Mašinsko učenje

Otkrivanje loših odgovora

Nemanja Maček

Sadržaj

- Uvodne napomene
- Skup podataka
- Inženjering obeležja i formiranje klasifikatora
- Kako rešavamo problem niske tačnosti?
- Upotreba logističke regresije
- Preciznost i odziv
- Završne napomene

Uvodne napomene

U čemu je izazov za vlasnike Q&A sajtova?

- Stalni izazov za vlasnike tzv. Q&A („daj odgovor na pitanje“) sajtova je održavanje pristojnog nivoa kvaliteta objavljenog sadržaja – konkretno – ispravnih i zadovoljavajućih odgovora na postavljena pitanja.
- Sajtovi poput StackOverflow na specifičan način stimulišu korisnike da ulože više truda na formulisanje i davanje ispravnih i zadovoljavajućih odgovora na postavljena pitanja.
- Jedan od načina stimulacije je davanje mogućnosti osobi koja je postavila pitanje da odgovore na svoje pitanje označi kao prihvaćene (ispravne, zadovoljavajuće) ili ne, pri čemu se osoba koja postavlja pitanje na određeni način takođe stimuliše da označava odgovore na prethodno pomenut način.

Uvodne napomene

U čemu je izazov za vlasnike Q&A sajtova?

- Postavićemo sledeće pitanje: da li je izvodljivo napraviti sistem koji će omogućiti osobi koja odgovara na postavljena pitanja da „automatski“ dobije povratnu informaciju o kvalitetu odgovora?
- Drugim rečima, da li Web sajt može „automatski“ da vrednuje rad osobe koja je dala odgovor i obezbedi povratne informacije o tome da li odgovor konvergira ka lošem ili ne?
- Ovakav automatizam bi ohrabrio osobu koja odgovara na pitanja da uloži više truda u pisanju korektnih odgovora, i na taj način poboljšao sistem u celini.

Uvodne napomene

Šta ćemo raditi na ovom času?

- Pokušaćemo da izgradimo sistem koji na osnovu zašumljenih podataka može rešiti prethodno pomenuti problem automatskog vrednovanja odgovora.
- Naravno, ne možemo obučiti klasifikator koji će dostići 100% tačnosti – uzmite u obzir da se čak i ljudi međusobno ne slažu da li je odgovor zadovoljavajući, odnosno tačan ili ne.
- Klasifikatori koje ćemo koristiti u ovom izlaganju su metoda najbližih suseda, za koju ćemo pokazati da je nezadovoljavajuća u datom problemskom domenu, i logistička regresija.

Uvodne napomene

Šta je klasifikacija (neformalni podsetnik)?

- Klasifikacija se jednostavno može objasniti kao dodeljivanje vrednosti klasnog obeležja instancama podataka za koje je vrednost obeležja klase prethodno nepoznata.
- Da bi ovo izveli, potrebno je da odgovorimo na neka pitanja:
  - na koji način ćemo predstaviti instance podataka,
  - koji model i koje parametre ćemo koristiti u klasifikatoru, itd.

Uvodne napomene

- **Drugim rečima**, ...

- Klasifikacija je razvrstavanje nepoznate instance u jednu od unapred ponuđenih klasa.
  - Svaka instanca može se predstaviti skupom njenih obeležja.
  - Takođe, svakoj instanci se može dodati kao obeležje i oznaka klase kojoj instanca pripada.

sepallength	sepalwidth	petallength	petalwidth	class
5.3	3.7	1.5	0.2	Iris-setosa
5	3.3	1.4	0.2	Iris-setosa
7	3.2	4.7	1.4	Iris-versicolor
6.2	2.8	4.8	1.8	Iris-virginica

- Klasifikacija je odredivanje vrednosti obeležja klase na osnovu preostalih obeležja instance.

Uvodne napomene

Šta je, u ovom slučaju, instanca podataka a šta klasno obeležje?

- U ovom slučaju, instanca podataka je tekst odgovora.
- Klasno obeležje je binarna vrednost koja ukazuje na to da li je osoba koja je postavila pitanje prihvatila tekst kao ispravan, odnosno zadovoljavajući odgovor, ili ne.
- Kao što je rečeno (v. belešku 3), sa stanovišta mašinskog učenja, običan („sirov“) tekst je u najvećem broju slučajeva beskoristan – običan tekst je nepogodna reprezentacija podataka za algoritame za mašinsko učenje.
- Međutim, tekst koji je na neki način pretvoren u numeričke vrednosti od značaja je u nekim slučajevima upotrebljiv – drugim rečima, naš zadatak je da izvučemo upotrebljiva obeležja iz „sirovog“ teksta na osnovu kojih algoritam može da „nauči“ kako da tekstu dodeli odgovarajuće klasno obeležje.

Uvodne napomene

Odabir klasifikatora.

- Kada sakupimo dovoljno parova tipa (tekst, oznaka klase), možemo da obučimo klasifikator.
- Postoji veliki broj klasifikatora, od kojih svaki ima svoje prednosti i nedostatke.
- Neki od često korišćenih klasifikatora su stabla odlučivanja, veštačke neuronske mreže, metoda vektora oslonca, Naïve Bayes, itd.
- U ovom izlaganju ćemo izvršiti uporednu analizu metode najbližeg suseda i logističke regresije.

Uvodne napomene

Metode mašinskog učenja koje se koriste za klasifikaciju.

- Metode mašinskog učenja koje se koriste za klasifikaciju mogu se podeliti na:
  - osnovne (veštačke neuronske mreže, metoda vektora oslonca, stabla odlučivanja, Naïve Bayes, itd.),
  - hibridne (na primer, stablo odlučivanja na čijim se listovima nalazi Naïve Bayes klasifikator obučen instancama koje pripadaju tom listu),
  - inkrementalne (Naïve Bayes Updatable),
  - hibridne inkrementalne (Hoeffding Tree),
  - osnovne ensemble (Random Forest),
  - hibridne ensemble (slaganje),
  - hibridne inkrementalne ensemble (Ada Hoeffding Option Tree).

Skup podataka

Gde možemo pronaći skup podataka?

- Licenca sajta StackOverflow je cc-wiki, što znači da možemo da eksperimentišemo sa tim podacima.
- Podaci su dostupni na stranici https://archive.org/details/stackexchange; od značaja za naš eksperiment je datoteka stackoverflow.com-Posts.7z koja je veličine ≈11GB (zadnji put provereno 24.3.2018).

Skup podataka

Kako izgleda skup podataka?


```
<?xml version="1.0" encoding="utf-8"?>
<posts>
...
<row Id="4572748" PostTypeId="2" ParentId="4568987" CreationDate="2011-01-01T00:01:03.387" Score="4"
ViewCount="" Body="&lt;p&gt;IANAL, but &lt;a href=&quot;http://support.apple.com/kb/HT2931&quot;
rel=&quot;nofollow&quot;&gt;this&lt;/a&gt; indicates to me that you cannot use the loops in your
application:&lt;/p&gt;&#xA;&#xA;&lt;blockquote&gt;&#xA; &lt;p&gt;...however, individual audio loops may&#xA;
not be commercially or otherwise&#xA; distributed on a standalone basis, nor&#xA; may they be repackaged in
whole or in&#xA; part as audio samples, sound effects&#xA; or music beds.&quot;&lt;/p&gt;&#xA; &#xA;
&lt;p&gt;So don't worry, you can make&#xA; commercial music with GarageBand, you&#xA; just can't distribute
the loops as&#xA; loops.&lt;/p&gt;&#xA;&lt;/blockquote&gt;&#xA;" OwnerUserId="203568" LastActivityDate="2011-
01-01T00:01:03.387" CommentCount="1" />
…
</posts>
```

Skup podataka

Kako izgleda skup podataka?

Ime	Tip	Opis
Id	Integer	Jedinstveni identifikator.
PostTypeId	Integer	Kategorija – Q (pitanje) ili A (odgovor), ostalo ignorišemo.
ParentId	Integer	ID pitanja kome odgovor pripada, ne postoji za pitanja.
CreationDate	DateTime	Datum postovanja.
Score	Integer	„*Score*“ posta (tuđa procena tačnosti odgovora).
ViewCount	Int. or Empty	Broj korisničkih pregleda posta.
Body	String	Kompletni post kao HTML.
OwnerUserId	Id	ID korisnika koji je postovao; ako je 1, onda je to wiki pitanje.
Title	String	Naslov pitanja, ne postoji za odgovore.
AcceptedAnswerId	Id	ID prihvaćenog odgovora, ne postoji za pitanja.
CommentCount	Integer	Broj komentara za dati post.

Inženjering obeležja i formiranje klasifikatora

Osnovno filtriranje.

- Kako bi ubrzali eksperiment, performanse klasifikatora nećemo ispitivati na celoj XML datoteci.
- Podskup datoteke u kome je vrednost atributa CreationDate 2012 sadrži preko 6 miliona postova (oko 2,3 miliona pitanja i 4 miliona odgovora) – što je dovoljno za eksperiment.
- Osim toga, XML format je nepogodan za obradu, tako da ćemo odabrani podskup pretvoritu u TSV datoteku (*tab-separated value*).

Inženjering obeležja i formiranje klasifikatora

Analiza početnog skupa.

- Da bi još malo „ošišali“ naš skup podataka, odbacićemo neka obeležja koja nisu od značaja za klasifikaciju odgovora.
  - Atribut PostTypeId je neophodan za razdvajanje pitanja i odgovora – iako ga nećemo koristiti kao obeležje, neophodan je za filtriranje podataka.
  - Atribut CreationDate je zanimljiv zato što se na osnovu njega može odrediti interval između postovanja pitanja i odgovora, pa ćemo ga zadržati.
  - Atribut Score je od značaja zato što je to indikator tuđe procene tačnosti odgovora.
  - Atribut ViewCount nije od značaja i možemo da ga odbacimo.
  - Atribut Body sadrži informacije koje su za nas od najvećeg značaja. Pošto je u HTML formatu, moraćemo da ga pretvorimo u običan tekst.
  - Atribut OwnerUserID je koristan u slučaju da u obzir uzimamo korisnički-zavisna obeležja; pošto ih ovde ne uzimamo u obzir, atribut možemo da odbacimo.

Inženjering obeležja i formiranje klasifikatora

Analiza početnog skupa.

- Atribut Title takođe možemo da odbacimo, iako u sebi sadrži izvesnu količinu dodatnih informacija o pitanju.
- Atribut CommentCount, slično atributu ViewCount nije od značaja za klasifikator u trenutku kad je odgovor postovan i možemo takođe da ga odbacimo.
- Atribut AcceptedAnswerId je sličan atributu Score u tom smislu da predstavlja indikator kvalieta odgovora. Na osnovu ovog atributa formiraćemo obeležje IsAccepted, koje će imati vrednosti 0 ili 1 za odgovore, dok će za pitanja biti ignorisano.

- Nakon odabira i obrade obeležja dobićemo sledeće:

```
Id <TAB> ParentId <TAB> IsAccepted <TAB> TimeToAnswer <TAB> Score <TAB> Text
```

Inženjering obeležja i formiranje klasifikatora

Analiza početnog skupa.

- Za detalje o parsiranju, pogledajte kod dat u datotekama `so_xml_to_tsv.py` i `choose_instance.py` u dodatku knjige, direktorijum `ch05`.
- Da bi ubrzali obradu, podatke delimo u dve datoteke:
  - `meta.json`, koja sadrži identifikator posta i sve podatke osim teksta (npr., Score posta se nalazi u `meta[Id]['Score']`), i
  - `data.tsv` koja sadrži identifikator i Text posta, iz koje podatke možemo pročitati sledećom metodom:

```
def fetch_posts():
  for line in open("data.tsv", "r"):
    post_id, text = line.split("\t")
    yield int(post_id), text.strip()
```

Inženjering obeležja i formiranje klasifikatora

Šta je dobar odgovor?

- Pre nego što obučimo klasifikator, neophodno je da napravimo razliku između dobrih i loših odgovora, odnosno da kreiramo obučavajući skup – za sada imamo samo gomilu podataka, tako da moramo da formiramo klasna obeležja, tj. labele.
- Najjednostavnije, možemo da koristimo atribut IsAccepted kao oznaku klase.
  - Iako je ovo obeležje, na neki način, ocena odgovora, treba uzeti u obzir da je to samo mišljenje osobe koja je postavila pitanje.
  - Onaj ko postavlja pitanje želi brz odgovor i prihvatiće prvi koji mu se čini najboljim.
  - Međutim, uvek postoji mogućnost da će neko nakon izvesnog vremena postovati bolji odgovor od prihvaćenog, a da osoba koja postavlja pitanje neće ponovo posetiti stranicu i favorizovati taj odgovor.
  - Drugim rečima, imaćemo određeni broj pitanja sa odgovorima koji nisu najbolje ocenjeni.

Šta je dobar odgovor?

- Drugi mogući način bi bio uzimanje najboljeg i najgoreg odgovora po pitanju kao pozitivan i negativan primer.
  - Međutim, šta u tom slučaju možemo da uradimo sa pitanjima koja imaju samo dobre odgovore?
  - Da li u tom slučaju ima logike da odgovor ocenjen, npr ocenom 9 uzmemo kao pozitivan primer, a odgovor ocenjen ocenom 6 kao negativan?

Inženjering obeležja i formiranje klasifikatora

Šta je dobar odgovor?

- Rešenje problema je kompromis dva prethodno pomenuta načina rešavanja.
- Sve odgovore koji su ocenjeni ocenom većom od nule koristićemo kao pozitivne primere, a odgovore koji su ocenjeni ocenom nula ili manjom od nule kao negativne:

```
>>> all_answers = [q for q,v in meta.items() if v['ParentId']!=-1]
>>> Y = np.asarray([meta[answerId]['Score']>0 for answerId in all_answers])
```

Upotreba metode K-najbližih suseda – primer nevezan za naš problem.

```
>>> from sklearn import neighbors
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=2)
>>> print(knn)
KNeighborsClassifier(algorithm='auto', leaf_size=30,
metric='minkowski', n_neighbors=2, p=2, weights='uniform')
>>> knn.fit([[1],[2],[3],[4],[5],[6]], [0,0,0,1,1,1])
>>> knn.predict(1.5)
array([0])
>>> knn.predict(37)
array([1])
>>> knn.predict(3)
array([0])
```

Upotreba metode K-najbližih suseda – primer nevezan za naš problem.

- Verovarnoće pripadnosti primera pojedinim klasama možemo odrediti metodom `predict_proba()`. U prethodnom slučaju imamo dve klase, 0 i 1, pa će metoda vratiti niz od dva elementa:

```
>>> knn.predict_proba(1.5)
array([[ 1., 0.]])
>>> knn.predict_proba(37)
array([[ 0., 1.]])
>>> knn.predict_proba(3.5) # napomena tačka 3.5 je podjednako udaljena od 3 i 4.
array([[ 0.5, 0.5]])
```

Inženjering obeležja i formiranje klasifikatora

Inženjering obeležja.

- Vrednost atributa Text je, kao što je već rečeno, „sirov“ tekst i ne možemo da ga u tom obliku prosledimo klasifikatoru.
- Međutim, iz ovog atributa se mogu „izvući“ obeležja koja su upotrebljiva.
- Na primer, možemo da pretpostavimo da broj HTML linkova u odgovoru utiče na kvalitet odgovora, pri čemu broj linkova u delovima koda ignorišemo.

Inženjering obeležja i formiranje klasifikatora

Inženjering obeležja.

```
import re
code_match = re.compile('<pre>(.*?)</pre>', re.MULTILINE | re.DOTALL)
link_match = re.compile('<a href="http://.*?".*?>(.*?)</a>', re.MULTILINE | re.DOTALL)
tag_match = re.compile('<[^>]*>', re.MULTILINE | re.DOTALL)

def extract_features_from_body(s):
  link_count_in_code = 0
  for match_str in code_match.findall(s):
    link_count_in_code += len(link_match.findall(match_str))
  return len(link_match.findall(s)) - link_count_in_code
```

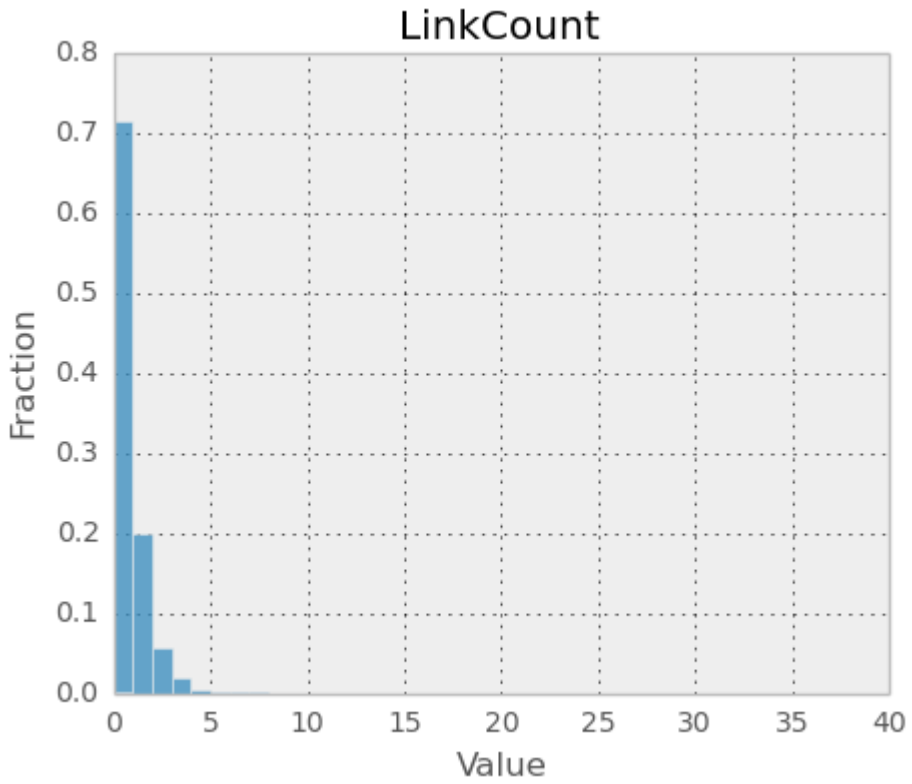
Inženjering obeležja i formiranje klasifikatora

Inženjering obeležja.

- Na ovaj način smo generisali jedno obeležje.
- Pre nego što obučimo klasifikator, nije loše da pogledamo čime ga obučavamo.
- Na grafiku na str. 26 iscrtane prikazano je koliko se često različite vrednosti obeležja pojavljuju u podacima.
- Sa grafika se vidi da većina postova uopšte nema linkove, tako da možemo da zaključimo da ovo obeležje neće biti dovoljno za obuku klasifikatora.
- Međutim, ipak ćemo obučiti klasifikator da bi smo imali uvid u „trenutno stanje eksperimenta“.

Inženjering obeležja i formiranje klasifikatora

Inženjering obeležja.



Inženjering obeležja i formiranje klasifikatora

Obuka klasifikatora.

- Trenutno, broj suseda možemo da nagađamo. Kada budemo imali bolji uvid u podatke, imaćemo i predstavu o poželjnom broj suseda. Za sada ćemo „šac metodom“ odabrati parametar *k*=5, odnosno koristiti 5 suseda.

```
X = np.asarray([extract_features_from_body(text) for post_id, text in
                fetch_posts() if post_id in all_answers])
knn = neighbors.KNeighborsClassifier()
knn.fit(X, Y)
```

Merenje performansi klasifikatora unakrsnom validacijom.

```
from sklearn.cross_validation import KFold
scores = []
cv = Kfold (n=len(X), k=10, indices=True)
for train, test in cv:
  X_train, y_train = X[train], Y[train]
  X_test, y_test = X[test], Y[test]
  clf = neighbors.KNeighborsClassifier()
  clf.fit(X, Y)
  scores.append(clf.score(X_test, y_test))
print("Mean(scores)=%.5f\tStddev(scores)=%.5f"\
      %(np.mean(scores), np.std(scores)))
```

Merenje performansi klasifikatora unakrsnom validacijom.

- Prethodni kod nam vraća sledeći izlaz:

```
Mean(scores)=0.50250      Stddev(scores)=0.055591
```

- Naš klasifikator postiže tačnost 55%, što znači da je beskoristan.
- Drugim rečima, potrebna su nam dodatna obeležja.

Inženjering obeležja i formiranje klasifikatora

Inženjering obeležja (nastavak).

- Kao što smo uveli pretpostavku da broj HTML linkova u odgovoru utiče na kvalitet odgovora, uvešćemo pretpostavku da broj linija koda (označeni tagovima <pre>...</pre>) takođe utiče na kvalitet odgovora.
- Nakon izdvajanja koda, prebrojaćemo reči, odnosno tokene u preostalom delu odgovora.
- Ispitivanjem performansi novog klasifikatora se može zaključiti da dodavanje novog obeležja blago povećava tačnost:

```
Mean(scores)=0.59800      Stddev(scores)=0.02600
```

- Drugim rečima, 4 odgovora od 10 klasifikovaćemo pogrešno.

Inženjering obeležja i formiranje klasifikatora

Inženjering obeležja (nastavak).

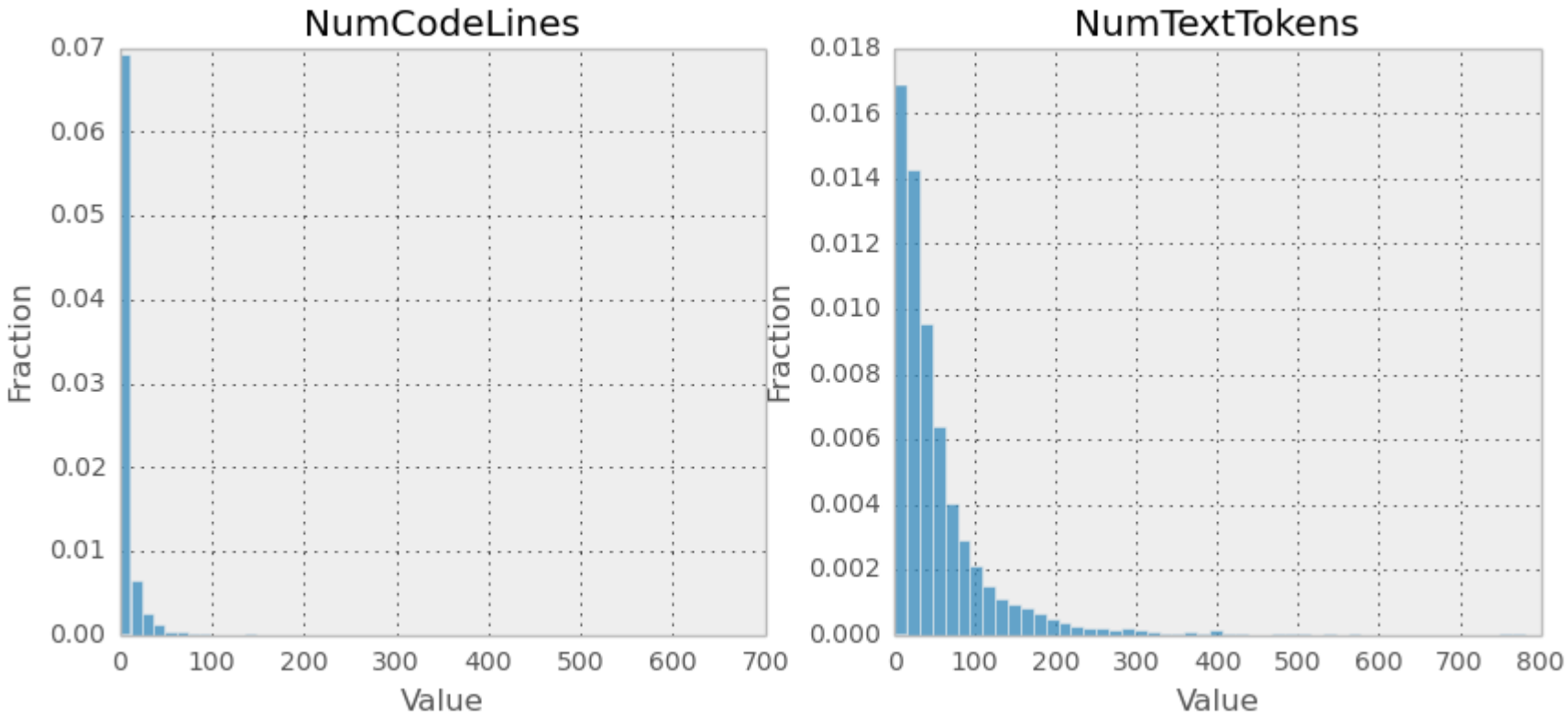
```
def extract_features_from_body(s):
  num_code_lines = 0
  link_count_in_code = 0
  code_free_s = s
  # ukloni kod i prebroj linije
  for match_str in code_match.findall(s):
    num_code_lines += match_str.count('\n')
    code_free_s = code_match.sub("", code_free_s)
    # ponekad kod sadrži linkove koje ne želimo da brojimo
    link_count_in_code += len(link_match.findall(match_str))
```

Inženjering obeležja (nastavak).

```
  links = link_match.findall(s)
  link_count = len(links)
  link_count -= link_count_in_code
  html_free_s = re.sub(" +", " ", tag_match.sub('', code_free_s)).replace("\n", "")
  # ukloni linkove iz teksta pre brojanja reči
  for link in links:
    if link.lower().startswith("http://"):
    link_free_s = link_free_s.replace(link,'')
    num_text_tokens = html_free_s.count(" ")
  return num_text_tokens, num_code_lines, link_count
```

Inženjering obeležja i formiranje klasifikatora

Inženjering obeležja (nastavak).

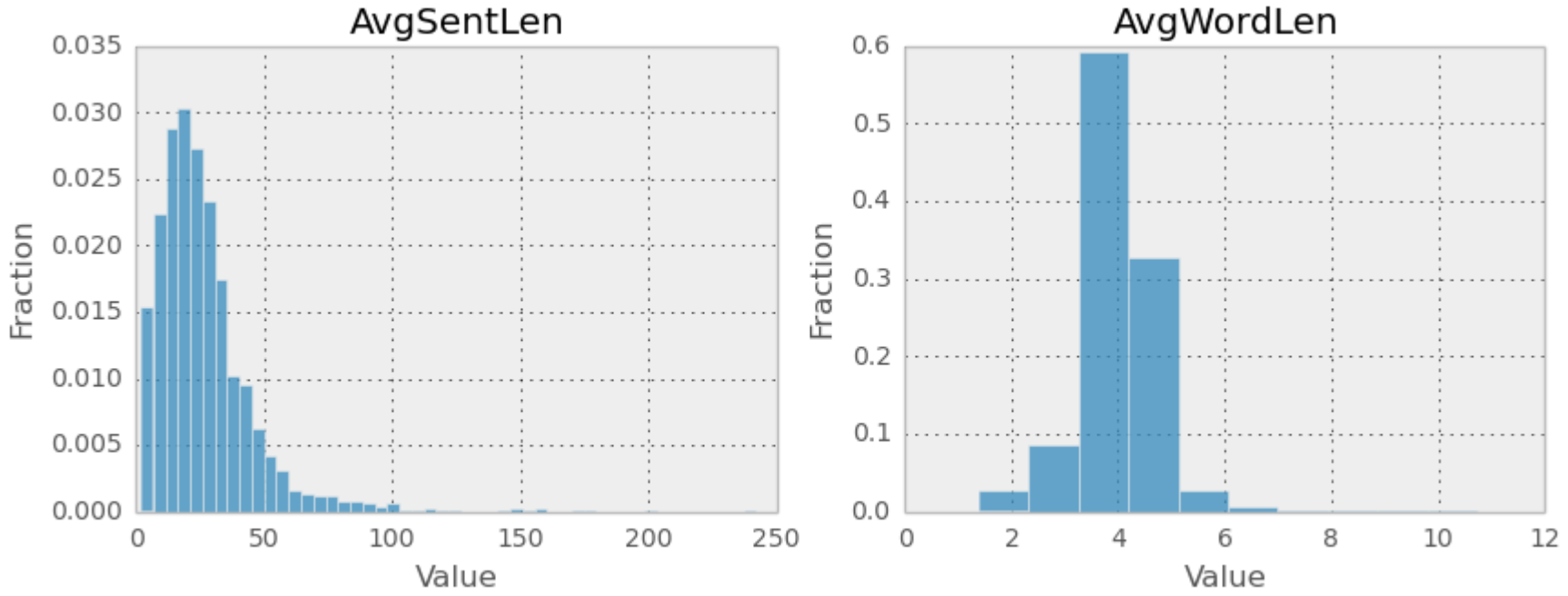


Inženjering obeležja (nastavak).

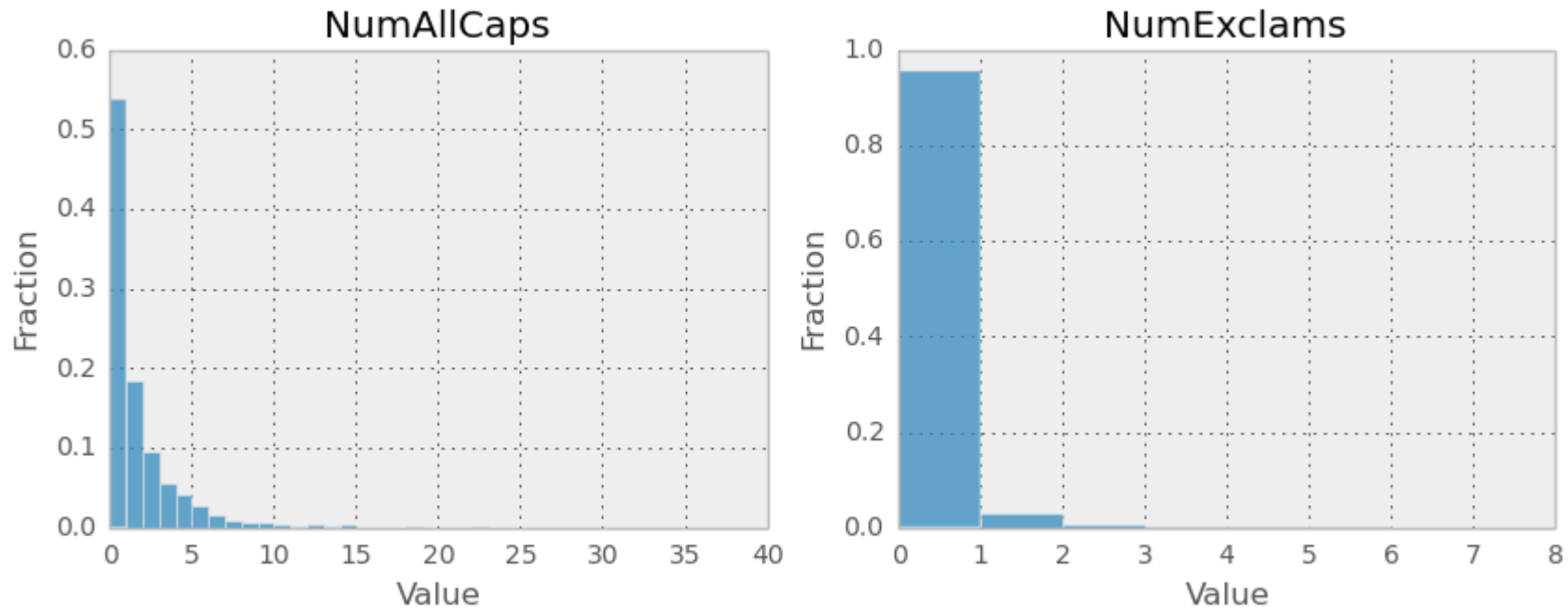
- Proširićemo vektor obeležja dodavanjem novih.
  - Vrednost obeležja AvgSentLen je prosečan reči u rečenici. Ovo uvodimo pod pretpostavkom da dobri postovi ne sadrže isuviše dugačke rečenice, što za čitaoca može biti odbojno.
  - Vrednost obeležja AvgWordLen je prosečna dužina reči u postu, odnosno prosečan broj karaktera jedne reči u postu.
  - Vrednost obležja NumAllCaps je broj reči napisanih velikim slovima, što se smatra lošim stilom pisanja.
  - Vrednost obeležja NumExclams je broj znakova uzvika.

Inženjering obeležja i formiranje klasifikatora

Inženjering obeležja (nastavak).



Inženjering obeležja (nastavak).



Inženjering obeležja i formiranje klasifikatora

Inženjering obeležja (nastavak).

- Dodali smo još četiri obeležja i trenutno ih imamo sedam. Da li smo napredovali?

```
Mean(scores)=0.61400      Stddev(scores)=0.02154      # nakon dodavanja 4 obeležja
Mean(scores)=0.59800      Stddev(scores)=0.02600      # pre dodavanja 4 obeležja
```

- Postavljamo sledeće pitanje: zašto smo tako malo napredovali?

Inženjering obeležja i formiranje klasifikatora

Zašto loše napredujemo?

- Da bi ste shvatili zašto loše napredujemo, podsetite se kako kNN radi.
- Naš 5NN klasifikator klasifikuje post tako što određuje vrednosti sedam obeležja, pronalazi pet najbližih postova, a zatim dodeljuje klasu na osnovu klase većine (u 5 najbližih postova).
- Najbliži postovi se određuju računanjem Euklidskog rastojanja, što znači da se svih sedam obeležja tretiraju na isti način.
- kNN ne uči, na primer, da je obeležje NumTextTokens manje značajno od obeležja NumLinks i slično.

Zašto loše napredujemo?

- Pretpostavićemo da imamo dva posta (A i B) koji se razlikuju samo u dva prethodno pomenuta obeležja i uporedićemo ih sa novim postom.

Post	NumLinks	NumTextTokens
A	2	20
B	0	25
novi	1	23

- Iako mi možemo da pretpostavimo da su linkovi značajniji od čistog teksta, post B će na osnovu Euklidske metrike biti sličniji novom postu nego post A.
- Dakle, kNN ima problem sa obradom ovih podataka.

Inženjering obeležja i formiranje klasifikatora

Na koji način možemo da poboljšamo rezultate eksperimenta?

- Generalno, postoje četiri načina za poboljšanje rezultata:
  - dodavanje novih podataka,
  - reinženjering obeležja, odnosno izmena vektora obeležja (dodavanje novih, uklanjanje postojećih itd.),
  - izmena parametara modela,
  - upotreba drugog algoritma.
- U ovom trenutku ljudi često pokušavaju da povećaju performanse slučajnim odabirom prethodno pomenutih opcija, nadajući se da će u jednom trenutku naći odgovarajuće rešenje.
- Međutim, ovaj način rešavanja problema je pogrešan jer se u najvećem broju slučajeva gubi isuviše vremena na pretraživanje svih mogućih rešenja.

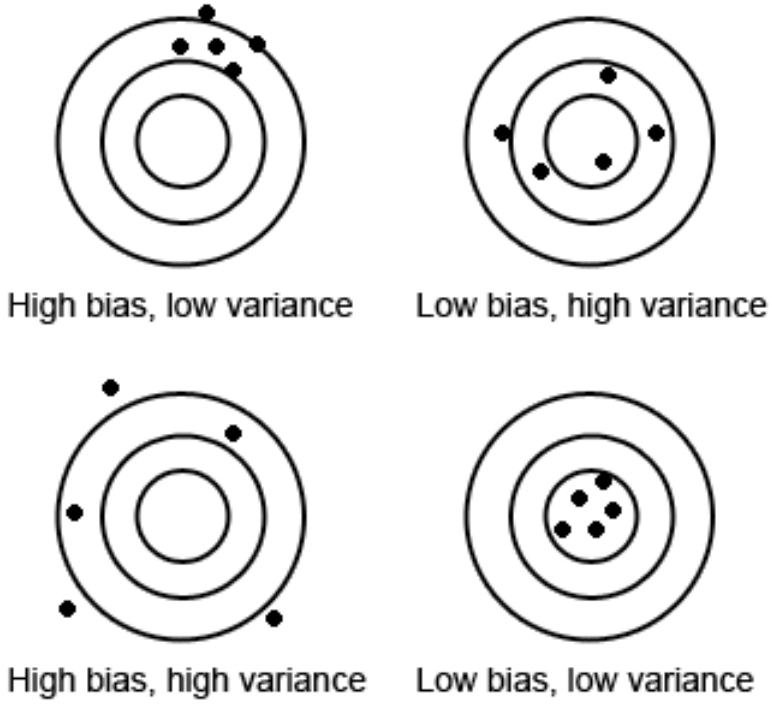
Kako rešavamo problem niske tačnosti?

Šta je pomeraj a šta varijansa?

- Visok pomeraj (engl. *bias*) je karakterističan za modele koji su nedovoljno prilagođeni obučavajućim podacima (engl. *underfitting*).
- Problem visokog pomeraja možemo rešiti:
  - dodavanjem novih obeležja,
  - povećanjem složenosti modela, ili
  - zamenom modela.
- Visoka varijansa (engl. *variance*) je karakteristična za modele koji su isuviše prilagođeni obučavajućim podacima (engl. *overfitting*).
- Problem visoke varijanse možemo rešiti:
  - dodavanjem instanci za obučavanje modela, ili
  - smanjenjem složenosti modela.

Kako rešavamo problem niske tačnosti?

Šta je pomeraj a šta varijansa?



Kako rešavamo problem niske tačnosti?

Šta je pomeraj a šta varijansa?

- Na osnovu prethodne slike (analiza gađanja mete u streljani):
  - *high bias*, *low variance* – strelac grupiše metke, ali udaljeno od centra;
  - *low bias*, *high* variance – strelac ne grupiše metke, ali su svi relatvno blizu centra;
  - *high bias*, *high variance* – strelac ne grupiše metke, i svi su udaljeni od centra;
  - *low bias*, *low variance* – strelac grupiše metke i svi su blizu centra mete.
- Ponekada je potrebno napraviti kompromis između pomeraja i varijanse.
- Da li možete da shvatite šta ovaj kompromis znači kada strelac gađa „metu“?

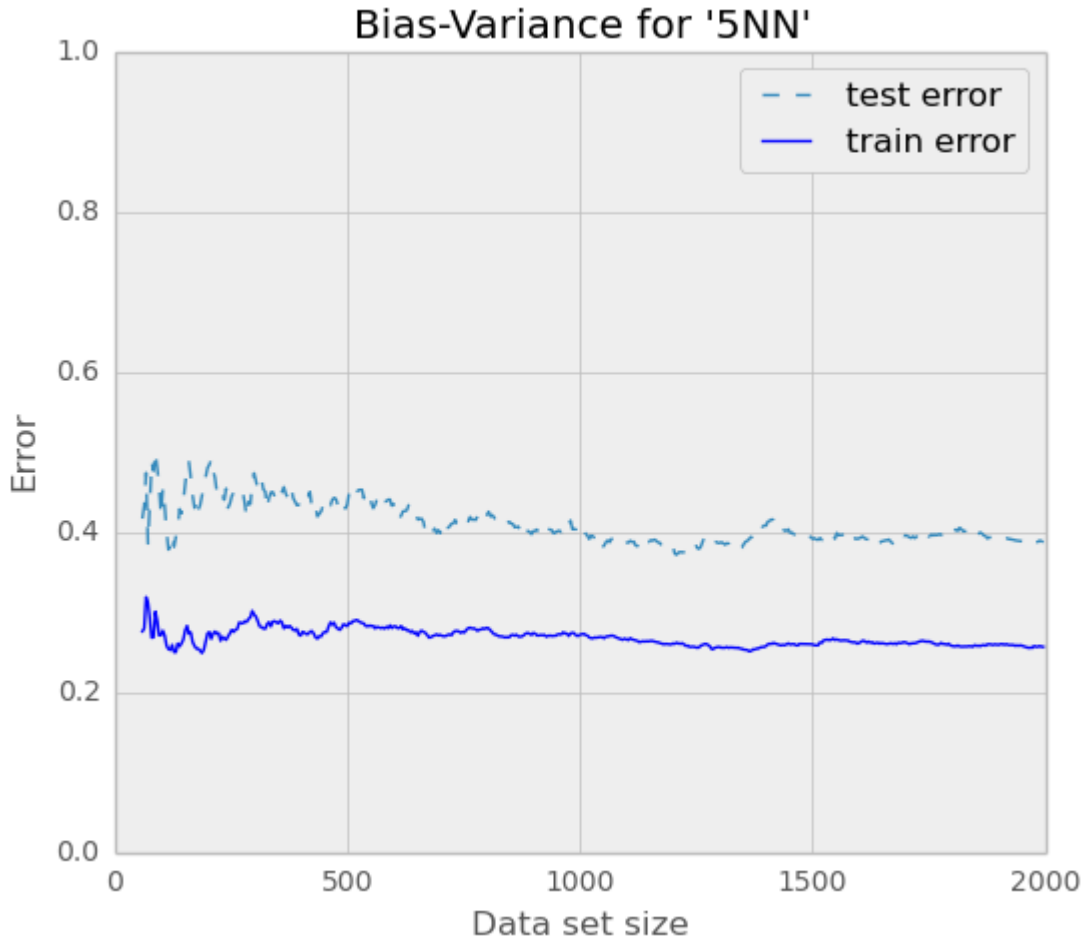
Kako rešavamo problem niske tačnosti?

Kako možemo da otkrijemo odakle problem potiče?

- Da bi smo otkrili odakle problem potiče, trebamo da iscrtamo greške na obučavajućem i test skupu za različite veličine skupa podataka (počev od malih ka velikim skupovima).
- Visok pomeraj se na grafiku tipično otkriva kao greška na test skupu koja dostiže neprihvatljivo visoku vrednost zajedno sa greškom obuke za veće skupove podataka.
- Visoka varijansa se na grafiku tipično otkriva kao veliki razmak između grešaka na obuci i testu.
- Greške na grafiku za različite veličine skupova za 5NN (v. str. 45) ukazuju na problem visoke varijanse.

Kako rešavamo problem niske tačnosti?

Greške za različite veličine skupova za 5NN.



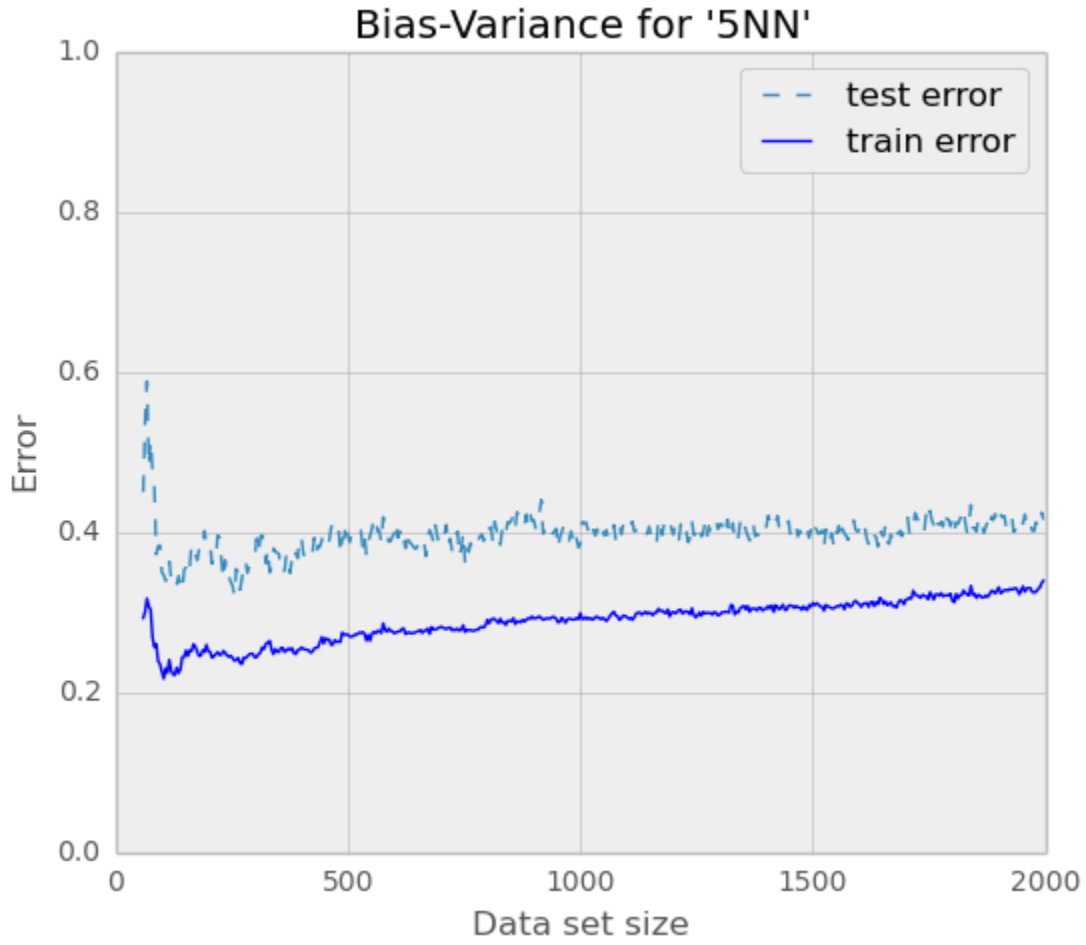
Kako rešavamo problem niske tačnosti?

Kako možemo rešimo problem?

- Na osnovu grafika možemo da zaključimo da dodavanje obučavajućih podataka neće pomoći, zato što greška testa ne pada ispod 0,4 sa porastom veličine skupa.
- Dve preostale opcije su smanjenje složenosti (na primer, povećanjem broja suseda, odnosno parametra *k*) ili smanjenje dimenzionalnosti vektora u obučavajućem skupu.
- Smanjenje dimenzionalnosti vektora u obučavajućem skupu ne pomaže, što se jednostavno može ustanoviti iscrtavanjem grafika za model obučen vektorima koji sadrže samo obeležja LinkCount i NumTextTokens (v. str. 47).

Kako rešavamo problem niske tačnosti?

Greške za različite veličine skupova za jednostavniji model.



Kako rešavamo problem niske tačnosti?

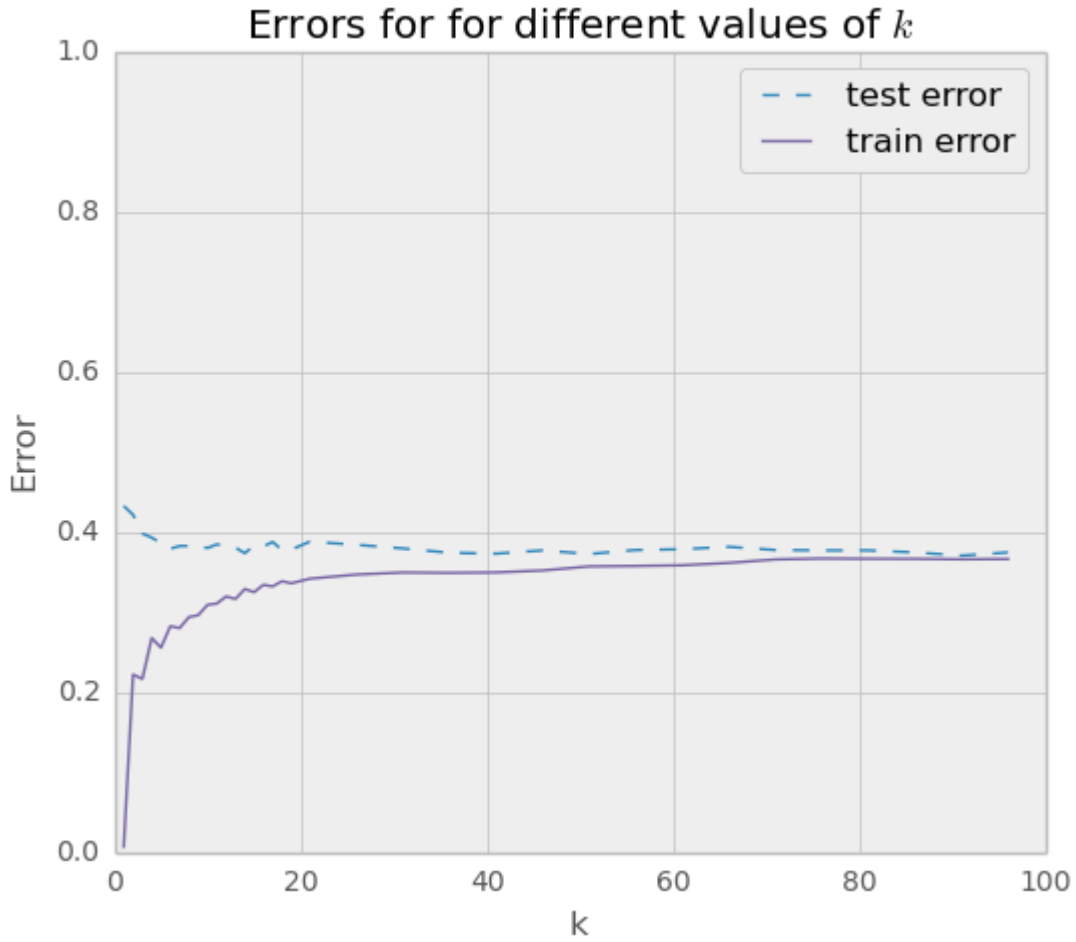
Da li smanjenje složenosti modela može da pomogne?

- Smanjenje složenosti modela povećanjem parametra *k* utiče blago pozitivno na performanse (v. str. 49).

k	mean (scores)	stddev (scores)
40	0,62800	0,03750
10	0,62000	0,04111
5	0,61400	0,02154

Kako rešavamo problem niske tačnosti?

Greške za različite vrednosti parametra *k*.



Kako rešavamo problem niske tačnosti?

Da li smanjenje složenosti modela može da pomogne?

- Međutim, iako je za *k*=40, greška na testu manja, za klasifikaciju novog posta nam treba 40 najbližih kako bi doneli odluku da li je odgovor ispravan ili ne.
- Drugim rečima, možemo da zaključimo da metoda najbližih suseda u ovom slučaju ne daje najbolje razultate i da primenimo drugi metod klasifikacije za koji je pokazano da postiže dobre rezultate pri klasifikaciji teksta.

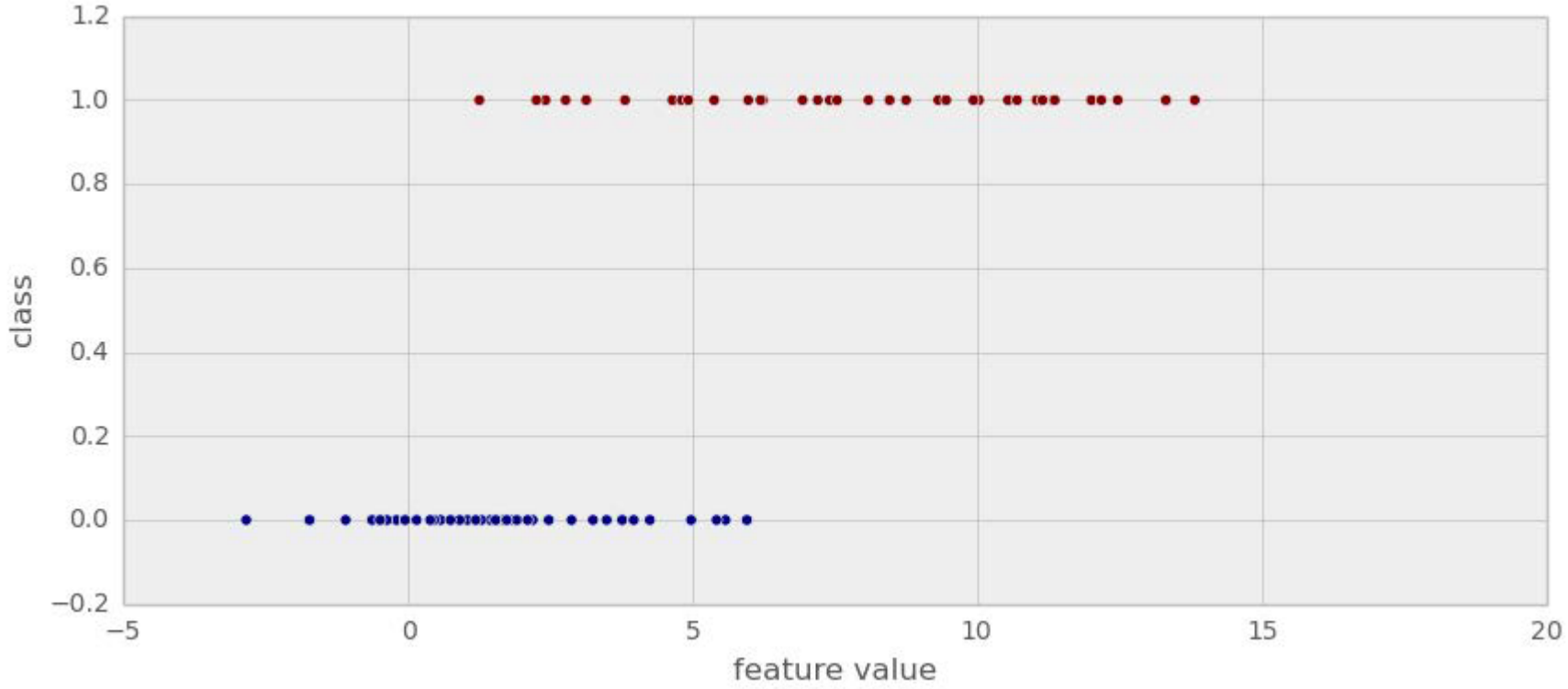
Upotreba logističke regresije

Jednostavno objašnjenje.

- Suprotno svom imenu, logistička regresija je metoda klasifikacije a ne regresije.
- Logistička regresija se može jednostavno objasniti na sledećem primeru:
  - posmatrajte grafik sa instancama (v. str. 52) kojima su za sintetičke vrednosti obeležja X dodeljena klasna obeležja 0 i 1.
  - Podaci su zašumljeni jer se klase „preklapaju“ u opsegu vrednosti obeležja [1, 6].
- U ovom slučaju, umesto modelovanja diskretnih klasa, bolje je modelovati verovatnoće da za datu vrednost obeležja instanca pripada klasi 1, tj. $P(X)$.
- Kada obučimo model, klasu nove instance predviđamo kao klasu 1 ukoliko je $P(X) > 0$, a u suprotnom kao klasu 0.
- Napomena: verovatnoće moraju da pripadaju opsegu [0, 1].

Upotreba logističke regresije

Jednostavno objašnjenje.



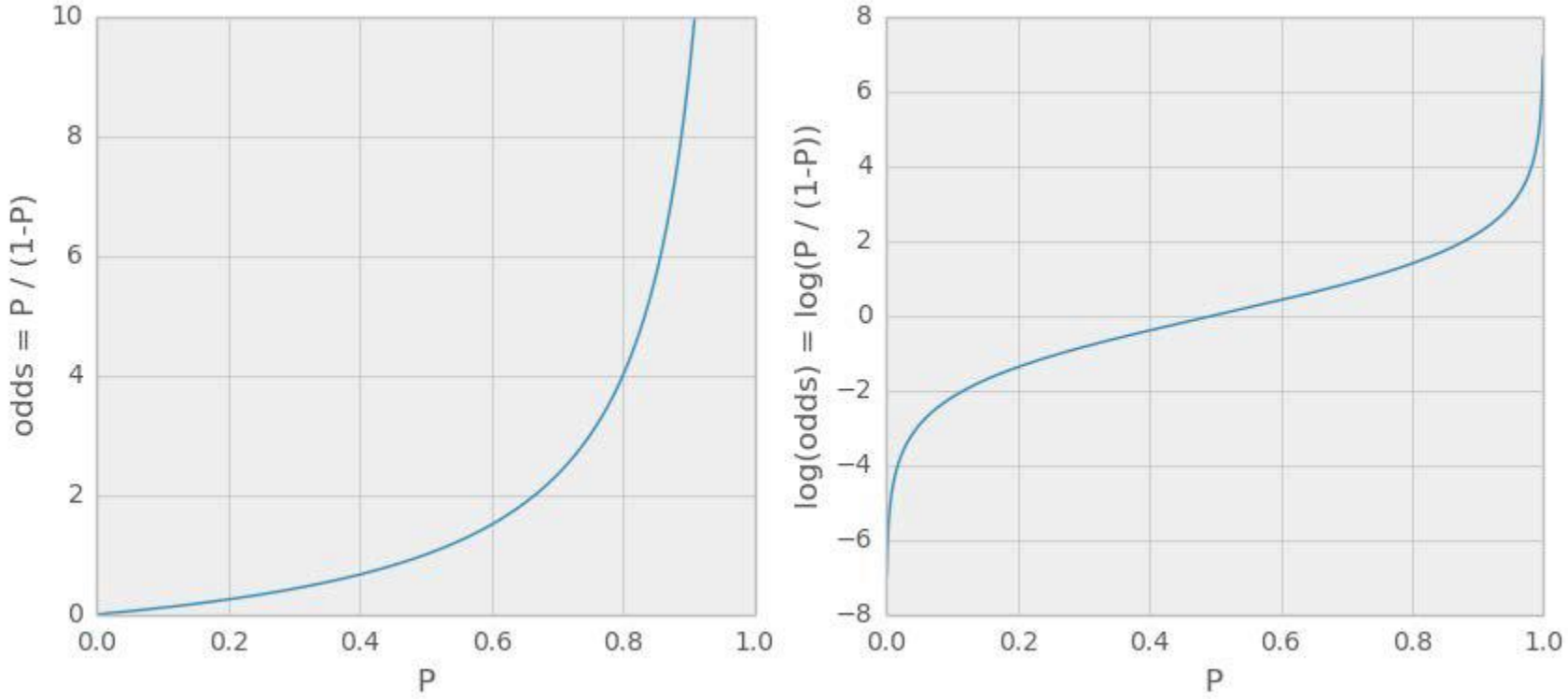
Upotreba logističke regresije

Jednostavno objašnjenje.

- Neka obeležje ima verovatnoću 0,9 da pripada klasi 1, tj. $P(y = 1) = 0{,}9$.
- Odnos verovatnoća (engl. *odds*) u ovom slučaju je $P(y = 1)/P(y = 0) = 0{,}9/0{,}1 = 9$, što znači da je šansa 9:1 da instanca pripada klasi 1.
- U slučaju da je $P(y = 0{,}5)$, šansa da instanca pripadne klasi 1 bila bi 1:1.
- Odnos verovatnoća je ograničen nulom, a može da ode u beskonačnost (v. str. 54, levo).
- Ukoliko u priču ubacimo logaritam, onda sve verovatnoće sa intervala [0, 1] možemo da mapiramo u opseg [-∞, + ∞] , pri čemu i dalje veća verovatnoća vodi ka većem odnosu verovatnoća.

Upotreba logističke regresije

Jednostavno objašnjenje.



Upotreba logističke regresije

Jednostavno objašnjenje.

- To znači da linearnu kombinaciju obeležja možemo da uklopimo u logaritam odnosa verovatnoća. Drugim rečima, linearno $y = c_0 + c_1 x_i$ menjamo sa:

$$\log\left(\frac{p_i}{1-p_i}\right) = c_0 + c_1 x$$

- Ovo rešavamo za p_i:

$$p_i = \frac{1}{1 + e^{-(c_0 + c_1 x_i)}}$$

- Dakle, tražimo koeficijente tako jednačina proizvede najmanju grešku za parove (x_i, y_i) u našem skupu podataka.

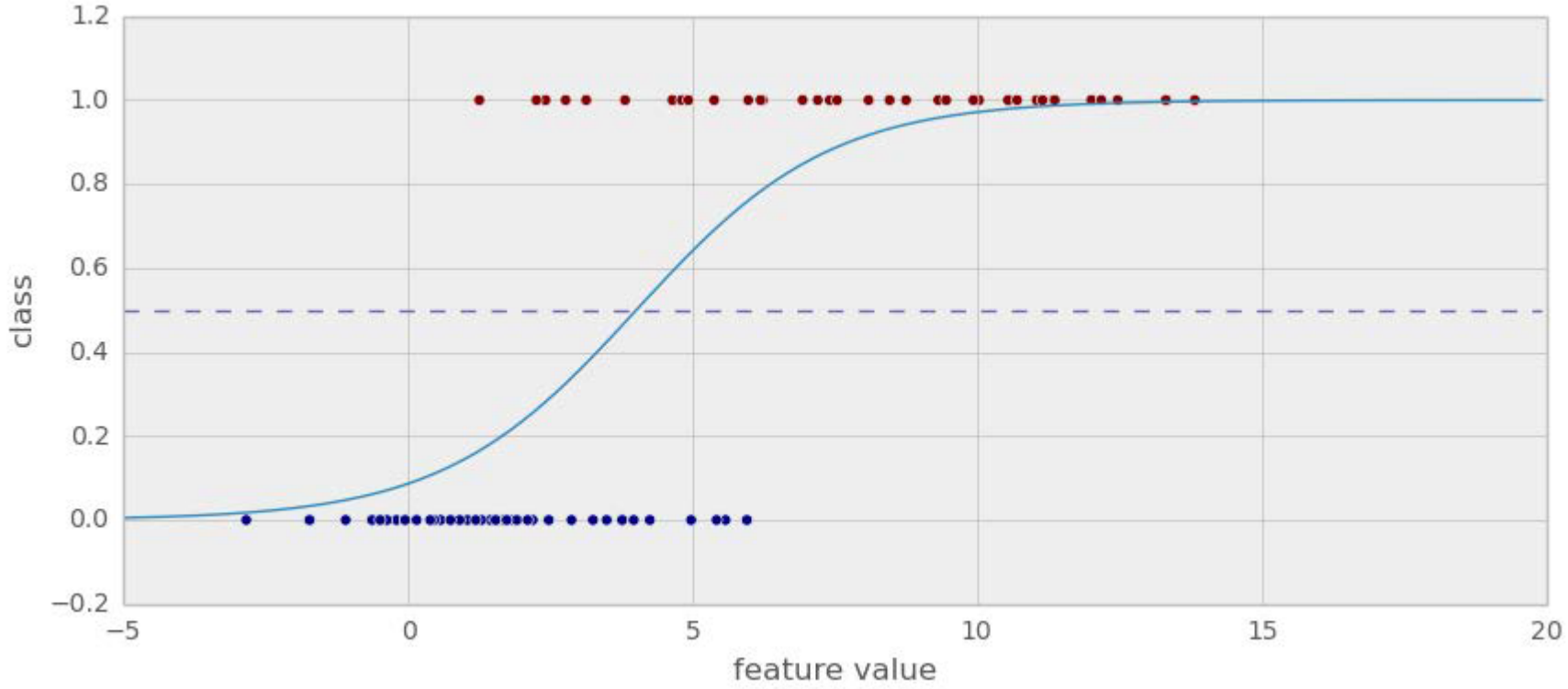
Upotreba logističke regresije

Upotreba logističke regresije iz paketa scikit-learn.

```
>>> from sklearn.linear_model import LogisticRegression
>>> clf = LogisticRegression()
>>> print(clf)
LogisticRegression(C=1.0, class_weight=None, dual=False,
fit_intercept=True, intercept_scaling=1, penalty=l2, tol=0.0001)
>>> clf.fit(X, y)
>>> print(np.exp(clf.intercept_), np.exp(clf.coef_.ravel()))
[ 0.09437188] [ 1.80094112]
>>> def lr_model(clf, X): return 1 / (1 + np.exp(-(clf.intercept_ + clf.coef_*X)))
>>> print("P(x=-1)=%.2f\tP(x=7)=%.2f"%(lr_model(clf, -1), lr_model(clf, 7)))
P(x=-1)=0.05 P(x=7)=0.85
```

Upotreba logističke regresije

Upotreba logističke regresije iz paketa scikit-learn.



Upotreba logističke regresije

Primena logističke regresije na naš skup podataka.

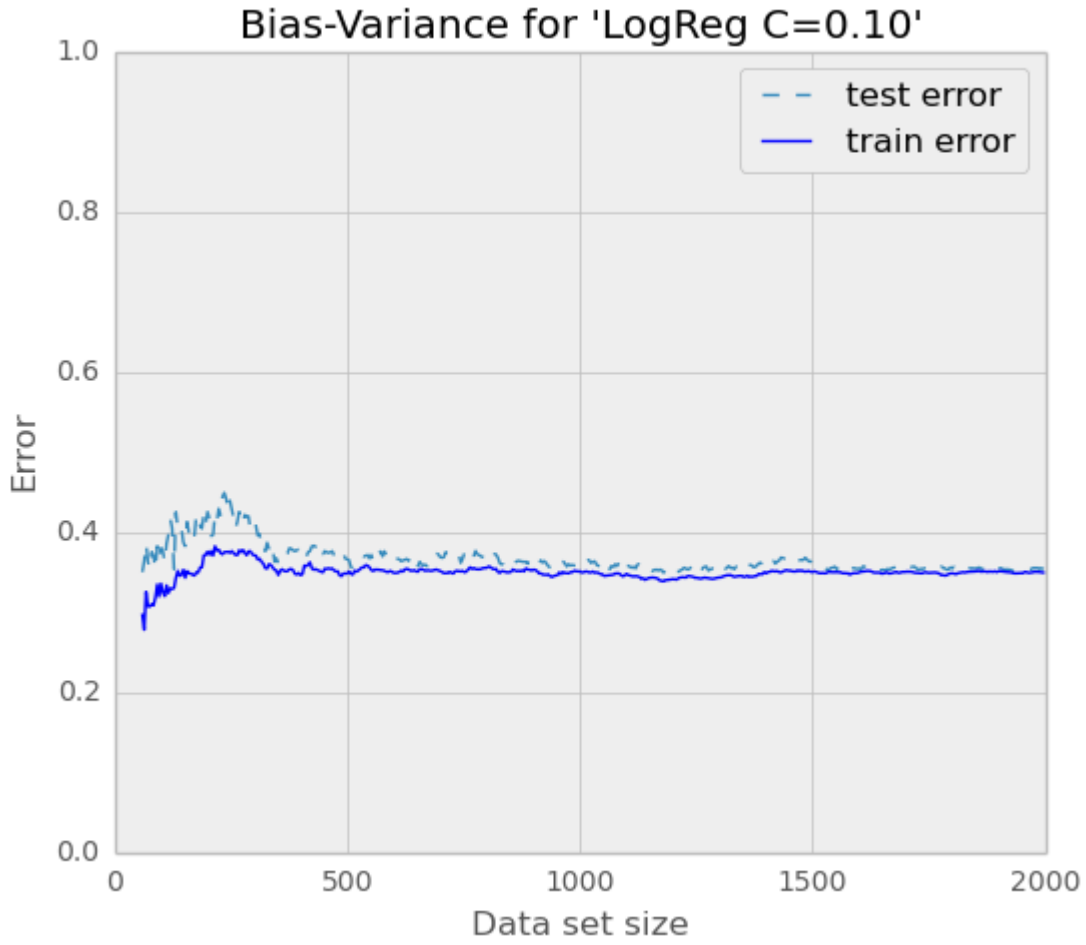
- Koristimo kNN sa parametrom *k*=40 kao osnovu za poređenje, pri čemu menjamo vrednosti regularizacionog parametra *C* za logističku regresiju (manje vrednosti parametra za posledicu imaju veću kaznu zbog složenosti modela).

Metod	mean (scores)	stddev (scores)
LogReg, C=0.1	0,64650	0,03139
LogReg, C=1	0,64650	0,03155
LogReg, C=10	0,64550	0,03102
LogReg, C=0.01	0,63850	0,01950
kNN, k=40	0,62800	0,03750

- Šta možemo da zaključimo na osnovu *bias-variance* grafika za model za koji je *C*=0,1?

Upotreba logističke regresije

Primena logističke regresije na naš skup podataka.



Upotreba logističke regresije

Primena logističke regresije na naš skup podataka.

- Analizom *bias-variance* grafika zaključujemo da model logističke regresije sa parametrom *C*=0,1 ima visok pomeraj – greške na obuci i testu su bliske ali neprihvatljivo visoke.
- Ovo znači da se u trenutnom prostoru obeležja logistička regresija ne može dovoljno prilagoditi podacima, što znači da se ne može formirati klasifikacioni model na osnovu datog algoritma i skupa obeležja.

Upotreba logističke regresije

Šta možemo učiniti dalje?

- Zamenili smo klasifikacioni algoritam i prilagodili parametre novog modela, međutim i dalje ne možemo da obučimo upotrebljiv klasifikator.
- Ovo znači da su:
  - podaci isuviše zašumljeni, što značajno može da oteža rešavanje problema, ili
  - da naš skup obeležja nije prikladan da dovoljno dobro razdvoji klase.

Preciznost i odziv

Šta smo zapravo hteli da postignemo?

- Do sada smo performanse klasifikatora merili koristeći tačnost, međutim, nama nije potreban klasifikator koji savršeno predviđa dobre i loše odgovore.
- Međutim, ako klasifikator možemo podesiti tako da bude dobar u predviđanju jedne klase, mi možemo prilagoditi povratne informacije korisniku.
- Ako, na primer, imamo klasifikator koji vrlo dobro predviđa da li je odgovor loš, mi ne moramo da dajemo povratne informacije sve dok klasifikator ne otkrije da je odgovor loš.
- Slično, ako klasifikator dobro predviđa ispravne i zadovoljavajuće odgovore, mogli bismo na početku korisniku dati komentare u vidu saveta i ukloniti ih kada je klasifikator detektovao da je odgovor dobar.

Preciznost i odziv

Podsetnik.

		Klasifikovan kao	
		Pozitivan	Negativan
Stvarna klasa	Pozitivan	Stvarno pozitivan (TP)	Lažno negativan (FN)
	Negativan	Lažno pozitivan (FN)	Stvarno negativan (TN)

Preciznost i odziv

Šta je preciznost a šta odziv?

- **Preciznost** (engl. *precission*) se definiše na sledeći način:

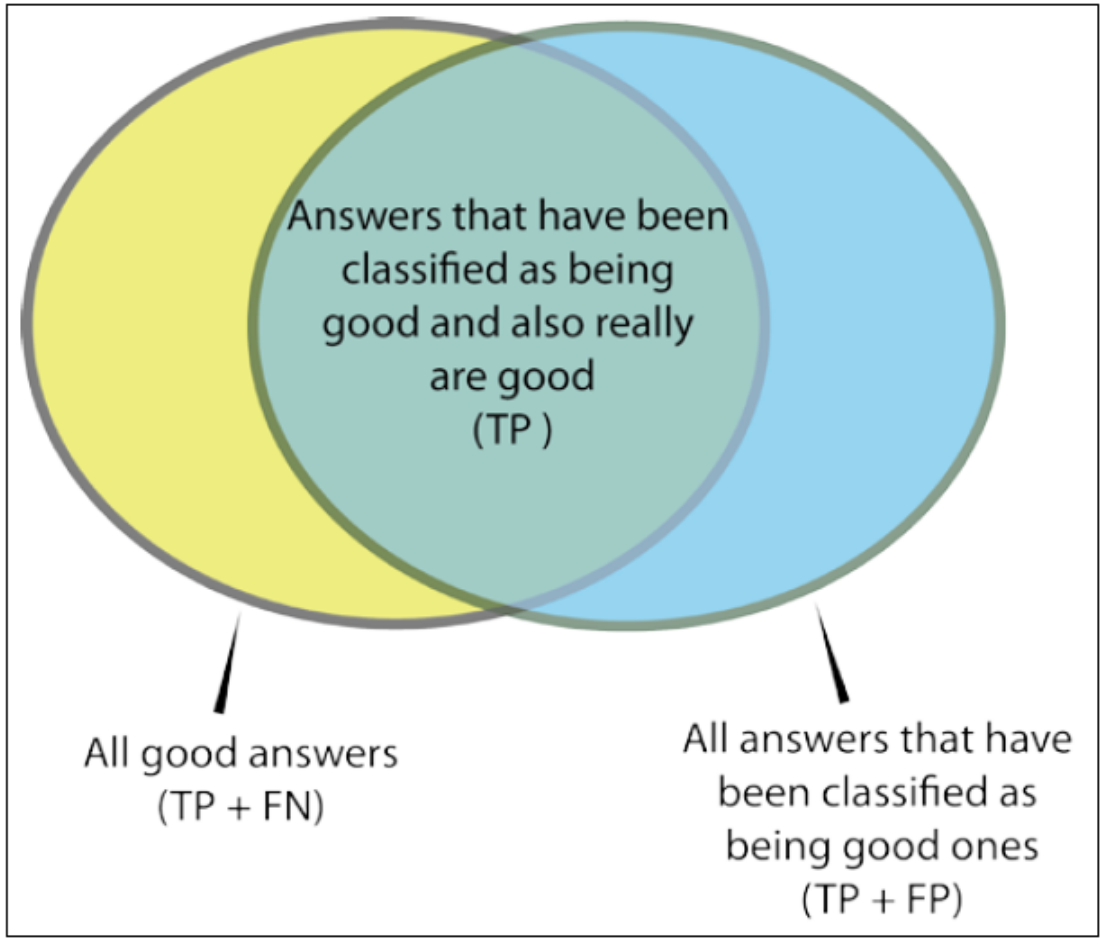
$$P = \frac{TP}{TP + FP}$$

- **Odziv** (engl. *recall*) se definiše na sledeći način:

$$R = \frac{TP}{TP + FN}$$

Preciznost i odziv

Šta je preciznost a šta odziv?



Šta je balansirana F-mera?

- **F-mera** je parametar koji u obzir uzima i preciznost i odziv sistema:

$$F = 1/\left(\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}\right), \alpha \in [0,1]$$

- Za $\alpha < 1/2$, F-mera nadglašava odziv, a za $\alpha = 0$, F-mera je jednaka odzivu, tj. $F = R$.
- Za $\alpha > 1/2$, F-mera nadglašava preciznost, a za $\alpha = 1$, F-mera je jednaka preciznošću.
- Za $\alpha = 1/2$, F-mera predstavlja harmonijsku sredinu preciznosti i odziva, i naziva se balansiranom F-merom:

$$F = \frac{2PR}{P+R}$$

Preciznost i odziv

Kako možemo da optimizujemo sistem u smeru preciznosti?

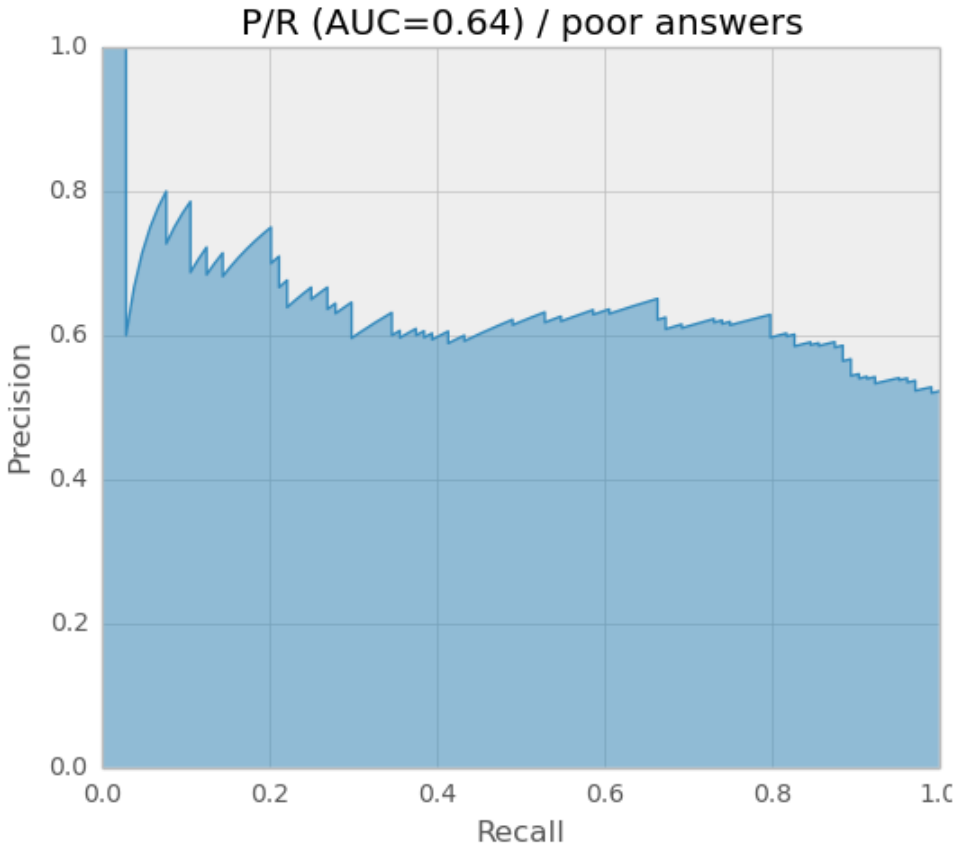
- Do sada smo koristili 0,5 kao prag da odlučimo da li je odgovor dobar ili ne.
- Sada možemo da izračunamo brojeve TP, FP i FN prilikom promene praga od 0 do 1.
- Sa tim brojevima možemo iscrtati grafik preciznost / odziv.

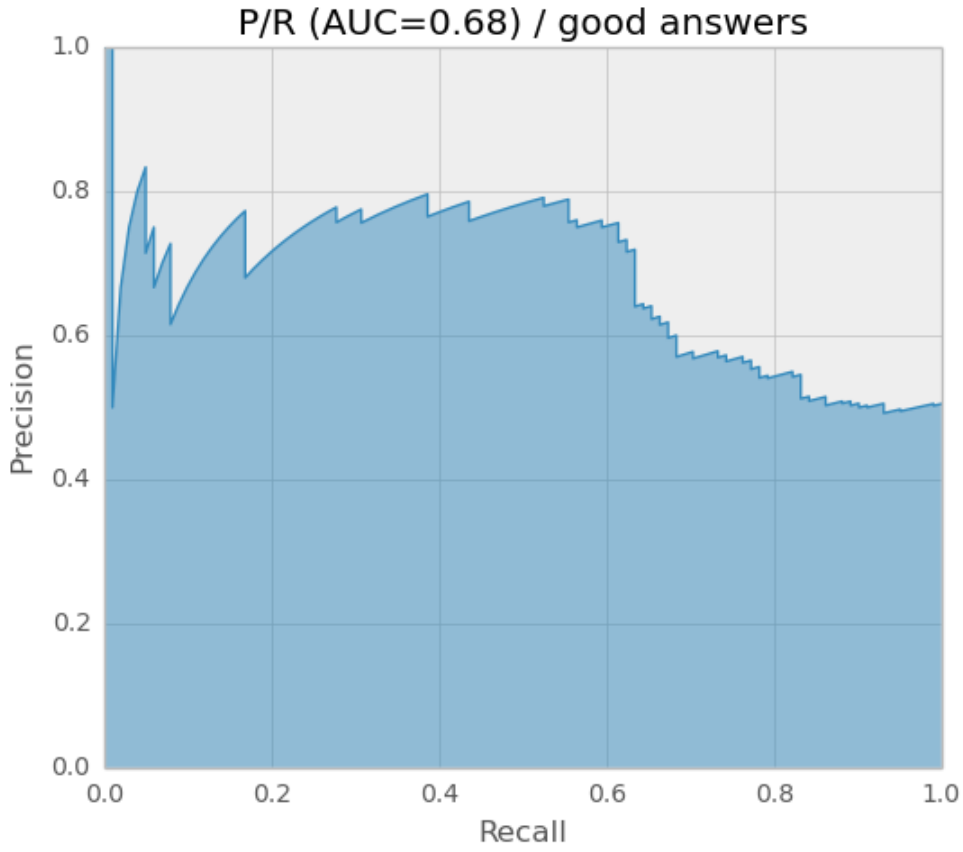
```
>>> from sklearn.metrics import precision_recall_curve
>>> precision, recall, thresholds = precision_recall_curve(y_test, clf.predict(X_test))
```

- Predviđanje jedne klase sa prihvatljivim performansama ne znači uvek da je klasifikator takođe prihvatljiv za predviđanje druge klase.
- Ovo se može videti na naredna dva grafika na kojima su date krive preciznosti / odziva za klasifikaciju loših odgovora (v. levi grafikon) i dobrih odgovora (v. desni grafikon, str. 68).

Preciznost i odziv

Kako možemo da optimizujemo sistem u smeru preciznosti?





Preciznost i odziv

Šta možemo da zaključimo na osnovu grafika?

- Sa levog grafikona se vidi da smo u osnovi jako loše predviđali loše odgovore. Preciznost je vrlo niska i pada do neprihvatljivih 60%.
- Prilikom predviđanja dobrih odgovora, postigli smo preciznost preko 80% sa odzivom od oko 40%.

Preciznost i odziv

Primena praga u procesu predviđanja.

- Koristićemo klasifikator koji nije ni previše loš ni previše dobar da otkrijemo pogodan prag za postizanje visoke preciznosti (> 80%).

```
>>> medium = np.argsort(scores)[int(len(scores) / 2)]
>>> thresholds = np.hstack(([0],thresholds[medium]))
>>> idx80 = precisions>=0.8
>>> print("P=%.2f R=%.2f thresh=%.2f" % (precision[idx80][0],
                                         recall[idx80][0], threshold[idx80][0]))
P=0.80 R=0.37 thresh=0.59
```

- Sa pragom 0,59, dostižemo 80% preciznosti prilikom detekcije dobrih odgovora ukoliko prihvatimo nizak odziv od 37%. To znači da ćemo jedan u tri odgovora detektovati kao zaista dobar, dok za ostale možemo da prikažemo dodatne savete kako se odgovor može poboljšati.

Preciznost i odziv

Primena praga u procesu predviđanja.

- Koristićemo metodu `predict_proba()` koja vraća verovatnoću pripadnosti klasama, umesto metode `predict()` koja vraća pripadnost klasi.
- Nakon toga metodom `classification_report()` proveravamo da li smo u odgovarajućem opsegu preciznosti i odziva.

```
>>> thresh80 = threshold[idx80][0]
>>> probs_for_good = clf.predict_proba(answer_features)[:,1]
>>> answer_class = probs_for_good>thresh80
```

Preciznost i odziv

Primena praga u procesu predviđanja.

```
>>> from sklearn.metrics import classification_report
>>> print(classification_report(y_test, clf.predict_proba [:,1]>0.63,
          target_names=['not accepted', 'accepted']))
             precision  recall  f1-score  support
not accepted  0.59       0.85    0.70      101
accepted      0.73       0.40    0.52      99
avg / total   0.66       0.63    0.61      200
```

- Imajte na umu da korišćenje praga ne garantuje da smo uvek iznad vrednosti preciznosti i odziva koje smo prethodno odredili!

Završne napomene

Obeležja sa niskim značajem za klasifikaciju.

- Uvek je korisno pogledati značaj pojedinih obeležja, odnosno njihov doprinos klasifikaciji.
- Koeficijente za logističku regresiju možemo preuzeti iz (`clf.coef_`) i na taj način steći utisak o značaju obeležja za klasifikaciju.
- Što je koeficijent veći, obeležje ima značajniju ulogu u određivanju da li je odgovor na pitanje dobar ili loš.
- To znači da obeležja sa niskim značajem za klasifikaciju možemo da izbacimo iz vektora obeležja, i da pri tome zadržimo, ili neznačajno umanjimo performanse klasifikatora.

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.