

UVRŠČANJE

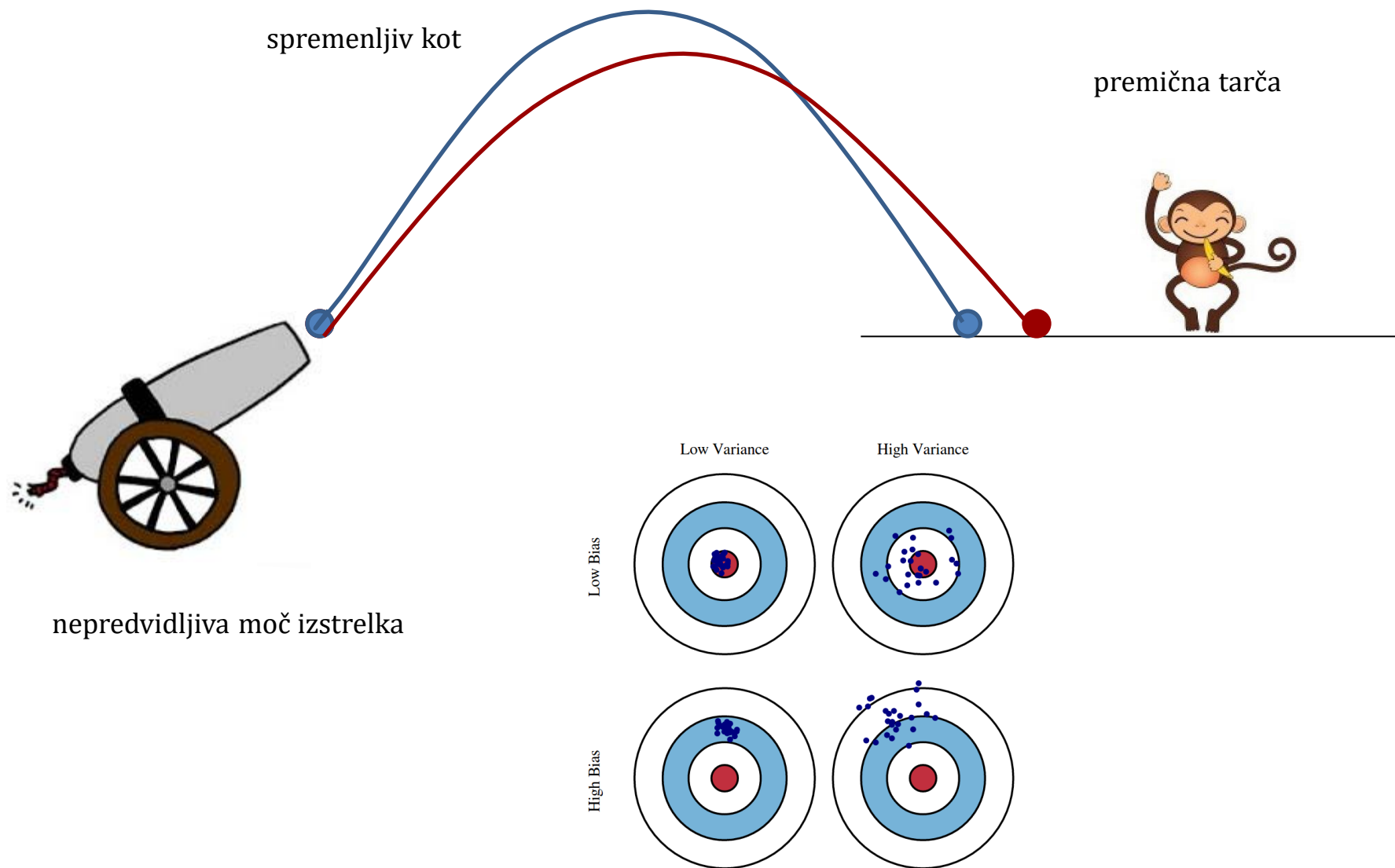
classification

doc. dr. Matej Guid

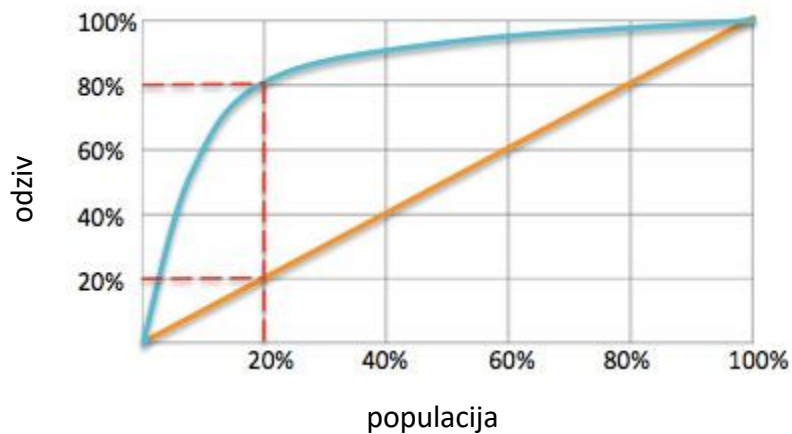
Fakulteta za računalništvo in informatiko
Univerza v Ljubljani

februar 2022

ODKLON, VARIANCA, ŠUM



PRIMER S PODROČJA DIREKTNEGA MARKETINGA



promocija preko direktne pošte...

1.000.000 naslovov: odziv je 0.1% (1.000 odgovorov)

s pomočjo napovednega modela lahko rezultate izboljšamo...

200.000 naslovov: odziv je 0.4% (800 odgovorov)

25.000 naslovov: odziv je 0.8% (200 odgovorov)

Komu naj najprej pošljemo pošto?

Kako razvrstimo potencialne kupce?

Koliko pošiljk naj pošljemo, da maksimiziramo dobiček?

MERJENJE NAPAK PRI NAPOVEDIH

confusion matrix

		RESNIČNOST	
		dobro	slabo
NAPOVED	dobra ocena	TRUE POSITIVE tp	FALSE POSITIVE fp
	slaba ocena	FALSE NEGATIVE fn	TRUE NEGATIVE tn

vsi priporočeni izdelki

$$Precision = \frac{tp}{tp + fp} = \frac{|good\ movies\ recommended|}{|all\ recommendations|}$$

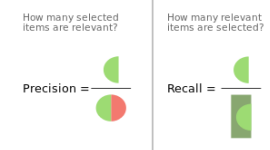
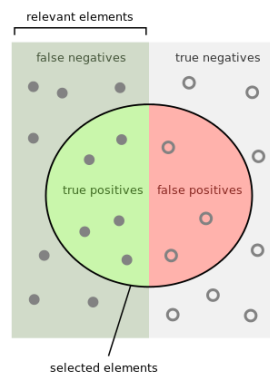
% priporočenih filmov, ki so res dobri

vsi dobri izdelki

$$Recall = \frac{tp}{tp + fn} = \frac{|good\ movies\ recommended|}{|all\ good\ movies|}$$

% priporočenih od vseh dobrih filmov

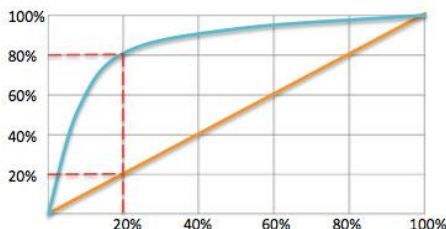
priklic
recall



$$\text{klasifikacijska točnost} = \frac{TP + TN}{P + N}$$

Kaj je hujše: FP ali FN?

- banka: posoditi denar nekomu, ki ga ne vrača **ali** ne odobriti kredita?
- detekcija vlomov: sprožiti lažen alarm **ali** ne detektirati vloma?
- medicina: postaviti napačno diagnozo zdravemu **ali** bolnemu pacientu?
- marketing: poslati pošto nezainteresiranemu kupcu **ali** ne poslati pošte zainteresiranemu kupcu?



Kaj pa, če nimamo na voljo vseh podatkov o naslovnikih?

Kaj se lahko naučimo iz preteklih marketinških akcij?

Iz česa (iz katerih atributov) se lahko naučimo največ?

Kako oceniti, kako dober je naš klasifikator?

imamo podano

classification

množica podatkov ...

... opisana z značilkami (atributi) ...

... posamezni primeri pripadajo različnim razredom

dobiti želimo

klasifikacijski model oz. klasifikator, ki bo primere uvrstil v pravilen razred

Uspešnosti klasifikacije nikoli ne ocenjujemo na učnih primerih!

Množico primerov D zato najprej razdelimo:

- na množico učnih primerov D_L
- na množico testnih primerov D_T

$$D_L \cap D_T = \emptyset$$

$$D_L \cup D_T = D$$



učni podatki



testni podatki

prečno preverjanje reda k (*k-fold cross validation*)



izloči enega (leave-one-out) ... primerno za manjše množice podatkov



metoda stremena (*bootstrap*) ... iz množice N primerov z vračanjem naključno izberemo enako veliko množico za učenje, vse neizbrane primere uporabimo za testiranje...
vzorčenje, gradnja modela, vrednotenje... ponovimo tipično od 100 do 1000-krat

Izbor metode in njenih parametrov, **klasična napaka**:

- na celotni množici primerov s prečnim preverjanjem ocenimo, pri kateri vrednosti parametrov (npr. k pri kNN) daje izbrana metoda najboljše rezultate
- nato poročamo o izračunani meri točnosti

Kaj je tu narobe?

Izbor optimalnih parametrov (tako kot gradnjo modela) je potrebno preveriti na neodvisni množici podatkov! Najprej notranje prečno preverjanje, nato zunanja zanka!



diskretni (oz. binarni) klasifikator

binary classifier

binarni \rightarrow v primeru dveh razredov

P pozitivnih, N negativnih primerov, klasifikator (pravilno/napačno) določi razred

verjetnostni (oz. točkovalni) klasifikator

probabilistic classifier ... probability estimating... scoring

funkcija $f: X \rightarrow [0,1]$... vsak primer x preslika v realno vrednost

$f(x)$... točkovna ocena

prag (*threshold*) t ... primeri x , za katere velja $f(x) \geq t$ so klasificirani kot P, ostali kot N

vsak par (verjetnostni klasifikator, t) predstavlja binarni klasifikator

TP(t) in FP(t) sta monotono padajoči funkciji ... Zakaj?

s spreminjanjem vrednosti praga t dobimo **družino binarnih klasifikatorjev**

ROC (*Receiver Operating Characteristics*) ... karakteristika sprejemnika (iz zgodovine, II. svetovna vojna)

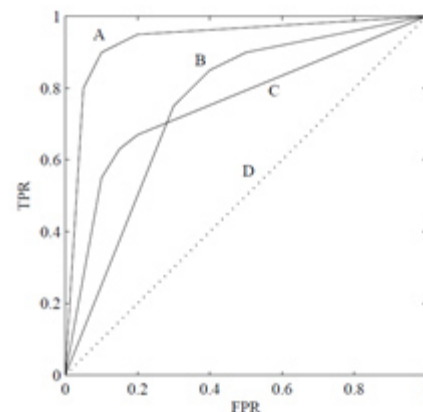
analiza ROC metoda za vrednotenje, primerjanje in izbiro klasifikatorjev

krivulja ROC krivulja na dvodimenzionalnem grafu..

prikazuje sposobnost klasifikatorja za izdajanje dobrih točkovnih ocen (*scores*)

ponazoritev kompromisa med

- pogostost zadetka (*hit rate*)
- pogostost lažnega alarma



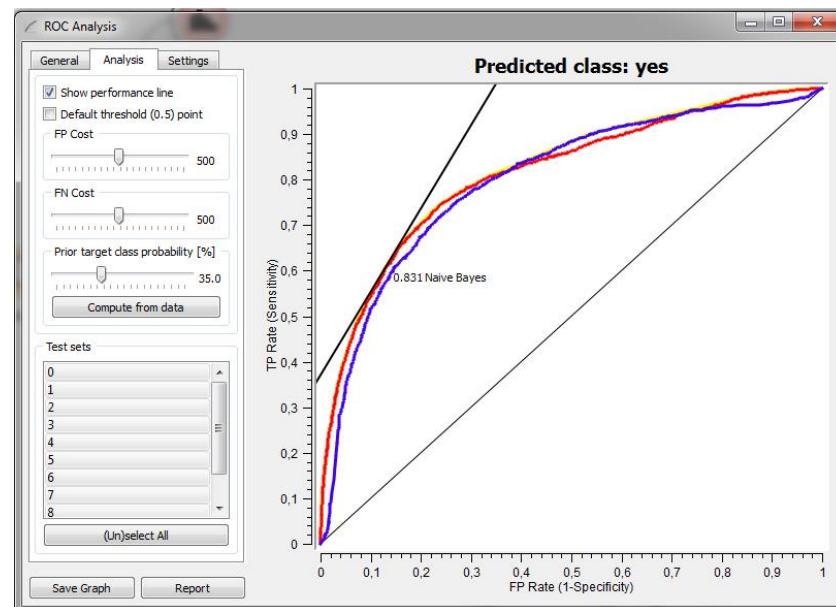
mera AUC (*Area Under the ROC Curve*) ... površina pod ROC krivuljo
primerjanje krivulj je lahko težavno... AUC povzema krivulje ROC v obliki številske informacije

ROC KRIVULJE

$$\text{senzitivnost} = TPR = \frac{TP}{P} = \frac{TP}{TP+FN}$$

$$1 - \text{specifičnost} = FPR = \frac{FP}{P} = \frac{FP}{TP+FN}$$

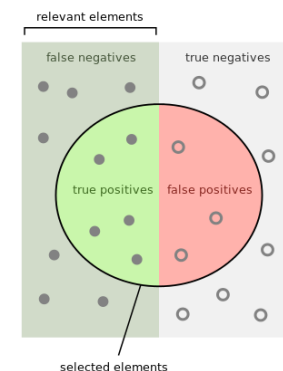
		dejanski razred		
		p	n	
napovedani razred	p'	TP	FP	P'
	n'	FN	TN	N'
		P	N	



(0,0) ... (TPR = 0, FPR = 0) → klasifikator, ki nikoli ne napove pravilnega razreda

(0,1) ... (TPR = 1, FPR = 0) → klasifikator, ki vedno napove pravilen razred

klasifikatorji na diagonalni → naključno ugibanje



ROC krivulje so neodvisne od P/N razmerja (kaj to pomeni?) ... glej stolpca v kontingenčni tabeli...

How many selected items are relevant?

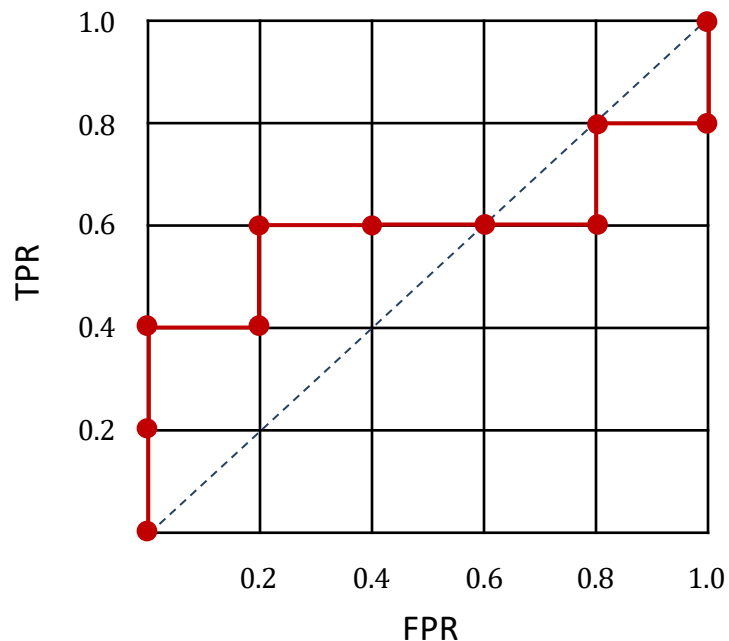
$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

RISANJE ROC KRIVULJ

Primer	razred	točke	TP	FP	TN	FN	TPR	FPR
		1.00	0	0	5	5	0	0
1	+	0.95	1	0	5	4	0.2	0
2	+	0.93	2	0	5	3	0.4	0
3	-	0.87	2	1	4	3	0.4	0.2
4	+	0.85	3	1	4	2	0.6	0.2
5	-	0.85	3	2	3	2	0.6	0.4
6	-	0.85	3	3	2	2	0.6	0.6
7	-	0.76	3	4	1	2	0.6	0.8
8	+	0.53	4	4	1	1	0.8	0.8
9	-	0.43	4	5	0	1	0.8	1
10	+	0.25	5	5	0	0	1	1



Potek risanja krivulje:

$(0, 0) \rightarrow (0.2, 0) \rightarrow (0.4, 0) \rightarrow (0.4, 0.2) \rightarrow (0.6, 0.2) \rightarrow (0.6, 0.2)$
 $\rightarrow (0.6, 0.4) \rightarrow (0.6, 0.6) \rightarrow (0.6, 0.8) \rightarrow (0.8, 0.8) \rightarrow (0.8, 1) \rightarrow (1, 1)$

interpretacija AUC

verjetnost, da ima naključni pozitivni primer več točk od naključno izbranega negativnega primera

KLASIFIKACIJSKA TOČNOST IN ROC KRIVULJE

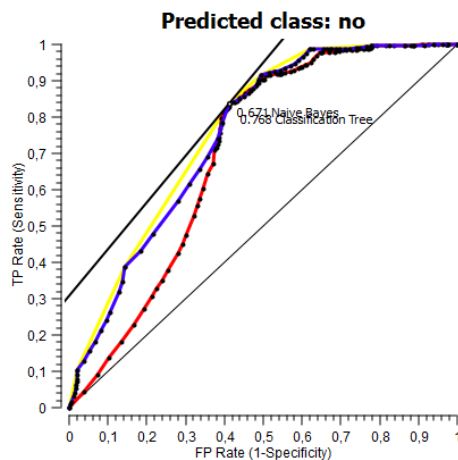
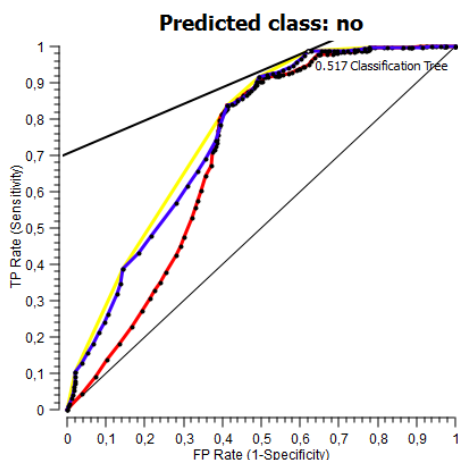
Kaj pa optimalna klasifikacijska točnost?

... celo verjetnostni klasifikator s popolno ROC ima optimalno klasifikacijsko točnost le v eni točki! levo zgoraj... (0,1)

$$\text{klasifikacijska točnost (CA)} = \frac{TP + TN}{P + N} = \frac{TPR * P + (1 - FPR) * N}{P + N}$$

izpeljemo črto, pri kateri so klasifikacijske točnosti enake:

$$TPR = \frac{CA * (P + N) - (1 - FPR) * N}{P} = \frac{N}{P} * FPR + \frac{CA * (P + N) - N}{P}$$

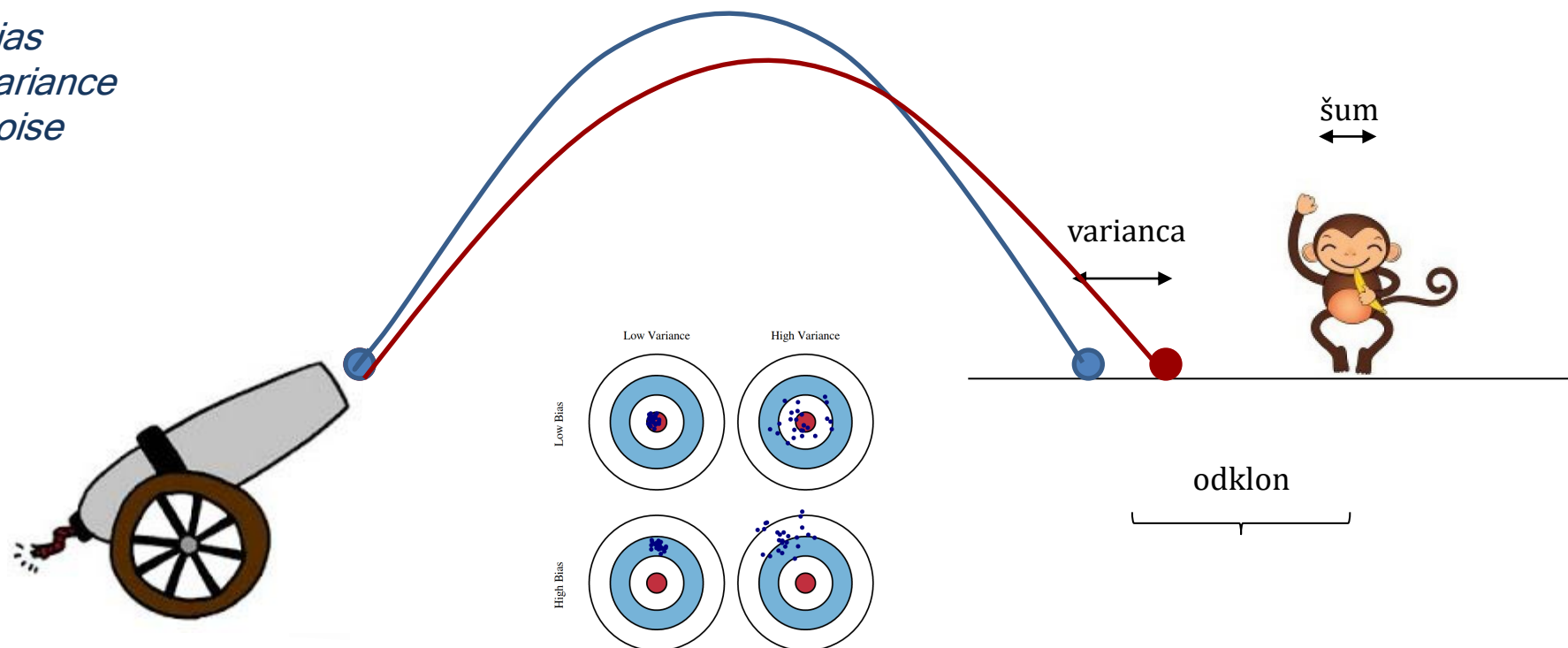


Kaj pa stroški FP in FN?

- dobimo paralelne črte, vsaka od njih predstavlja določeno klasifikacijsko točnost
- točka optimalne klasifikacijske točke je tista, kjer se srečata iso-performance tangenta in ROC krivulja
- želimo recept za določanje praga t , ki v obzir jemlje tudi stroške

ODKLON, VARIANCA, ŠUM

bias
variance
noise



varianca... napovedni model je morda preveč kompleksen

odklon... napovedni model je morda preveč enostaven

uporaba manj značilk

več in močnejše značilke

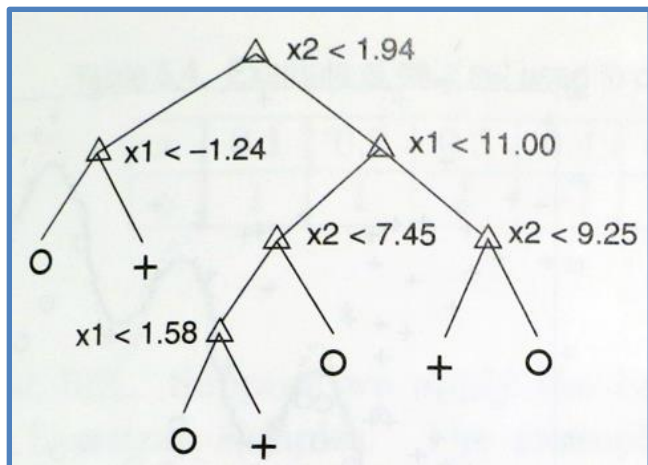
več podatkov za trening

povečati kompleksnost modela

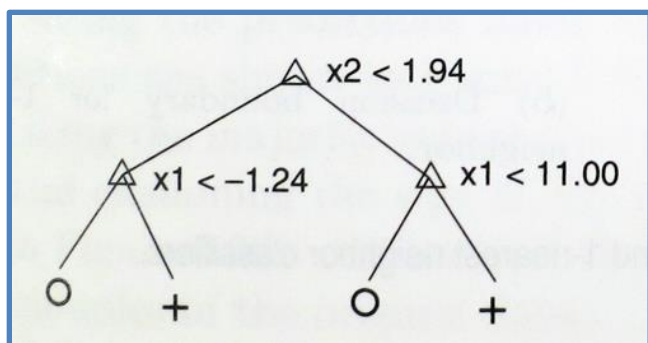
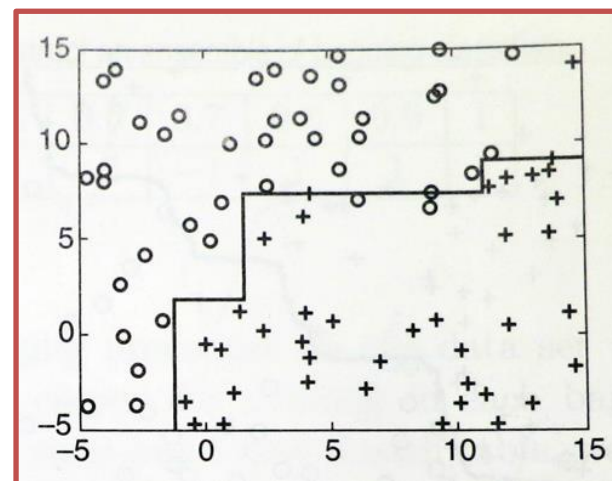
več regularizacije

manj regularizacije

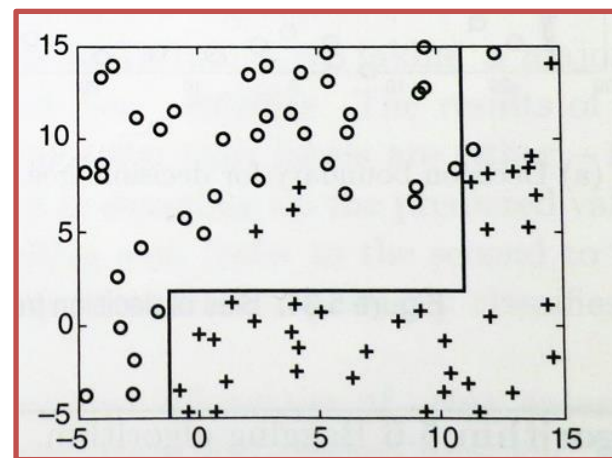
ILUSTRATIVEN PRIMER I



klasifikacijsko drevo

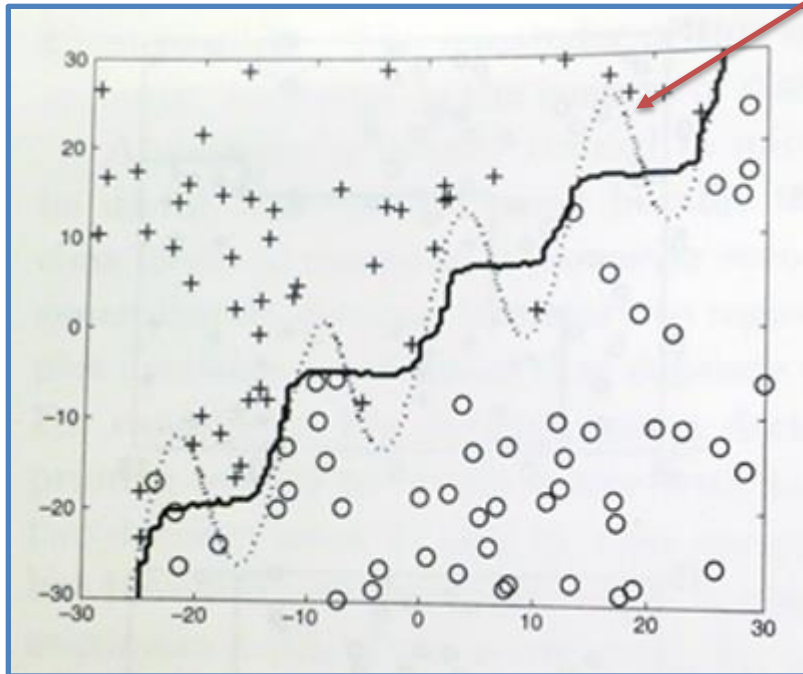


porezano klasifikacijsko drevo



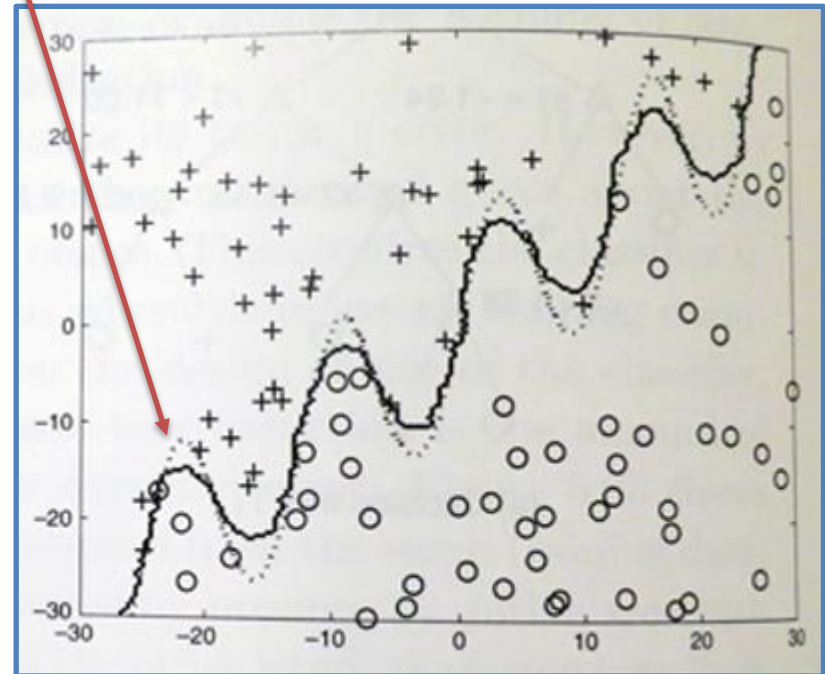
ILUSTRATIVEN PRIMER II

nižja varianca



resnična meja

nižji odklon



klasifikator: klasifikacijsko drevo

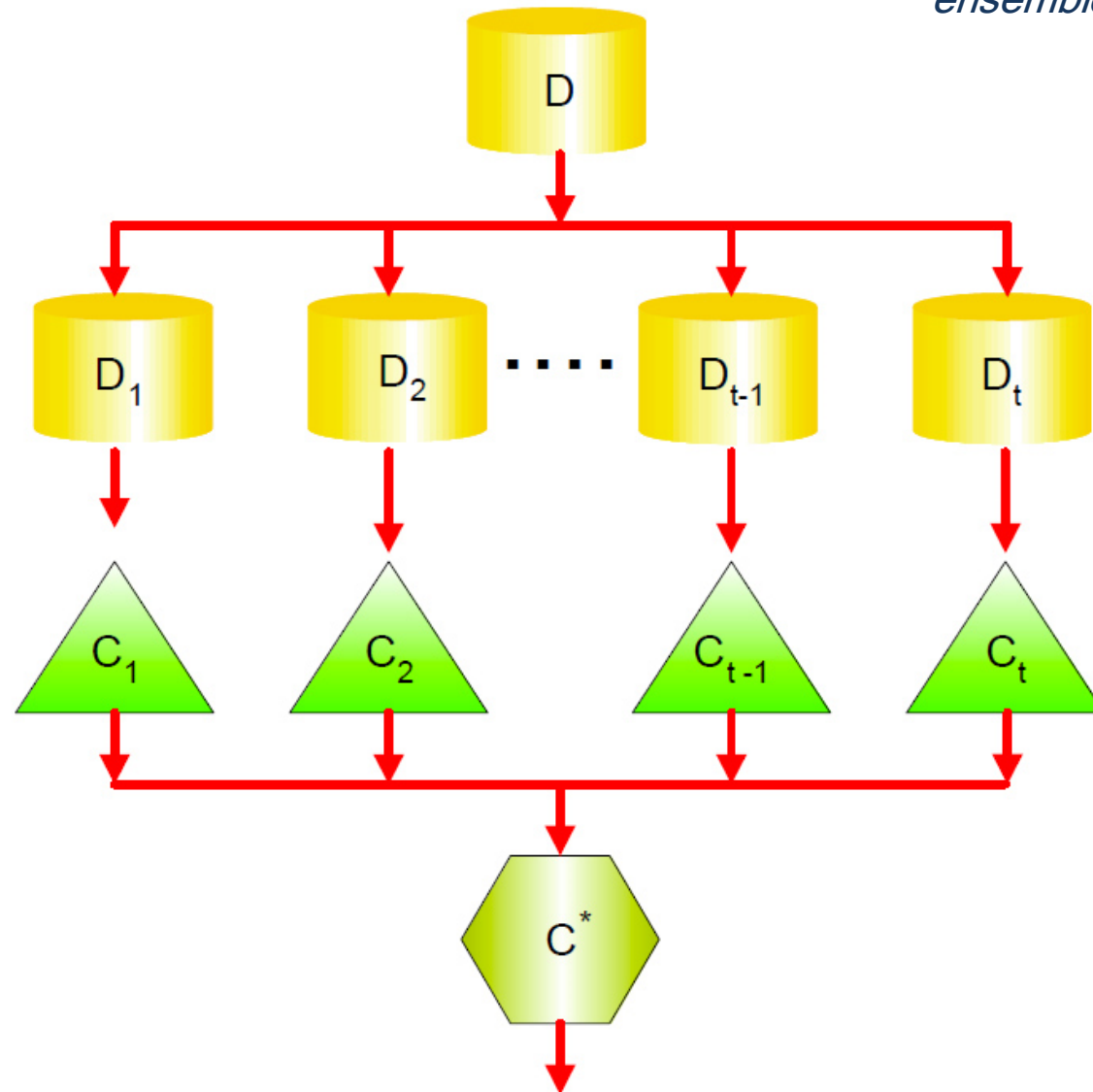
klasifikator: 1-najbližji sosed

100 učnih množic, vsaka vsebuje 100 učnih primerov, povprečenje modelov...

ustvarimo več množic
učnih podatkov

zgradimo več
klasifikatorjev

kombiniramo napovedi
klasifikatorjev



```
import orange, orngEnsemble, orngTree
import orngTest, orngStat

tree = orngTree.TreeLearner(mForPruning=2, name="tree")
bg = orngEnsemble.BaggedLearner(tree, name="bagged tree")
forest = orngEnsemble.RandomForestLearner(trees=50, name="forest")

data = orange.ExampleTable("marketing.tab")

learners = [tree, bg, forest]
results = orngTest.crossValidation(learners, data, folds=10)
print "Learner    CA    Brier  AUC"
for i in range(len(learners)):
    print "%-12s %5.3f %5.3f %5.3f" % (learners[i].name, \
        orngStat.CA(results)[i],
        orngStat.BrierScore(results)[i],
        orngStat.AUC(results)[i])
```

Learner	CA	Brier	AUC
tree	0.754	0.413	0.736
bagged tree	0.768	0.351	0.811
forest	0.778	0.314	0.837

$$\text{Brier} = \frac{1}{c} \sum_{i=1}^c (p_i' - p_i)^2$$

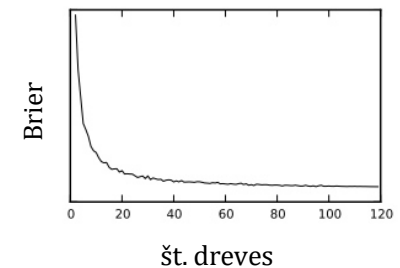
Metoda *bagging* (*bootstrap aggregating*)

- izkorišča nestabilnost (varianco) klasifikatorjev
- vzorčenje z zamenjavo oz. vračanjem („*bootstrap 0.632*“)
- napovedi klasifikatorjev združimo v skupno z večinskim glasovanjem

Naključni gozd

- kombinira napovedi več različnih odločitvenih dreves
- vsako drevo temelji na neodvisni množici naključnih vektorjev

Tipičen odnos med Brierjevo oceno (*Brier score*) in naključnim gozdom:



Naj bo x enodimenzionalni atribut, y pa razred:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
y	+	+	+	-	-	-	-	+	+	+

Za klasifikator uporabimo kar klasifikacijsko drevo globine 1:

- testni pogoj naj bo $x \leq k$,
- k naj bo vrednost, ki minimizira entropijo v listih tega drevesa

decision stump

Brez uporabe metode *bagging* bi učne primere razdelili pri $x \leq 0.35$ ali $x \leq 0.75$, v obeh primerih bi dosegli zgolj 70% klasifikacijsko točnost.

Postopek po korakih

- izvedemo vzorčenje z zamenjavo oz. vračanjem (*sampling with replacement*)
- na vsaki tako pridobljeni množici podatkov uporabimo klasifikator
- napovedi klasifikatorjev združimo v skupno z večinskim glasovanjem

METODA BAGGING: VZORČENJE IN KLASIFIKACIJA (PRIMER)

1

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	+	+	+	+	-	-	-	-	+	+

$x \leq 0.35 \rightarrow y = +$

$x > 0.35 \rightarrow y = -$

2

x	0.1	0.2	0.3	0.4	0.5	0.8	0.9	1	1	1
y	+	+	+	-	-	+	+	+	+	+

$x \leq 0.65 \rightarrow y = +$

$x > 0.65 \rightarrow y = +$

3

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	+	+	+	-	-	-	-	-	+	+

$x \leq 0.35 \rightarrow y = +$

$x > 0.35 \rightarrow y = -$

4

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
y	+	+	+	-	-	-	-	-	+	+

$x \leq 0.30 \rightarrow y = +$

$x > 0.30 \rightarrow y = -$

5

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
y	+	+	+	-	-	-	-	+	+	+

$x \leq 0.35 \rightarrow y = +$

$x > 0.35 \rightarrow y = -$

6

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
y	+	-	-	-	-	-	-	+	+	+

$x \leq 0.75 \rightarrow y = -$

$x > 0.75 \rightarrow y = +$

7

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
y	+	-	-	-	-	+	+	+	+	+

$x \leq 0.75 \rightarrow y = -$

$x > 0.75 \rightarrow y = +$

8

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	+	+	-	-	-	-	-	+	+	+

$x \leq 0.75 \rightarrow y = -$

$x > 0.75 \rightarrow y = +$

9

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1
y	+	+	-	-	-	-	-	+	+	+

$x \leq 0.75 \rightarrow y = -$

$x > 0.75 \rightarrow y = +$

10

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
y	+	+	+	+	+	+	+	+	+	+

$x \leq 0.05 \rightarrow y = -$

$x > 0.05 \rightarrow y = +$

METODA BAGGING: GLASOVANJE (PRIMER)

Iteracija	x = 0.1	x = 0.2	x = 0.3	x = 0.4	x = 0.5	x = 0.6	x = 0.7	x = 0.8	x = 0.9	x = 1.0
1	+	+	+	-	-	-	-	-	-	-
2	+	+	+	+	+	+	+	+	+	+
3	+	+	+	-	-	-	-	-	-	-
4	+	+	+	-	-	-	-	-	-	-
5	+	+	+	-	-	-	-	-	-	-
6	-	-	-	-	-	-	-	+	+	+
7	-	-	-	-	-	-	-	+	+	+
8	-	-	-	-	-	-	-	+	+	+
9	-	-	-	-	-	-	-	+	+	+
10	+	+	+	+	+	+	+	+	+	+
Vsota +	6	6	6	2	2	2	2	6	6	6
Vsota -	4	4	4	8	8	8	8	2	2	2
Razred	+	+	+	-	-	-	-	+	+	+

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
y	+	+	+	-	-	-	-	+	+	+

Kako in kdaj metoda *bagging* (ne) pomaga?

- zmanjšuje napako pri napovedih **s pomočjo zmanjševanja variance** osnovnega klasifikatorja
- če je osnovni klasifikator nestabilen, reducira napake povezane z naključnimi fluktuacijami v podatkih
- ne pomaga: če je osnovni klasifikator odporen na perturbacije v podatkih, vzrok napak pa je odklon

- skupina **klasifikacijskih dreves** pri klasifikaciji primera v razred glasuje
- napovemo razred, kamor primer uvrsti večina dreves
- zaželena je variabilnost med drevesi (jih dodatno „ponaključimo“)



- ena od najbolj zanesljivih tehnik uvrščanja, tipično višje napovedne točnosti
- slabost: izgubljena možnost interpretacije modela

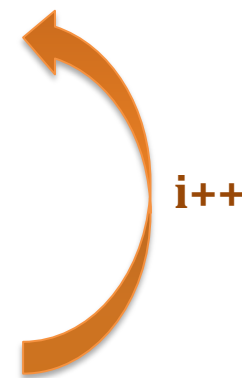
1. Pričnemo z učno množico D , ki vsebuje N primerov in M atributov.

Izberemo parametre...

2. Izberemo število m : med koliko naključno izbranih atributov iz primerov v vozliščih drevesa bomo izbrali najboljšega ... $m \ll M$, npr. $m = \sqrt{M}$
3. Izberemo tudi število dreves K v gozdu ... npr. $K = 100$

... in „gradimo“ gozd

4. Uporabimo metodo stremena (*bootstrap*): med N primeri učne množice D naključno z vračanjem izberemo novo množico primerov ... dobimo D_i
5. Na množici primerov D_i zgradimo klasifikacijsko drevo T_i :
 - v vsakem vozlišču naključno izberemo in uporabimo m atributov
 - med njimi za delitev teh primerov izberemo najbolj informativnega
 - dreves ne režemo... ali to pomaga pri reduciranju variance ali odklona?



UGOTAVLJANJE POMEMBNOСТИ ZNAČILK Z NAKLJUČNIM GOZDOM

```
import orange, orngEnsemble, random

data = orange.ExampleTable("marketing.tab")

measure = orngEnsemble.MeasureAttribute_randomForests(trees=50)

#call by attribute index
imp0 = measure(0, data)
#call by orange.Variable
imp1 = measure(data.domain.attributes[1], data)
print "first: %0.2f, second: %0.2f\n" % (imp0, imp1)

print "different random seed"
measure = orngEnsemble.MeasureAttribute_randomForests(trees=50, rand=random.Random(10))

imp0 = measure(0, data)
imp1 = measure(data.domain.attributes[1], data)
print "first: %0.2f, second: %0.2f\n" % (imp0, imp1)

print "All importances:"
imps = measure.importances(data)
for i,imp in enumerate(imps):
    print "%17s: %6.2f" % (data.domain.attributes[i].name, imp)
```

```
first: 0.30, second: 4.29

different random seed
first: 0.12, second: 3.35

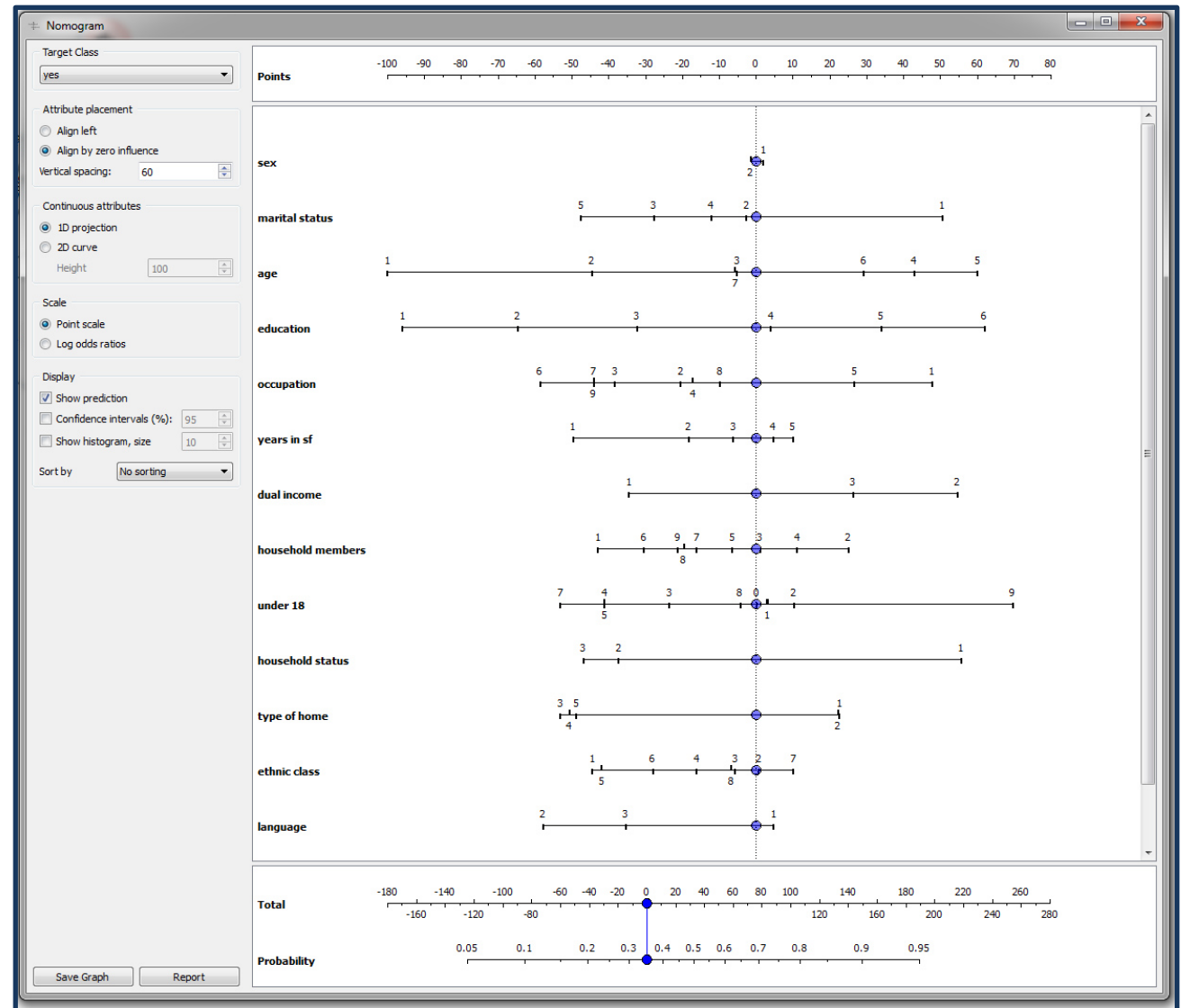
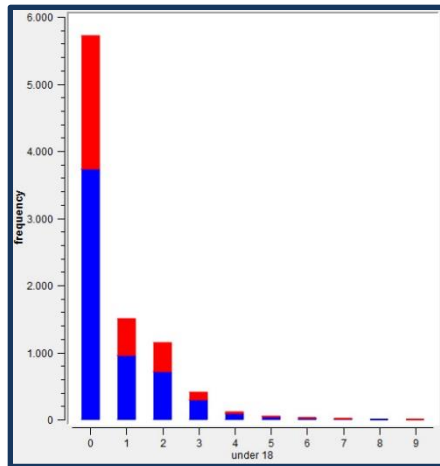
All importances:
           sex: 0.12
    marital status: 3.35
           age: 1.17
      education: 2.04
    occupation: 1.86
    years in sf: 0.04
    dual income: 2.62
household members: 0.14
        under 18: -0.01
    household status: 6.61
    type of home: 0.75
    ethnic class: 0.04
        language: 0.13
```

- drevo T_i smo izgradili iz učnih primerov D_i , ostali primeri so D'_i
- naj bo C_i število primerov iz D_i , za katere je drevo T_i pravilno napovedalo razred
- za vsak atribut X premešajmo njegove vrednosti v D_i
- na tej spremenjeni množici izmerimo število pravilno napovedanih $C_{i,X}$

$$RI(X) = \frac{1}{K} \sum_{i=1}^K C_i - C_{i,X}$$

zakaj?

NOMOGRAM



<http://orange.biolab.si/datasets/marketing.htm>

Kateri atribut je najmanj uporaben? Kateri atributi so najbolj uporabni?

UGOTAVLJANJE POMEMBNOСТИ ZNAČILK: *GRADIENTBOOSTING TREES*

```
# ensembles
ensembles = []
ensembles.append(('AB', AdaBoostClassifier()))
ensembles.append(('GB', GradientBoostingClassifier()))
ensembles.append(('RF', RandomForestClassifier()))
ensembles.append(('ET', ExtraTreesClassifier()))
```

```
results = []
names = []
for name, model in ensembles:
    kfold = KFold(n_splits=num_folds, random_state=seed, shuffle=False)
    cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
AB: 0.777778 (0.059252)
GB: 0.823611 (0.039308)
RF: 0.805556 (0.031056)
ET: 0.787500 (0.036244)
```

```
# prepare the model
```

```
model = GradientBoostingClassifier() # GB
model.fit(X_train, Y_train)
```

```
GradientBoostingClassifier(criterion='friedman_mse', init=None,
                           learning_rate=0.1, loss='deviance', max_depth=3,
                           max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=100,
                           presort='auto', random_state=None, subsample=1.0, verbose=0,
                           warm_start=False)
```

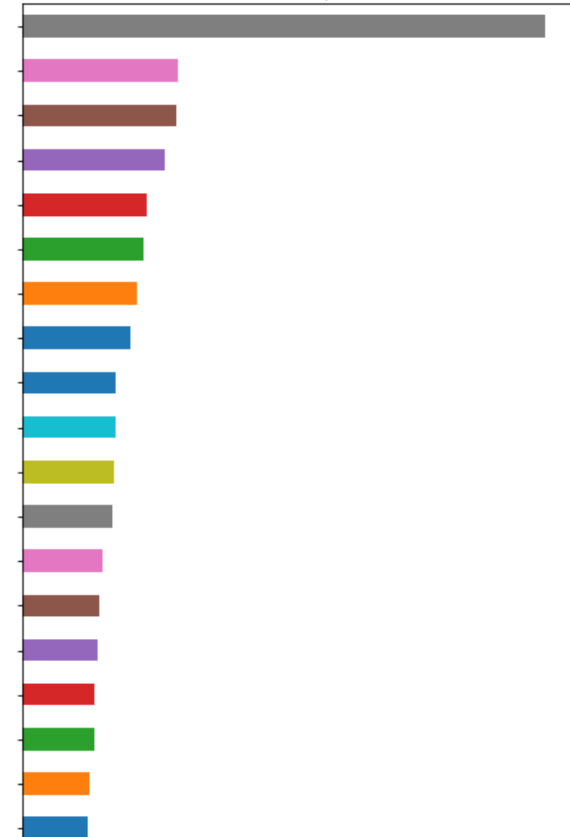
```
importance = dict(zip(feature_names, model.feature_importances_))
```

```
importance = sorted(importance.items(), key=operator.itemgetter(1))
importance.reverse()
```

```
df_importance = pd.DataFrame(importance, columns=['feature', 'score'])
```

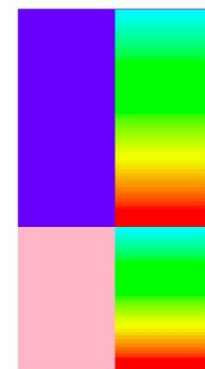
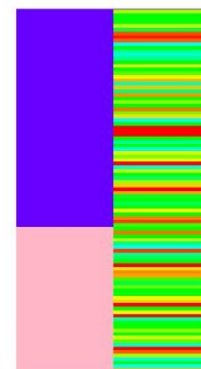
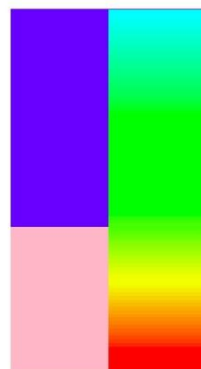
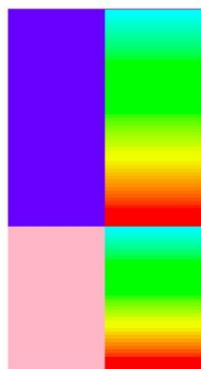
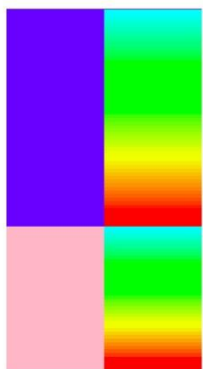
```
df_importance['score'] = df_importance['score'] / df_importance['score'].sum()
```

GB Feature Importance



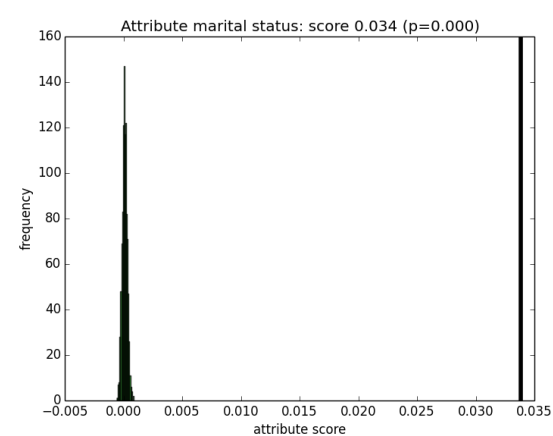
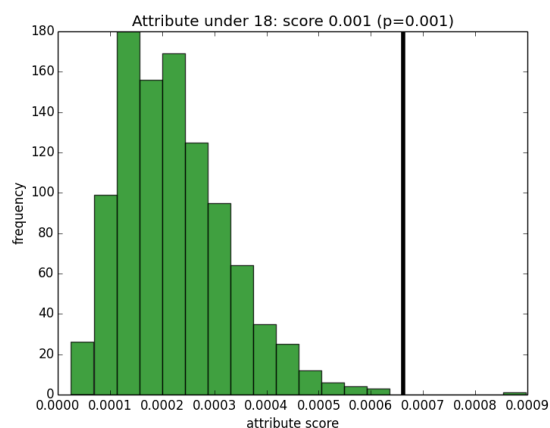
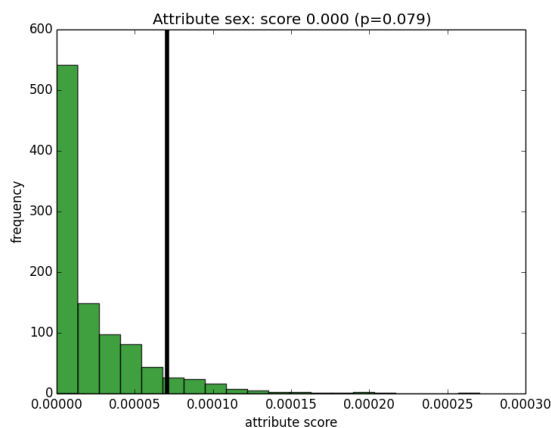
PERMUTACIJSKI TEST

ničelna hipoteza: premešati vrednosti atributa ne bo vplivalo na napovedi



null = true

null = false



- Tan P.-N., Steinbach M. in Kumar V. *Introduction to Data Mining*, Pearson Addison Wesley, 2006.

<http://www-users.cs.umn.edu/~kumar/dmbook/>

Classification (četrto poglavje)

Classification: Alternative techniques (peto poglavje)

- Witten, Ian H., and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*.

Morgan Kaufmann, 2005.

- Orange: Open source data visualization and analysis for novice and experts.

<http://orange.biolab.si/>



- scikit-learn: Machine Learning in Python

<https://scikit-learn.org/>

