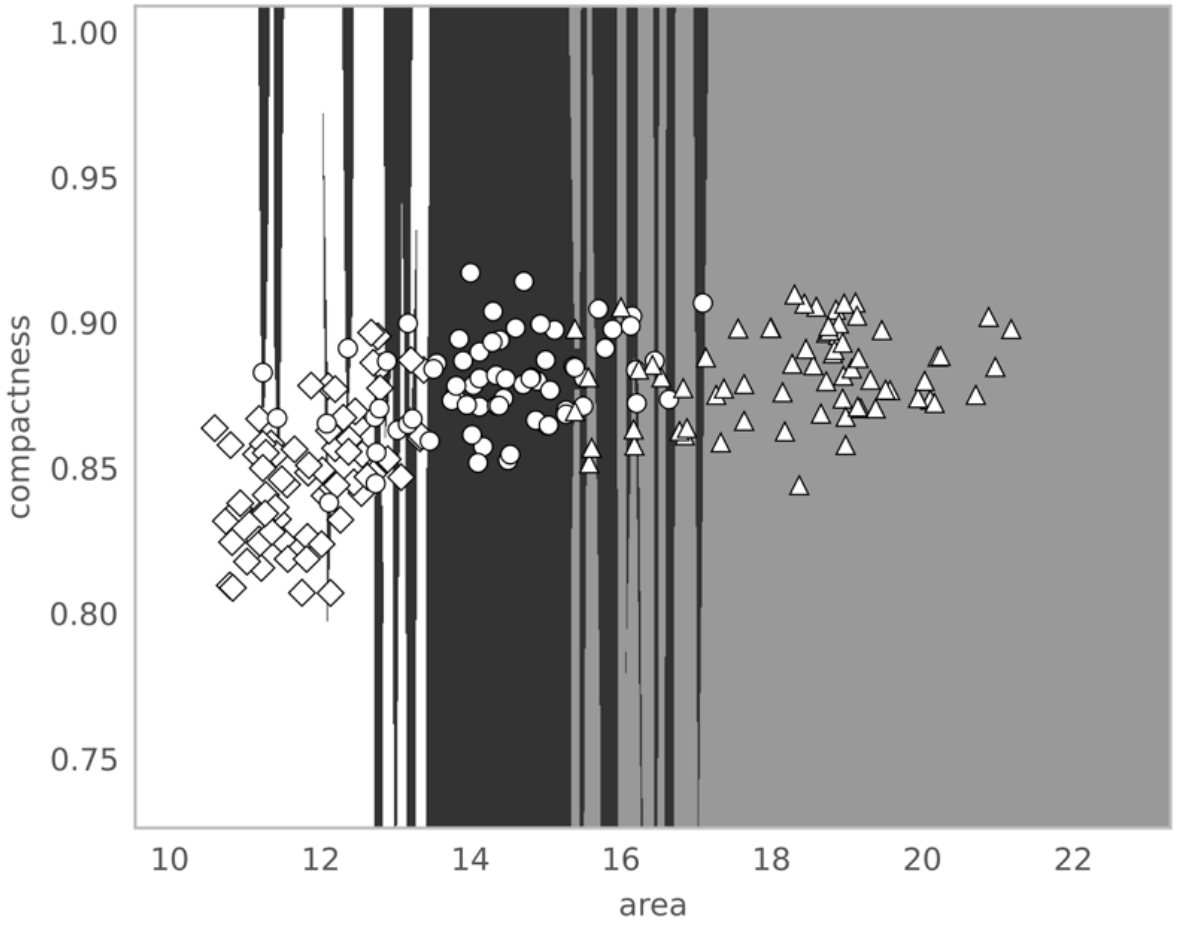
```
>>> from sklearn.cross_validation import KFold
>>> kf = KFold(len(features), n_folds=5, shuffle=True)
>>> means = []
>>> for training,testing in kf:
    # Prvo obučavamo model metodom fit, zatim ga testiramo metodom predict
    classifier.fit(features[training], labels[training])
    prediction = classifier.predict(features[testing])
    curmean = np.mean(prediction == labels[testing])
    means.append(curmean)
>>> print("Mean accuracy: {:.1%}".format(np.mean(means)))
Mean accuracy: 90.5%
```

Seeds skup podataka i metoda najbližeg suseda

Upotreba scikit-learn.

- Koristeći 5-struku unakrsnu validaciju, za ovaj skup podataka, sa ovim algoritmom, dobijamo 90.5% tačnosti.
- Kao što smo već pomenuli, preciznost unakrsne validacije je niža od tačnosti treninga, ali je ovo verodostojnija procena performansi modela.
- Sada ćemo ispitati granicu odluke. Da bismo ovo vizualizovali, pojednostavićemo slučaj i analizirati samo dva obeležja: area i compactness.
- Pogledajte sledeći grafik (v. str. 48), generisan kodom iz knjige (v. direktorijum 02 u dodatku knjige); primeri tipa Canadian su predstavljeni kao dijamanti, Koma kao krugovi a Rosa semena kao trouglovi. Njihove površine su bela, crna i siva, respektivno.

Seeds skup podataka i metoda najbližeg suseda



Seeds skup podataka i metoda najbližeg suseda

Da li primećujete nešto čudno na grafikonu?

- Možemo da primetimo da su regioni podeljeni skoro vertikalnim linijama i postavimo pitanje zašto grafik tako izgleda.
- Vrednosti na *x* osi (area) se kreću u opsegu od 10 do 22, dok se vrednosti na *y* osi kreću u opsegu od 0,75 do 1. Ovo potiče od mernih jedinica kojima su opisane vrednosti obeležja.
- To znači da je mala promena na jednoj osi zapravo mnogo veća od male promene na drugoj osi.
- Drugim rečima, kada računamo rastojanje između tačaka, u najvećem broju slučajeva uzimamo samo *x* osu.
- Ovo je takođe dobar primer zašto je dobra ideja da vizualizujemo naše podatke i proverimo da li postoje „iznenađenja“.

Seeds skup podataka i metoda najbližeg suseda

Kako možemo da rešimo problem?

- Potrebno je da normalizujemo sva obeležja.
- Ovo se može rešiti na više načina, a jedan od mogućih je *z-score* normalizacija.
- Neka je f stara vrednost obeležja, f' nova vrednost, μ srednja vrednost, a σ standardna devijacija, pri čemu se vrednosti μ i σ procenjuju na osnovu podataka za obučavanje.
- Normalizacija se obavlja na sledeći način:

$$f' = \frac{f - \mu}{\sigma}$$

- Nakon *z-score* normalizacije:
  - vrednost $f' = 0$ odgovara vrednosti $f = \mu$,
  - negativne vrednosti f' odgovaraju vrednostima manjim od srednje vrednosti μ,
  - pozitivne vrednosti f' odgovaraju vrednostima većim od srednje vrednosti μ.

Seeds skup podataka i metoda najbližeg suseda

Kako možemo da rešimo problem?

- Upotrebom modula scikit-learn normalizacija se jednostavno izvodi kao korak u predprocesiranju podataka.
- Koristićemo *pipeline* u kome je prvi element normalizacija a drugi klasifikacija.
- Da bi smo to izveli, uvozimo klase za rad sa *pipeline*-om i skaliranjem obeležja na sledeći način:

```
>>> from sklearn.pipeline import Pipeline
>>> from sklearn.preprocessing import StandardScaler
```

- Dalje radimo sledeće:

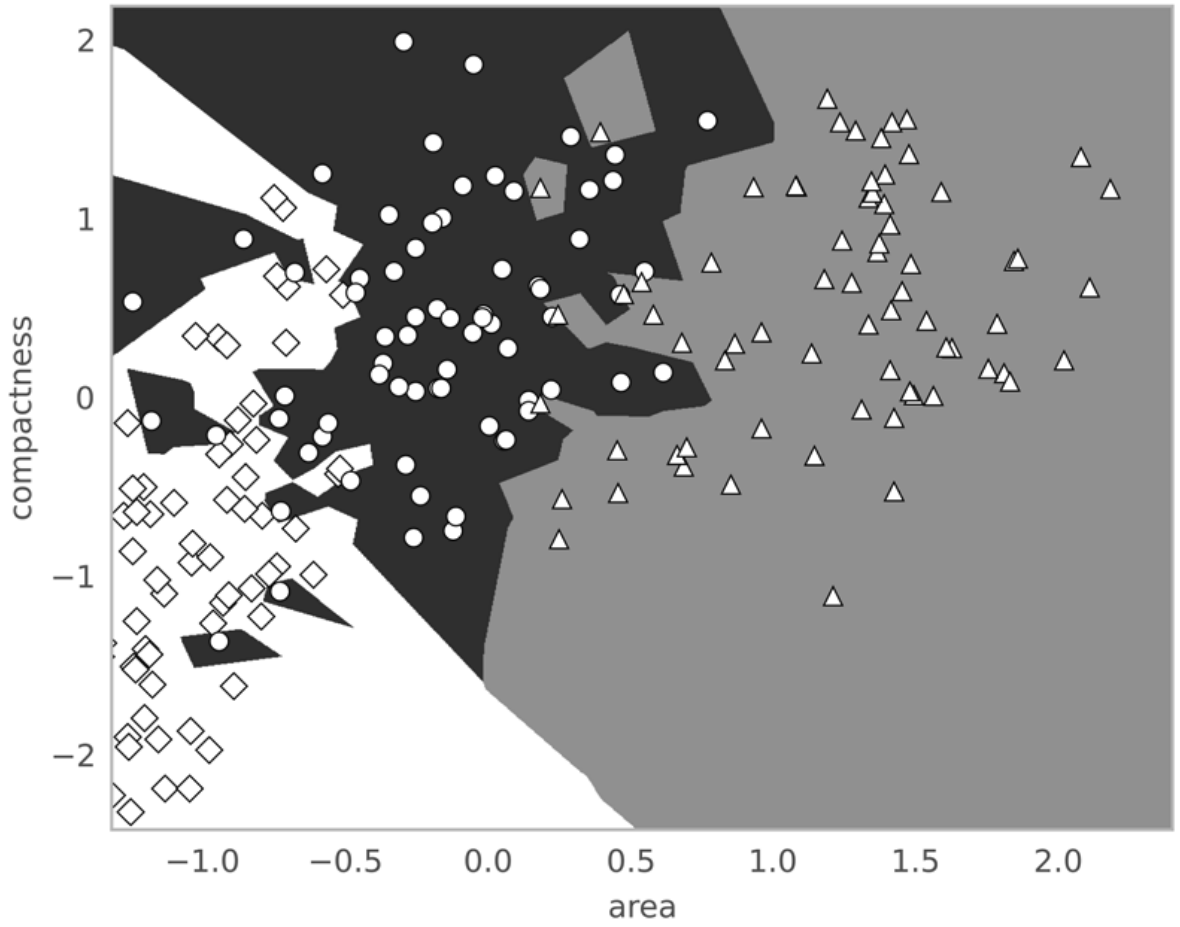
```
>>> classifier = KNeighborsClassifier(n_neighbors=1)
>>> classifier = Pipeline([('norm', StandardScaler()),  ('knn', classifier)])
```

Seeds skup podataka i metoda najbližeg suseda

Kako možemo da rešimo problem?

- Konstruktor *pipeline*-a zahteva listu parova tipa `(str, clf)`.
- Svaki par odgovara jednom koraku u *pipeline-u*, pri čemu je prvi element string koji imenuje korak, dok je drugi element objekat koji izvršava transformaciju.
- Nakon normalizacije, svako obeležje je u istim mernim jedinicama (tehnički, svako obeležje je sada bez dimenzija, jer nema mernu jedinicu).
- Ukoliko sada pokrenemo klasifikator zasnovan na metodi najbližeg suseda, dobićemo 93% tačnosti 5-struke unakrsne validacije sa kodom koji je prethodno razmotren.
- Granice na grafiku (v. str. 53) su sada različite i možete videti da oba obeležja utiču na ishod.
- U kompletnom skupu podataka, sve se dešava na sedmo-dimenzionalnom prostoru, koji je teško vizualizovati, ali se primenjuje isti princip: nekoliko obeležja dominira u originalnim podacima, a nakon normalizacije sva obeležja imaju isti značaj.

Seeds skup podataka i metoda najbližeg suseda



Binarna i višeklasna klasifikacija

Kako možemo da rešimo problem?

- Klasifikatori koje smo koristili za Iris skup podataka su binarni.
- Klasifikatori koje smo koristili za Seeds skup podataka su višeklasni.
- Višeklasni problem se mogu rešiti serijom binarnih odluka.
- Primer:
  - (1) Da li je u pitanju Iris Setosa (da ili ne)?
  - (2) Ako nije, da li je u pitanju Iris Virginica (da ili ne)?

Zaključne napomene

1. Greška na podacima za obuku nije merodavna. Uvek treba proceniti sposobnost generalizacije modela, odnosno izmeriti tačnost i greške u slučajevima da algoritam klasifikuje podatke kojima nije obučavan, odnosno izmeriti tačnost validacije.
2. Iako je tačnost vrlo bitna, u praksi se modeli ponekad prilagođavaju tako da proizvode manji broj grešaka konkretnog tipa (lažno negativno ili lažno pozitivno klasifikovani primeri).
3. Obeležja najčešće nisu predefinisana za vas. Izbor i dizajniranje obeležja je sastavni deo mašinskog učenja. Inženjering obeležja je vrlo često proces koji može dovesti do značajnog povećanja tačnosti.

- Ovo izlaganje je bilo pretežno teorijskog tipa i za cilj je imalo da shvatite osnovne koncepte i primenu tih koncepata na relativno jednostavnim skupovima podataka.

Literatura

- Beleške pripremljene prema knjizi – Luis Pedro Coelho, Willi Richert (2015): „Building Machine Learning Systems with Python, Second Edition“. Packt Publishing.

Hvala na pažnji

Pitanja su dobrodošla.