Osztályozás

Csima Judit

BME, VIK,
Számítástudományi és Információelméleti Tanszék

2017. február 20.

Osztályozás, classification

- adott egy rekordokból álló halmaz, a rekordoknak attribútumaik vannak
- az egyik attribútum a célváltozó, ez kategorikus attribútum, ez reprezentálja, hogy melyik osztályba tartozik az adott rekord
- cél, hogy egy olyan modellt építsünk fel, ami képes megjósolni a célváltozó értékét, ha a többi attribútum értéke adott

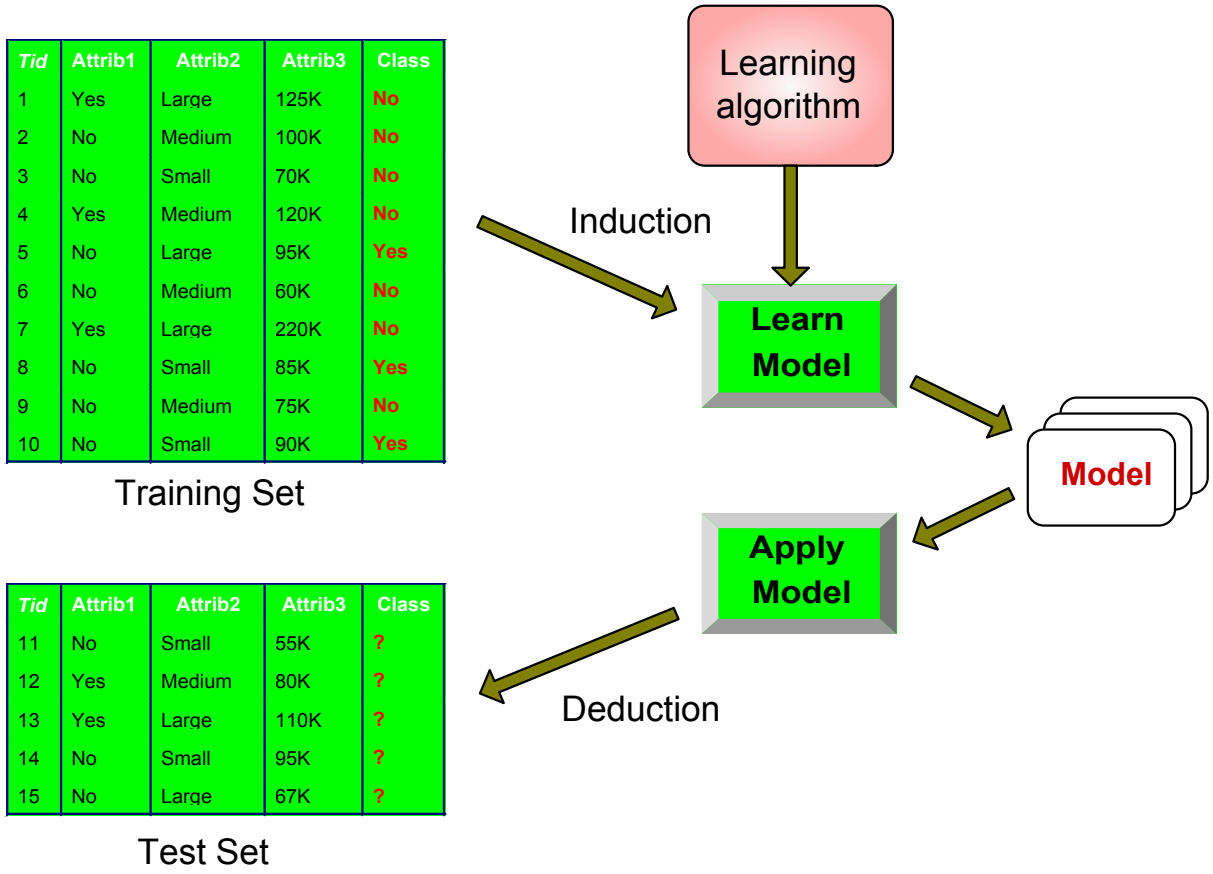
Példák

- egy beteg paraméterei alapján eldönteni, hogy jó- vagy rosszindulatú daganata van-e
- bankkártyás tranzakció adatait vizsgálva eldönteni, hogy van-e csalás vagy nincs (szabályos-e a tranzakció)
- adóbevallásban szereplő értékek alapján megtippelni, hogy gyanús-e
- spam-szűrés

Általános módszer

- van egy csomó rekordunk, ahol minden érték (a célváltozó értéke is) ismert
- ez alapján egy modellt építünk, amit majd olyan új rekordokon használunk, amiknél a célváltozó értéke nem ismert, de minden más attribútumot tudunk
- az ismert rekordokat két részre osztjuk: training set és test set
- a training set-en betanítunk valami modellt
- a test set-en lemérjük, hogy mennyire jó
- ezután használhatjuk élesben

Illustrating Classification Task



Kérdések

- Milyen modellek vannak?
  - Döntési fák
  - Bayes-osztályozók
  - Mesterséges neurális hálózatok
  - (SVM: Support Vector Machine)
- hogyan állítunk elő egy konkrét modellt? Pl. ha már tudjuk, hogy döntési fát csinálunk, akkor hogyan csináljuk meg; vagy egy ANN-nél hogyan állítjuk be a paramétereket?
- Hogyan mérjük az előállított modell jóságát?
  - accuracy: eltalált címkék száma osztva az összes sor számával
  - error-rate: hibás predikciók száma osztva az összes sor számával

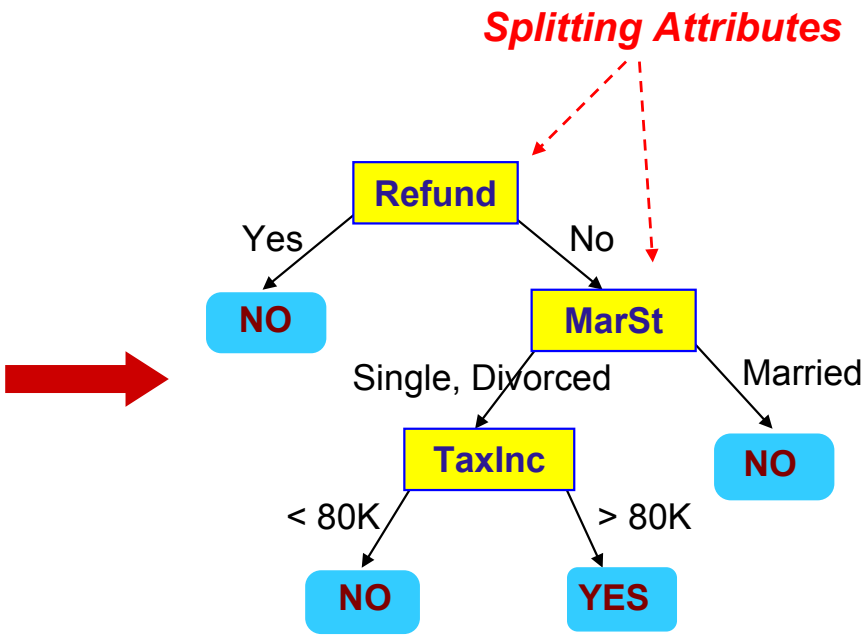
Döntési fa definíció

- gyökeres, lefelé irányított (legtöbbször) bináris fa
- belső csúcsok változókkal és ezekhez kapcsolódó feltételekkel címkézettek
- levelek a célváltozó valamely értékével címkézettek

Example of a Decision Tree

Tid	Refund	Marital Status	Taxable Income	Cheat
		categorical	*categorical*	*continuous*	*class*
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

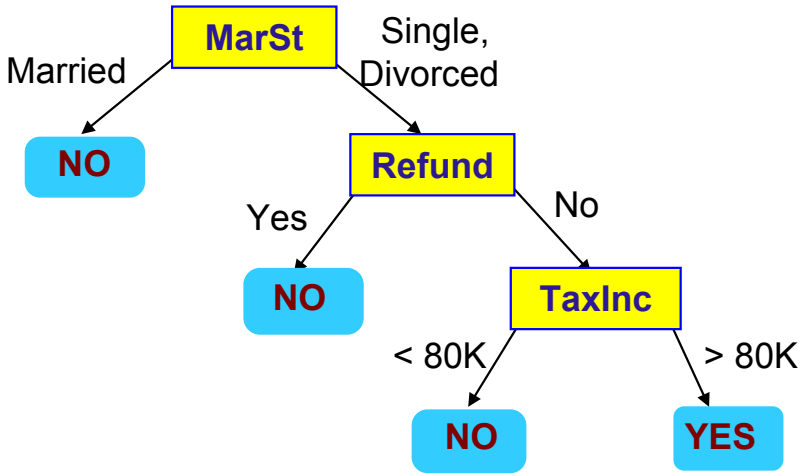
Training Data



Model: Decision Tree

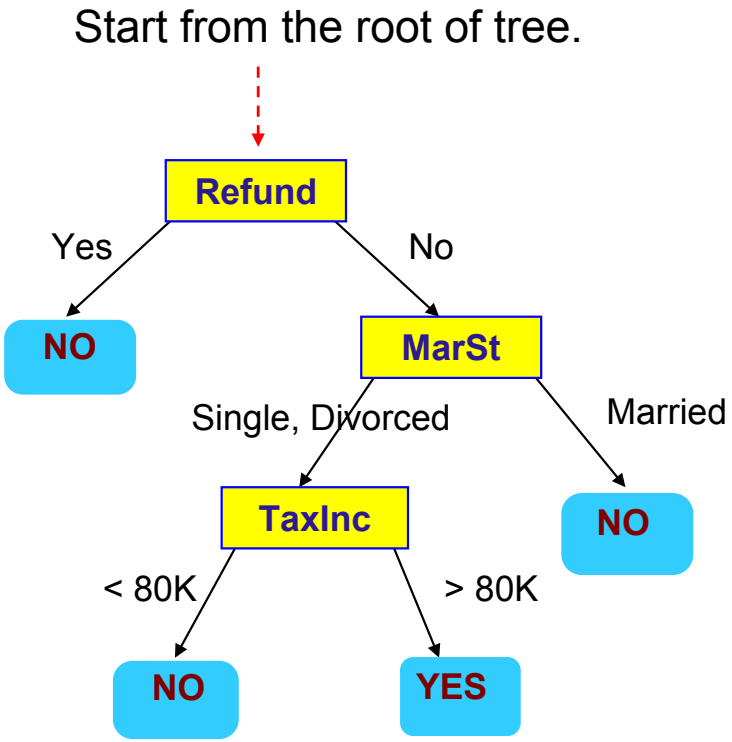
Another Example of Decision Tree

	categorical	categorical	continuous	class
Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



There could be more than one tree that fits the same data!

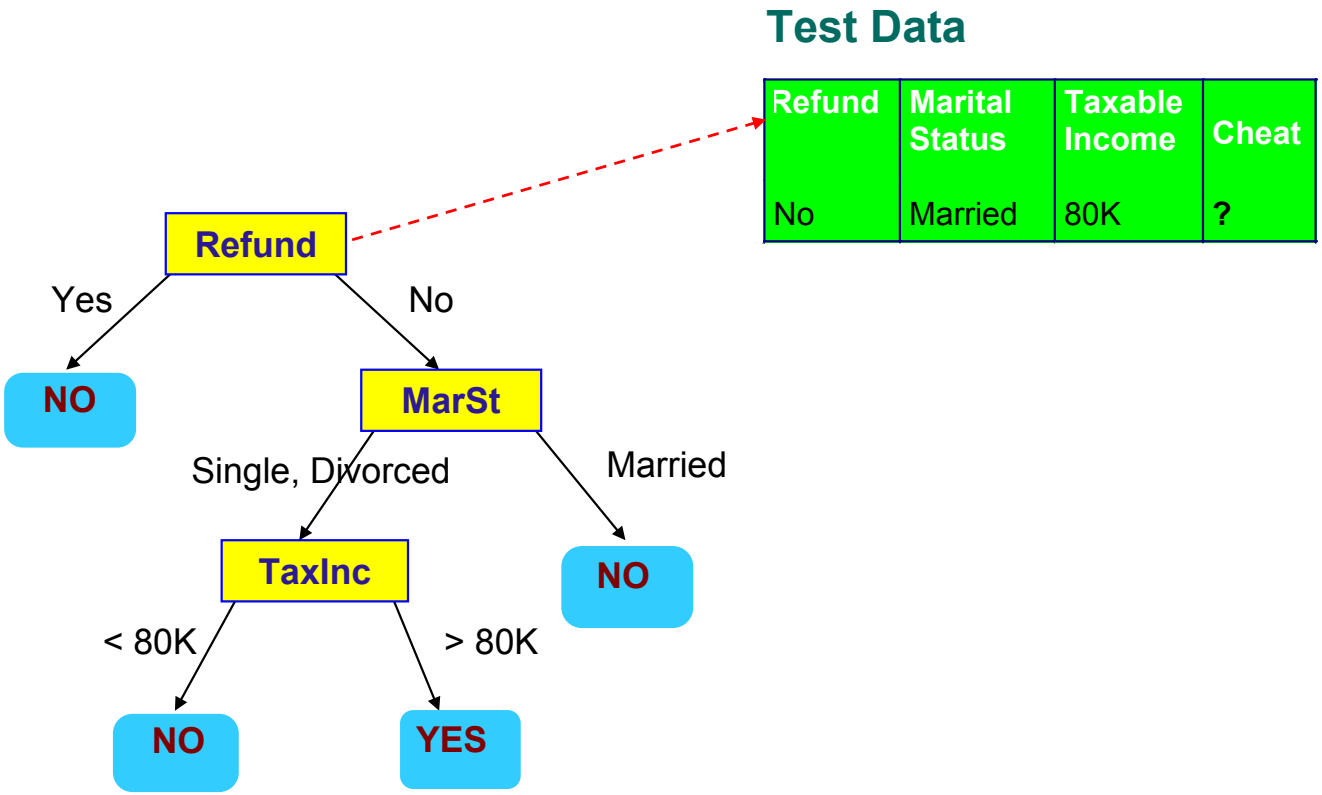
Apply Model to Test Data



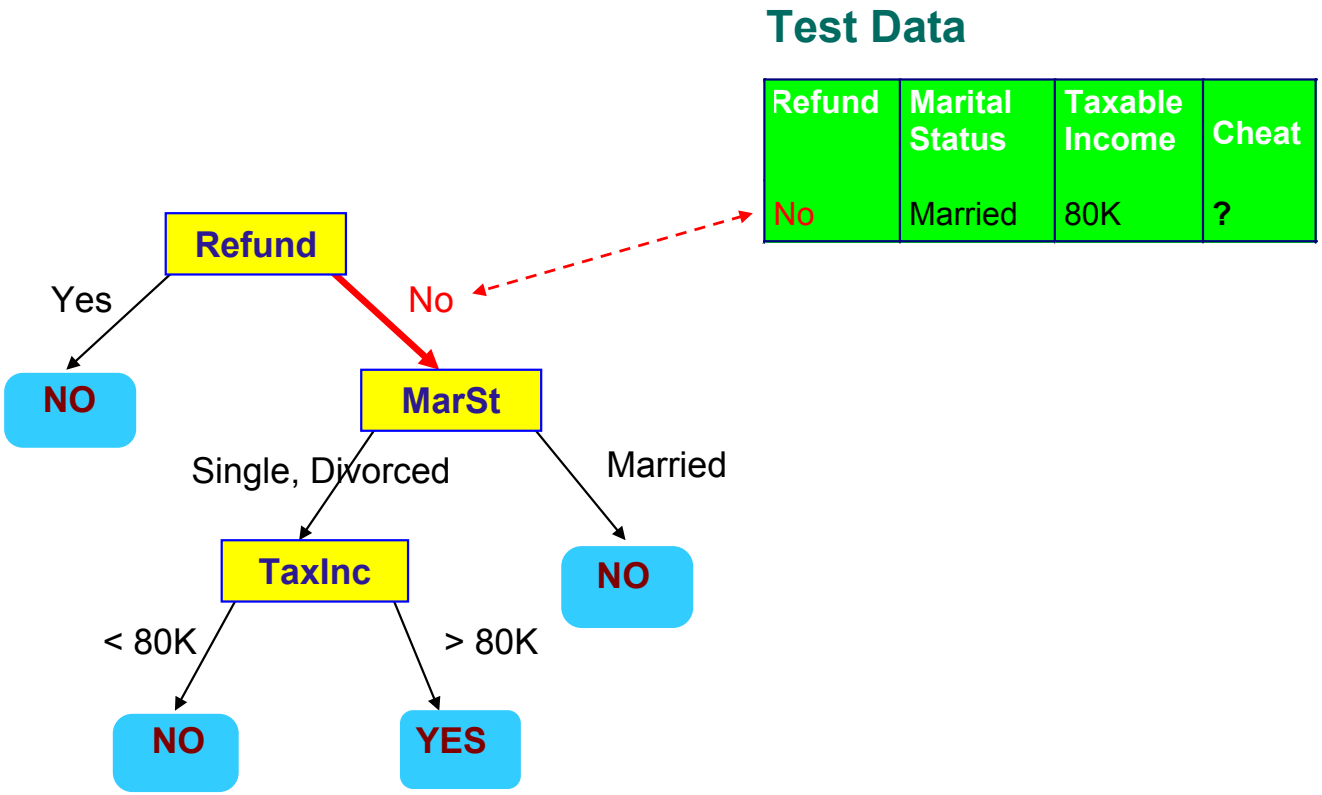
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

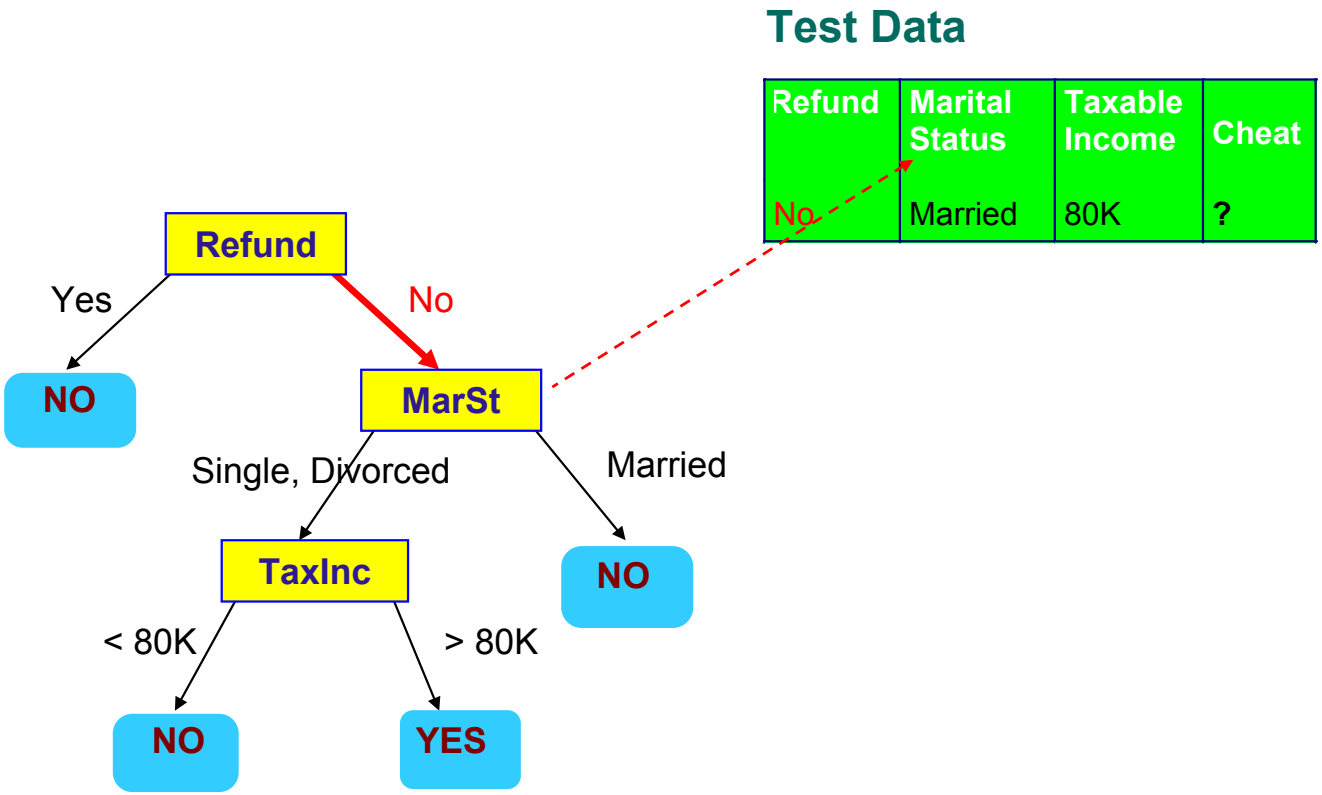
Apply Model to Test Data



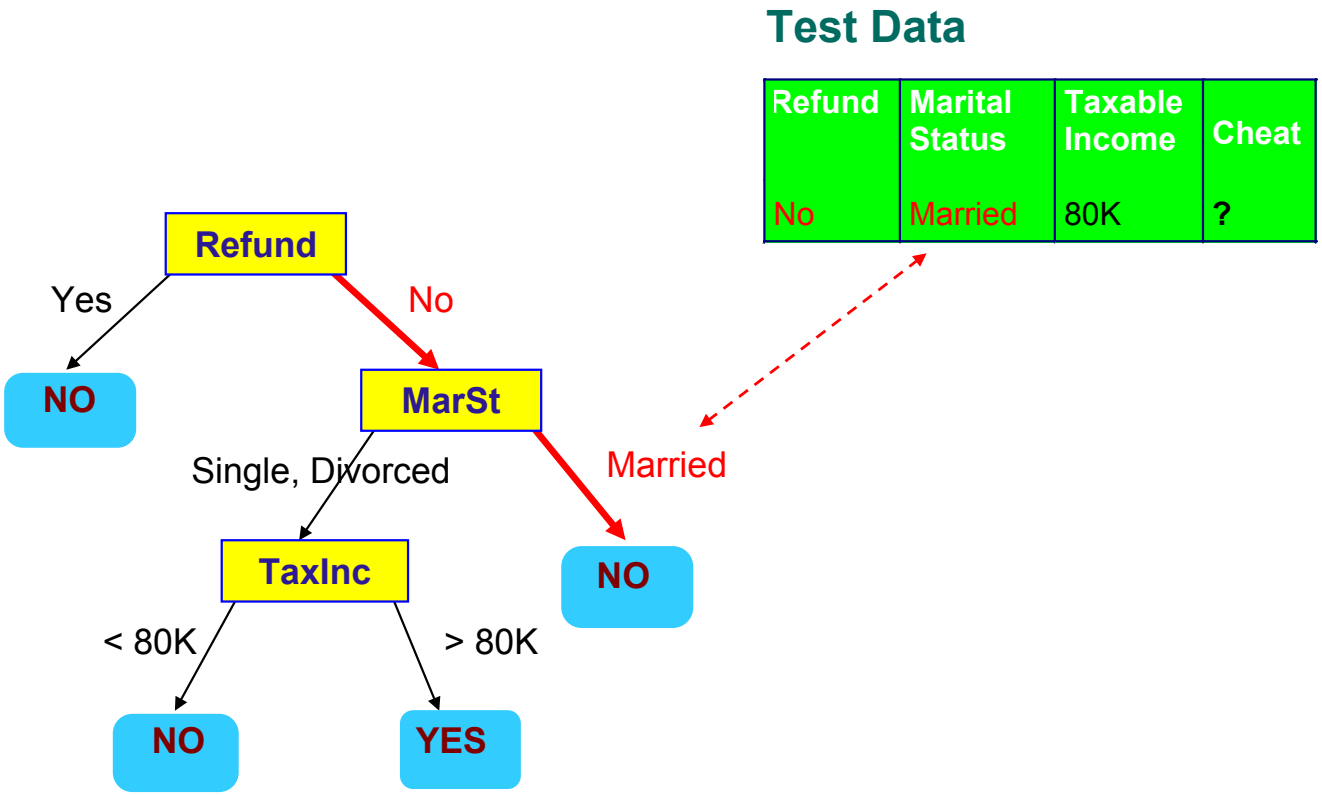
Apply Model to Test Data



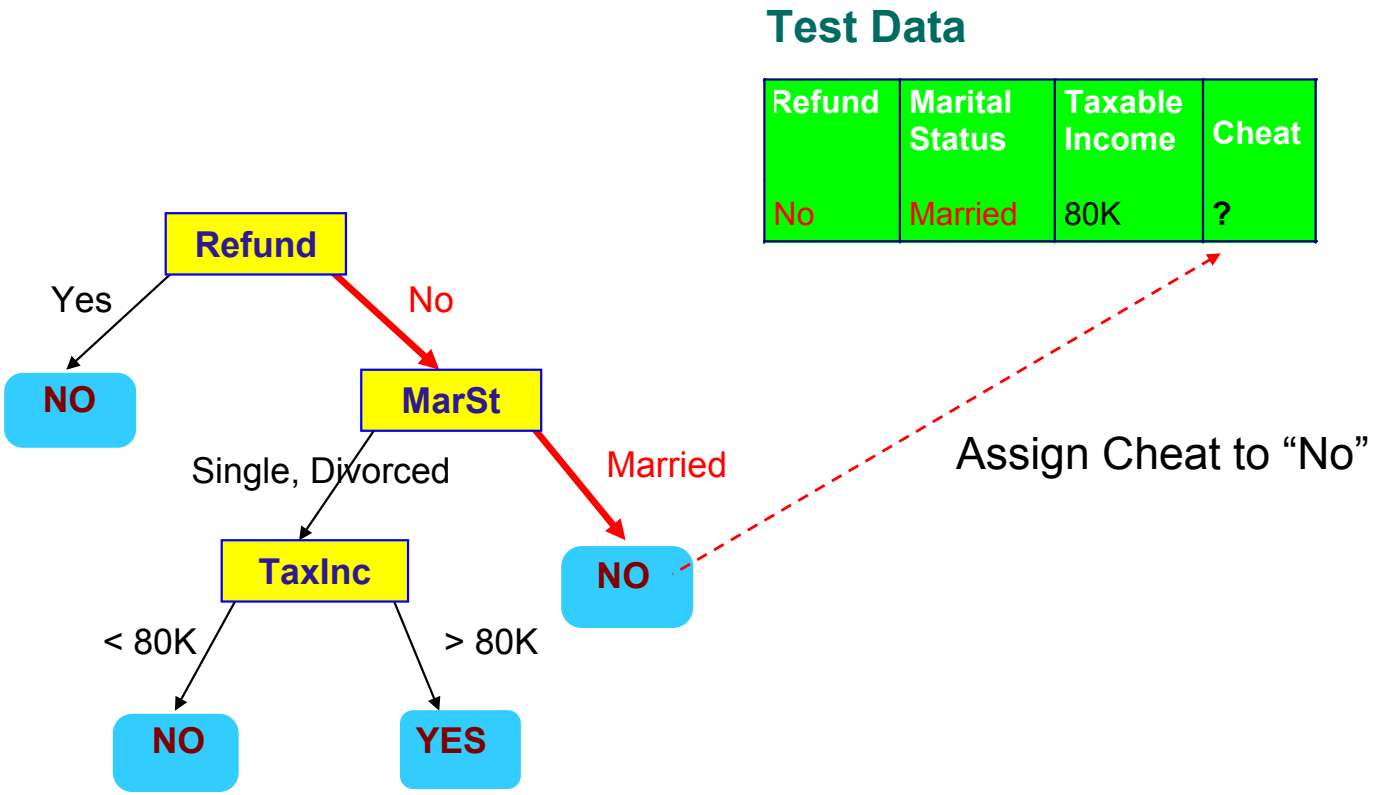
Apply Model to Test Data



Apply Model to Test Data



Apply Model to Test Data



Algoritmus döntési fa készítésére

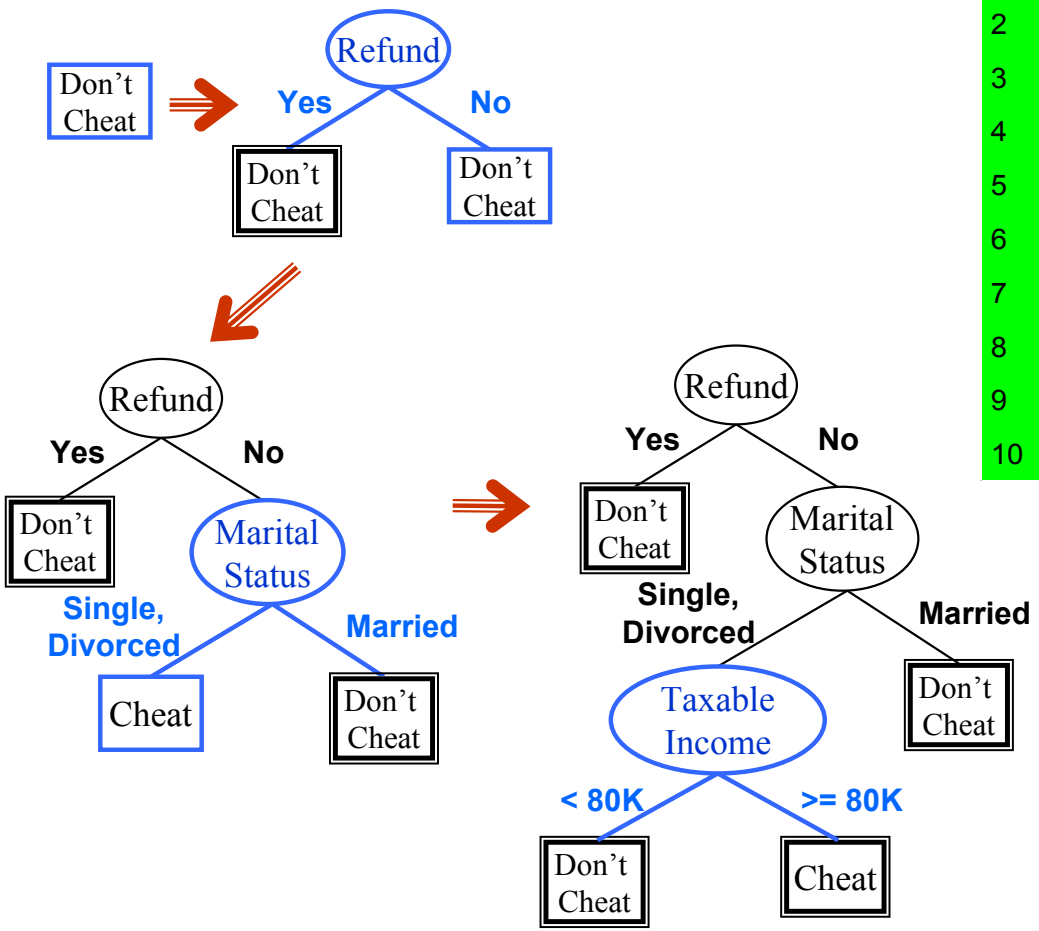
- egy általános algot nézünk, ennek különböző verziói futnak különböző programokban
- nem a legjobb fát akarjuk megtalálni (ez amúgy is aluldefiniált fogalom), hanem egy elég jót
- mohó módon, lokális döntéseket hozva, gyorsan

Hunt algoritmus, vázlat

- elején egy csúcs, ide tartozik minden rekord, címke a többségi címke
- később: választunk egy csúcsot, amit érdemes lenne szétvágni és valami attribútum mentén szétvágjuk egy vagy több részre
- vége: ha már nem érdemes vágni sehol

Hunt's Algorithm

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Hunt algo, kérdések

- mikor érdemes vágni?
- melyik csúcsot vágjuk, ha több lehetőség is van?
- hogyan vágunk?
- az új csúcsokat hogyan címkézzük?
- mikor van vége?

Hunt algo, kérdések

- Mikor van vége?
  - Ha már nincs olyan csúcs, amit vágni érdemes.
- Mikor nem érdemes vágni?
  - Ha nem akarunk tovább osztani: minden rekord azonos cél-címkéjű
  - Ha nem tudunk tovább osztani: olyan sorok vannak különböző címkével, amiknek minden más attribútuma megegyezik
- Melyik csúcsot vágjuk, ha több lehetőség is van?
  - Valami bejárás szerint, pl. szélességi, mélységi.
- Hogyan vágunk?
  - Erről mindárt, ez érdekes.
- Az új csúcsokat hogyan címkézzük?
  - Többségi címkével.

Milyen attribútum mentén és hogyan vágjunk?

Fő elv: sokféle vágást kipróbálunk és a legjobb szerint vágunk.

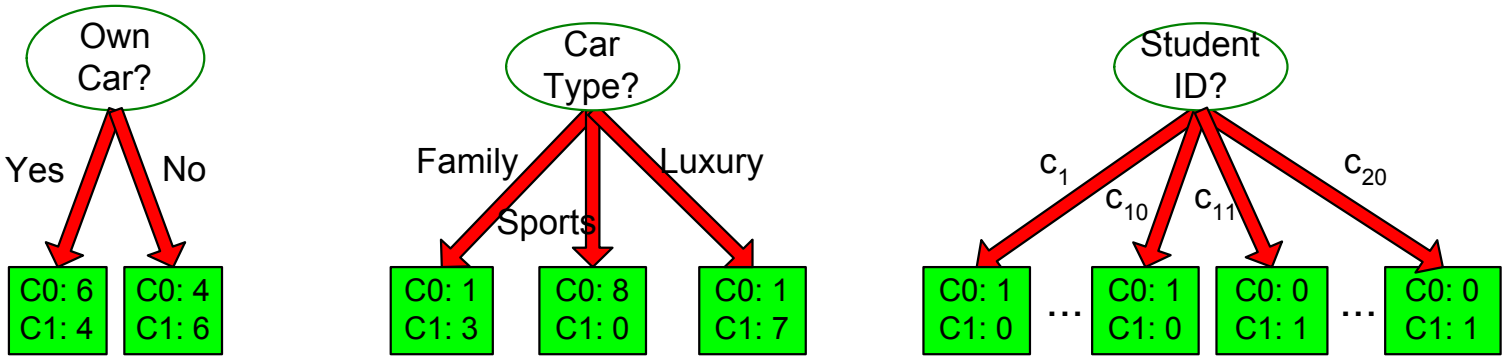
- Mik a lehetséges vágások? (Függ a szóba jövő attribútumok típusától.)
- Mi egy vágás jóságának a mértéke? Hogyan mérjük ezt?

Lehetséges vágások az attribútum fajtája szerint

- bináris attribútum: igen vagy nem, két gyerek csúcs lesz
- kategória típusú attribútum:
  - multiway split: minden lehetséges értékhez egy gyerek, az üres csúcsok címkéja a szülő többségi címkéje lesz
  - bináris split részhalmaz szerint: ebből van $2^t - 1$, ahol a t a lehetséges kimenetelek száma
  - bináris vágás egy érték szerint: ebből van t darab (mint marital status az előző példában)
- folytonos attribútum
  - bináris vágás, az attribútum értéke kisebb-e egy adott küszöbnél (mint income az előző példában)
  - többes vágás: melyik sávba esik az érték

How to determine the Best Split

Before Splitting: 10 records of class 0,
10 records of class 1



Which test condition is the best?

Vágás jósága, alapelvek

- többféle mérőszám van, mindegyik egy számértéket rendel a vágáshoz
- így a különböző vágások összehasonlíthatóak
- nagyrészt konzisztensek egymással a különféle mérőszámok
- mindegyik azt méri, hogy mennyire lesz homogén a létrejövő gyerek poluláció a célváltozó címkézése szerint

Vágás jósága

- 3 fő mérőszámot nézünk
- fő elv ugyanaz mindegyiknél:
  - egy rekordhalmazra definiálunk egy mérőszámot, ami az adott rekordhalmaz diverzitását mutatja (ebből lesz három féle mérőszám)
  - egy vágás jóságát azzal mérjük, hogy a szülőcsúcs diverzitása és a létrejövő gyerekcsúcsok diverzitása mennyire tér (mennyit nyerünk azon, ha szétvágjuk a szülőt egy adott módon, mennyivel lesznek homogénebbek a gyerekek)

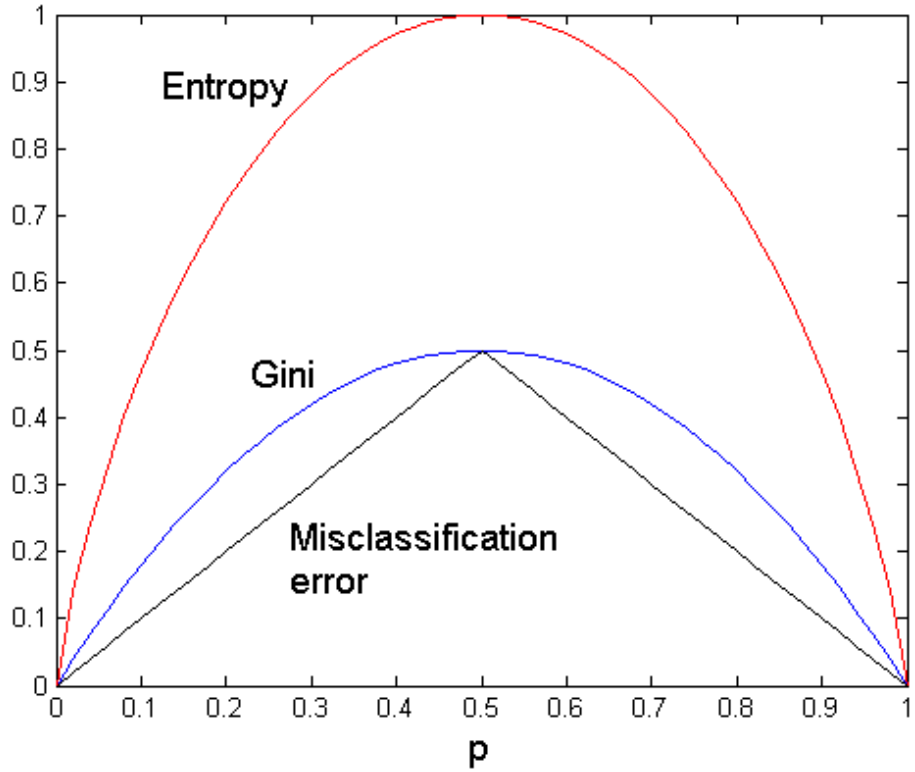
először azt nézzük, hogy hogyan lehet mérni egy rekordhalmaz, azaz a döntési fa egy csúcsának imhomogenitását

Inhomogenitás mérése: Gini-index, entrópia, classification error

- van egy t csúcsunk (egy rekordhalmaz), aminek egy c darab lehetséges értéket felvevő célváltozó szerinti homogenitását akarjuk mérni
- p_i jelöli a rekordhalmazban előforduló i értékű rekordok relatív gyakoriságát
- $\text{Gini} = 1 - \sum_{i=1}^{c} {p_i}^2$
- $\text{entrópia} = -\sum_{i=1}^{c} p_i \log p_i$
- $\text{classification error} = 1 - \max_{i \in 1,\ldots,c} p_i$

Comparison among Splitting Criteria

For a 2-class problem:



 Introduction to Data Mining 4/18/2004 44

Vágás jósága

- ha már eldöntöttük, hogy melyik csúcsnál vágunk
- az adott csúcsnál minden lehetséges attribútum alapján, minden (?) lehetséges módon vágunk
- $\Delta = I(\text{szülő}) - \sum_{i=1}^{k} \frac{n_i}{n} I(gyerek_i)$
- itt $I()$ a három inhomogenitási mérték közül az egyik
- n_i az i. gyerek rekordszáma, n a szülő rekordszáma
- arról van szó tehát, hogy a gyerekek inhomogenitását súlyozzuk a relatív nagyságukkal

ID alapján vágni?

- Azonosító alapján vágni (vagy más, nagyon kis elemszámú részhalmazra vágás) nem szerencsés:
  - ID esetén ez nem valódi nyereség
  - túl kicsi létrejövő halmazok esetén félő, hogy rosszul általánosít a modell (overfitting, erről mindjárt)
- de az algo ezt preferálja, mert itt lesz nagy a nyereség
- megoldások:
  - csak bináris vágás lehetséges
  - szűrjük ki a nyilvánvalóan felesleges attribútumokat (amik alapján enm akarunk úgyse vágni)
  - Δ_{info} helyett valami mással mérni a vágás jóságát: gain ratio

Gain ratio

- cél: büntessük azt, ha egy vágás túl sok részre vág szét
- ha entrópia az inhomogenitás mértéke: gain ratio
- $$\text{gain ratio} = \frac{\Delta_{info}}{-\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}}$$
- ez bünteti a szétszabdalást kicsi részekre

Az általános faépítő algo jellemzői

- gyorsan
- jól értelmezhető fát készít
- gyors az elkészült fával az osztályozás
- nem kell paramétereket beállítanunk előre (de ....)

Különböző programokban ennek verziói futnak:

- hogyan járjuk be a vágandó csúcsokat
- mi az inhomogenitás mértéke
- van-e multivágás vagy csak bináris

Leállási feltételek

- ha minden csúcs homogén
- ha nem tudunk tovább differenciálni
- valami globális leállási feltétel: levélszám, szintszámra korlát

A legjobb fa keresése nem kivitelezhető és nem is célszerű általában.

Overfitting

- nem szerencsés, ha a modell (pl. az elkészült döntési fa) túlságosan passzol a training set-re
- azért, mert nincs rá garancia, hogy a későbbi halmazok, amiken a modellt használjuk, teljesen ugyanilyenek lesznek (sőt...)
- egy egyszerűbb modell, aminek a training error-ja nagyobb, néha jobban általánosítható: jobban viselkedik a későbbi (a modell építése során nem látott) esetekre
- ez nem csak döntési fáknál léztező jelenség, hanem mindenhol, ahol egy training set alapján modellt építek, amit aztán korábban nem látott eseteken akarok használni

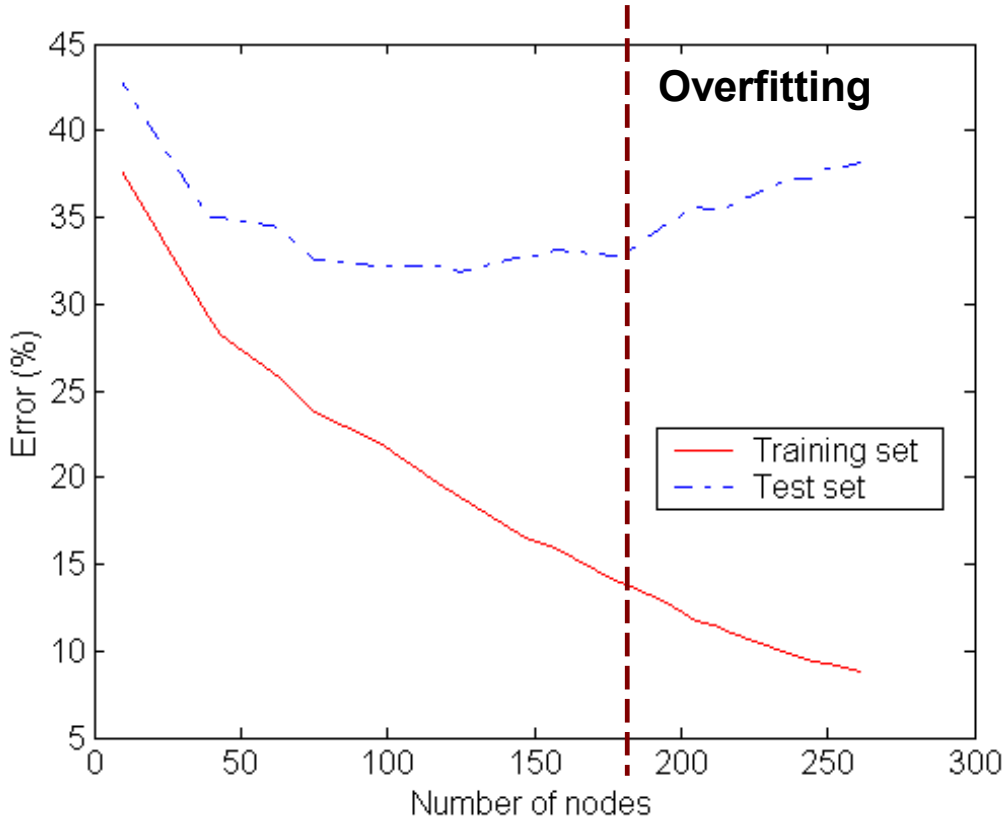
Training és test error

- felépítünk egy modellt (pl. egy döntési fát) aszerint, hogy mekkora a training error
- training error: a felépített modell hogyan osztályozza a training set rekordjait: accuracy, misclassification error
- de nekünk az lesz az érdekes, hogy a nem látott eseteken hogy viselkedik, hogy jelez előre
- ez lemérhető a teszt halmazon, de mi van, ha az derül ki, hogy ott nagy a hiba? (erről később)
- most az a fontos, hogy lássuk, hogy a training error és a test error két külön dolog

Underfitting és overfitting

- underfitting: a modell nem elég árnyalt ahhoz, hogy jól előrejelezzen
  - ekkor a training error és a test error is nagy
  - segíthet, ha bonyolultabb modellt építünk, ehhez esetleg kellhet több tanító adat
- overfitting: túlságosan jól passzol a modell a training set-re
  - csökkentsük a modell bonyolultságát valahogyan (erről mindjárt)

Underfitting and Overfitting

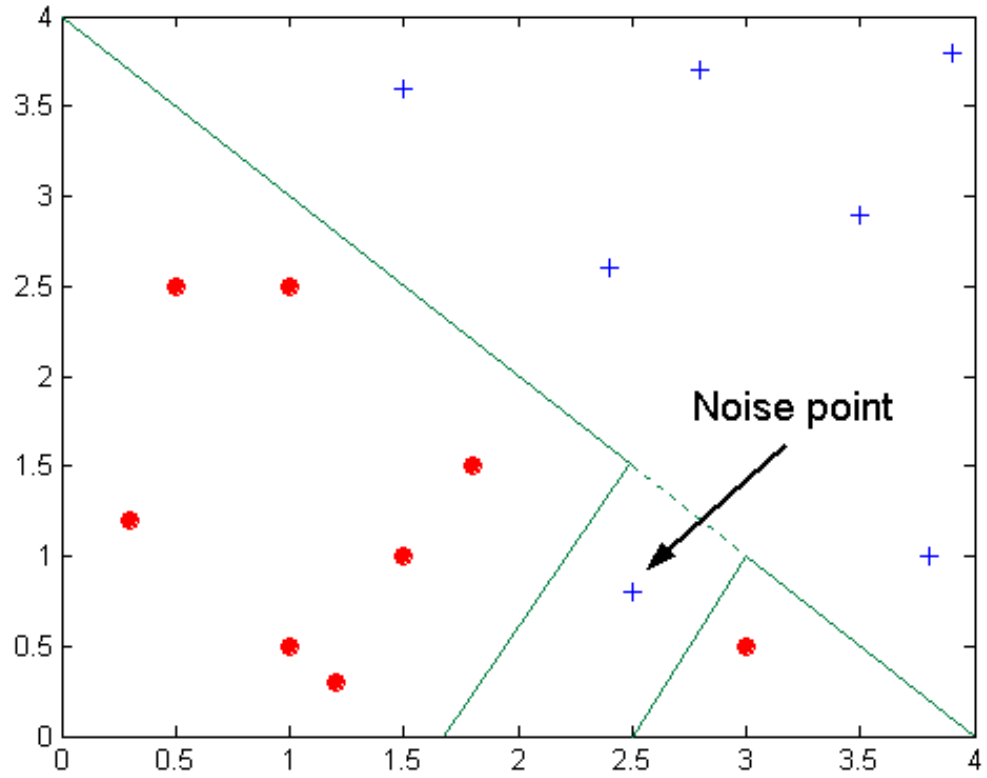


Underfitting: when model is too simple, both training and test errors are large

Overfitting oka

- közvetetten: túl erős, bonyolult modell, ami teljesen a training set-re szabható
- közvetlenül:
  - az adathalmaz nem reprezentatív (speciális esetek jelentős számban)
  - zaj
  - ha túl sok modell közül választhatunk: a legjobb nagyon jó lesz a training set-en, de semmi garancia nincs rá, hogy máshol is

Overfitting due to Noise



Decision boundary is distorted by noise point

Occam's razor

- Mindenképpen jó lenne elkerülni a túl bonyolult modellt
- elv: Occam's razor (Occam borotvája): ha van két modell, amik kábé ugyanazt tudják (hol?, hogyan?) akkor az egyszerűbbet, kisebbet válasszuk
- a modell építése során vegyük ezt az elvet figyelembe, építsük a modellt úgy, hogy az büntesse a túl bonyolultat

Kérdés

- Hogyan vegyük figyelembe a modell teljesítményét a modell építése során?
  - Nehézség: eddig a modell értékelésére a teszt halmazt használtuk, de azt nem nézhetjük meg az építés alatt.
- Vezessünk be egy újfajta hibamérést, ami figyelembe veszi a modell bonyolultságát is a training error-on kívül: generalization error
  - resubstitution error (optimista becslés)
  - pesszimista becslés (modellfüggő)
  - validation set

Resubstitution error

- feltételezzük, hogy a training set jól reprezentál
- optimistán azt gondoljuk, hogy az új adathalmazon is ugyanolyan jól fog osztályozni, mint a training set-en
- ekkor a modell hibája az, amit ennek addig hívtunk: misclassification error, azaz hány rekord lesz rosszul címkézve a training set rekordjai közül
- persze ha elég nagy fát (bonyolult modellt) építünk és kevés az adat, akkor ez simán lehet 0

Pesszimista verzió a generalization error-ra

- mivel a training set-re van szabva a modell, a valóságban nem lesz ilyen jó, nagyobb lesz a hiba
- az, hogy mennyire lesz rosszabb, az függ a modell bonyolultságától
- a pesszimista hiba két tagból áll: a training errorból és egy másik tagból, ami a modell bonyolultságát bünteti
- döntési fáknál pl. levelenként egy plusz konstans taggal megnöveljük a hibásan osztályozott rekordok számát
- pl. levelenként 1: a training errorhoz hozzá kell adni $\frac{\ell \cdot 1}{n}$-t, ahol ℓ a levelek száma, n pedig az összes rekord száma

Validation set

- a rendelkezésre álló adatokat nem két részre osztjuk (training és test), hanem háromra: training, validation és test
- ha két különböző modell (pl. két fa) között kell dönteni, akkor a validation set-en megnézzük az előrejelzésüket és a jobbat választjuk
- azért kell erre külön halmaz (nem a test set), mert a test set-et a végső modell tesztelésére tartogatjuk

Hogyan veszem figyelembe a modell becsült generalized error-ját a modell építése során?

- pre-pruning: a fa építése közben nézzük, hogy a generalized error nő-e és ha igen, akkor nem vágunk
- post-pruning (teljes fát építünk, az általános algoval, amíg lehet, addig vágunk), aztán alulról felfele haladva visszavágjuk, ha a visszavágott fa hibája kisebb (pesszimista hiba vagy validation error)

Example of Post-Pruning

Class = Yes	20
Class = No	10
Error = 10/30	

Training Error (Before splitting) = 10/30

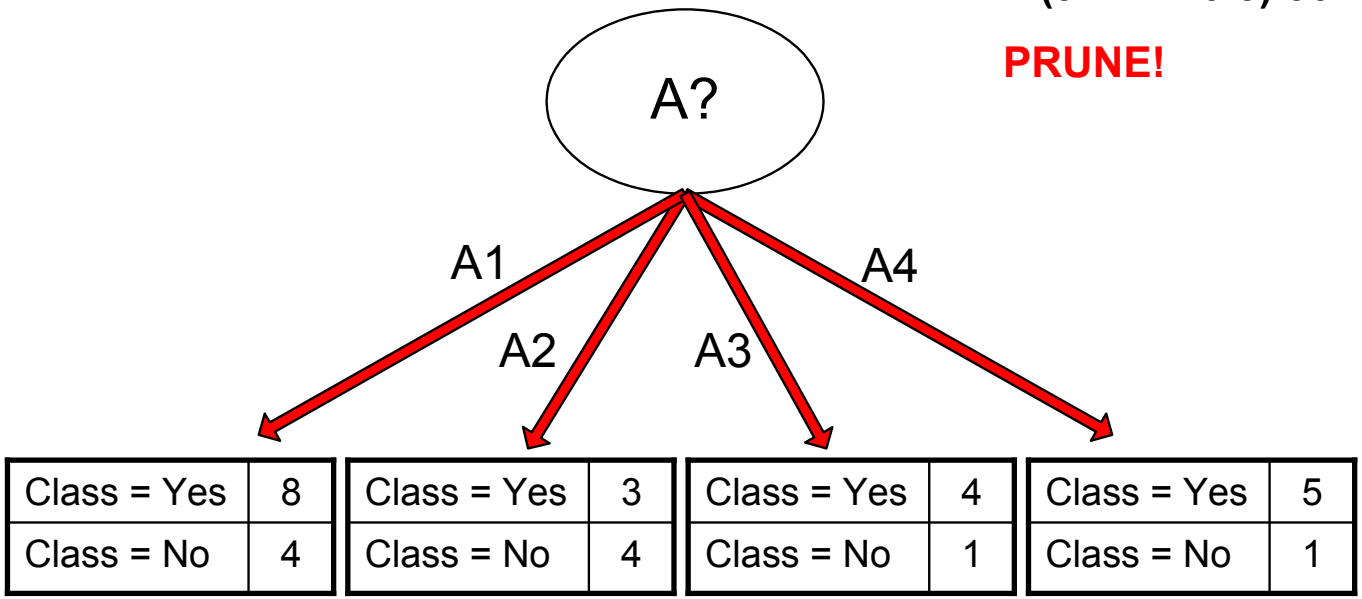
Pessimistic error = (10 + 0.5)/30 = 10.5/30

Training Error (After splitting) = 9/30

Pessimistic error (After splitting)

= (9 + 4 × 0.5)/30 = 11/30

PRUNE!



Examples of Post-pruning

- Optimistic error?

  Don't prune for both cases

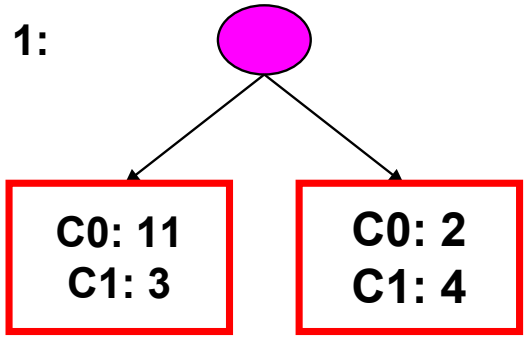
- Pessimistic error?

  Don't prune case 1, prune case 2

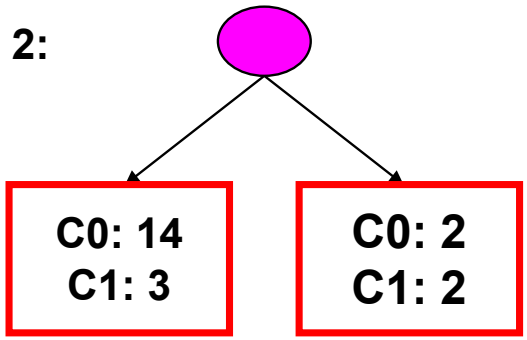
- Reduced error pruning?

  Depends on validation set

Case 1:



Case 2:



Misclassification error, változatok

- eddig accuracy és misclassification error volt: hibás előrejelzések száma az összes közül
- ez nem mindig jó, pl. ha nagyon kevés rekord van az egyik kategóriában és az az algo, hogy mindenkit a gyakori címkével címkézünk
- ezért költség-mátrixot is használhatunk, ha a hibás pozitív vagy hibás negatív osztályozást akarjuk nagyon büntetni
- vagy használhatunk accuracy helyett mást (precision, recall, F-measure)

Cost Matrix

	PREDICTED CLASS		
ACTUAL CLASS	C(i|j)	**Class=Yes**	**Class=No**
	Class=Yes	C(Yes|Yes)	C(No|Yes)
	Class=No	C(Yes|No)	C(No|No)

C(i|j): Cost of misclassifying class j example as class i

Computing Cost of Classification

Cost Matrix	PREDICTED CLASS		
	C(i|j)	+	-
ACTUAL CLASS	+	-1	100
	-	1	0

Model M_1	PREDICTED CLASS		
		+	-
ACTUAL CLASS	+	150	40
	-	60	250

Model M_2	PREDICTED CLASS		
		+	-
ACTUAL CLASS	+	250	45
	-	5	200

Accuracy = 80%

Cost = 3910

Accuracy = 90%

Cost = 4255

Másik lehetőség az osztályozó jóságának mérésére

- true positive (tp): hány olyan rekord van, ami pozitív címkét kap és a valóságban is pozitív
- true negative (tn): hány olyan rekord van, ami negatív címkét kap és a valóságban is negatív
- false positive (fp): hány olyan rekord van, ami pozitív címkét kap, de a valóságban negatív
- false negative (fn): hány olyan rekord van, ami negatív címkét kap, de a valóságban pozitív

Ez a confusion matrix

Másik lehetőség az osztályozó jóságának mérésére

- accuracy ezzel a jelöléssel $\frac{tp + tn}{tp + tn + fp + fn}$
- precision (= p): hány eset valóban pozitív a pozitívnak mondottak közül, azaz $\frac{tp}{tp + fp}$
- recall (=r): hány pozitív esetet találunk meg tényleg: $\frac{tp}{tp + fn}$
- F-measure $= \frac{2rp}{r + p} = \frac{2tp}{2tp + fp + fn}$

R-ben mi van?

- sok minden :)
- több package döntési fa készítésre: tree, rpart, party
- alap-paranccsal létrehozok egy fa típusú objektumot: megadom, hogy milyen training set-en, milyen változókat vegyen figyelembe
- a létrehozott fát használhatom előrejelzésre, ábrázolhatom
- lehet egyszerűsíteni (prune)

tree package

```
> library(tree)

> ir.tr <- tree(Species ~., iris)
> summary(ir.tr)
Classification tree:
tree(formula = Species ~ ., data = iris)
Variables actually used in tree construction:
[1] "Petal.Length" "Petal.Width" "Sepal.Length"
Number of terminal nodes:  6
Misclassification error rate:  0.02667 = 4 / 150

> plot(ir.tr)
> text(ir.tree)
```