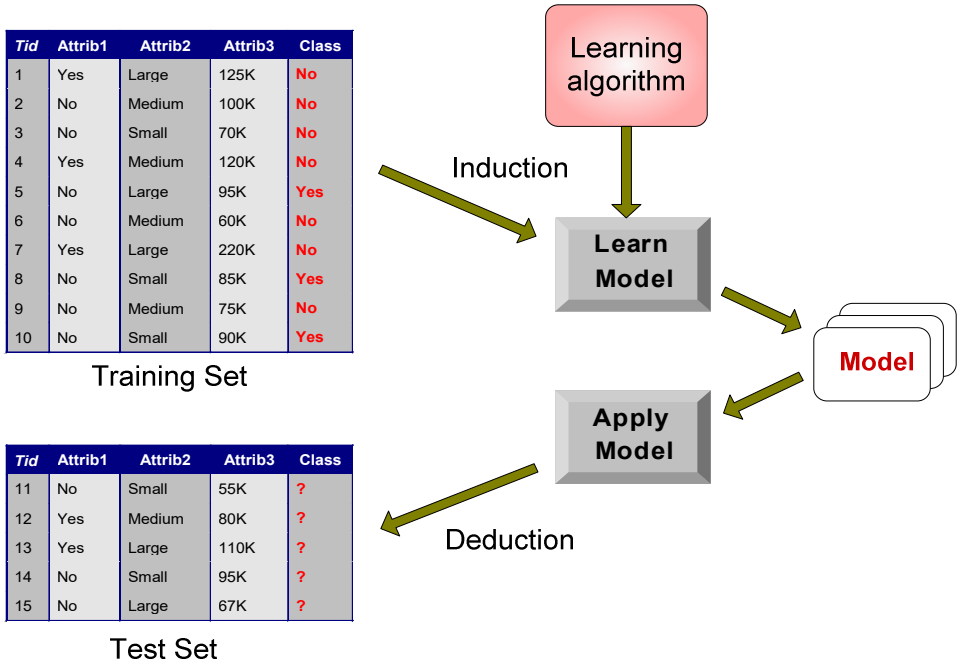
OVERFITTING
MODEL SELECTION
EVALUATION

--

Classification Errors

- **Training errors**: Errors committed on the training set
- **Test errors**: Errors committed on the test set
- **Generalization errors**: Expected error of a model over random selection of records from same distribution



Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

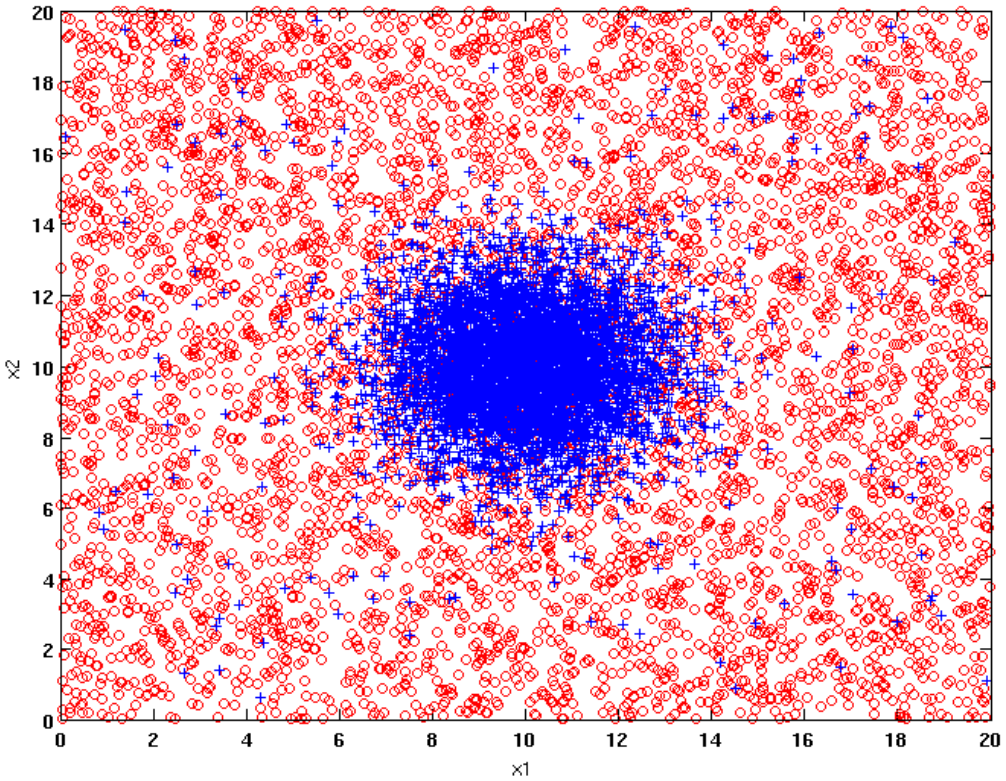
Training Set

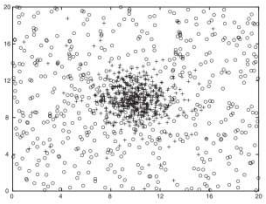
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

-
ارزیابی Classification:
وقتی که تسک Classification داریم سه خطا خواهیم داشت:
یک خطا داریم روی دیتای اموزشی یا Training errors
یک خطا داریم روی دیتای تست یا Test errors
خطای تعمیم یافتگی یا Generalization errors: این متوسط خطای مدل ما است روی دیتایی که توزیعش با همین دیتای ما یکی است ولی اون ها رو ندیده
همه این کارایی که ما داریم توی Classification انجام میدیم به امید این است که Generalization errors بهتری داشته باشیم

Example Data Set





(b) Training set using 10% data.

+: این 10 درصد دیتا است

Two class problem:

+ : 5400 instances

- **5000 instances generated from a Gaussian centered at (10,10)**
- **400 noisy instances added**

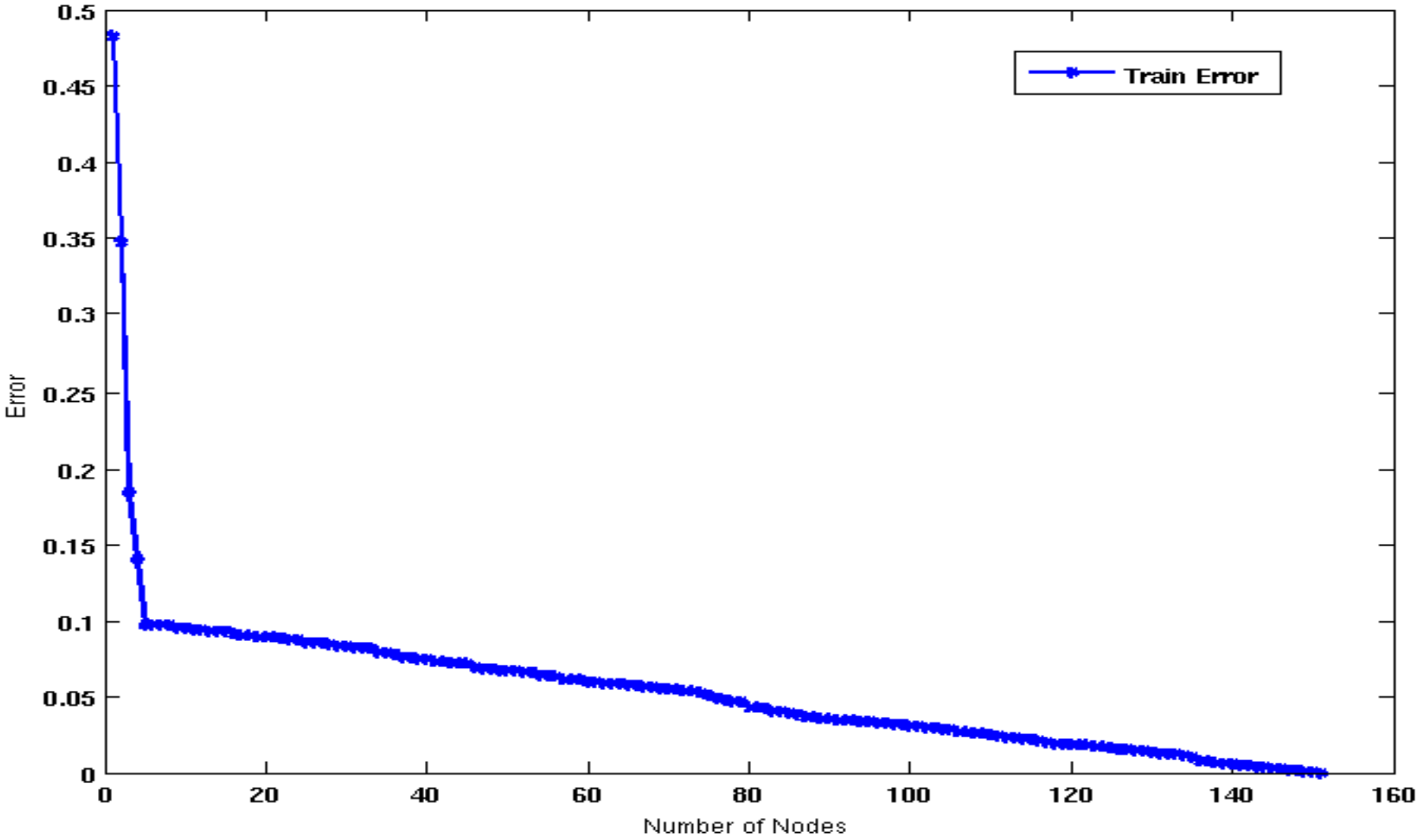
o : 5400 instances

- **Generated from a uniform distribution**

10 % of the data used for training and 90% of the data used for testing

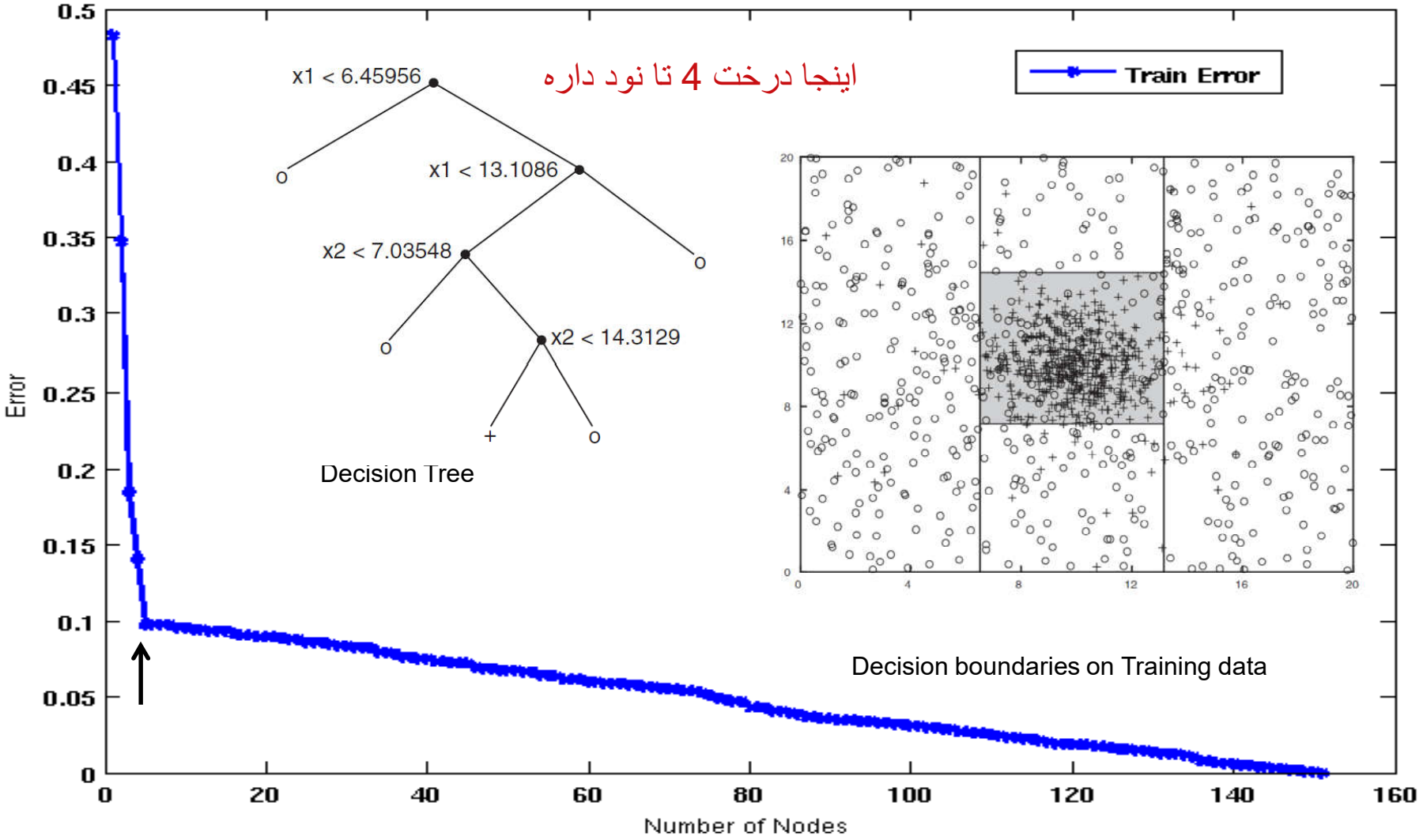
-
فرض کنید یک دیتایی داریم که دو بعد x , y داره این دیتا و دو کلاسه هم هست: 5400 تا کلاس قرمز و 5400 تا کلاس ابی
حالا 10 درصد دیتا رو به ما دادن و شده تصویر + --> ینی 10 درصد دیتا رو به ما دادن و به ما گفتن روی این بیا یک مدلی ایجاد بکن و در نهایت روی مدلی که ایجاد شده می خوایم قضاوت بکنیم
+ --> توزیع 10 درصد دیتا از لحاظ شکل مثل بالایی است ولی در واقعیت یکسری از واقعیت ها توش نیست مثلا تراکم عجیب و غریبی که توی اون مرکز داریم توی این نیست
اگر بخوایم + بدیم دست درخت تصمیم و بهش بگیم شروع بکن به یادگیری روی این دیتا + و این درخت تلاش میکنه برای اموزش
چجوری درخت رو می سازه؟ یک مستطیل اون وسط درست میکنه در واقع

Increasing number of nodes in Decision Trees



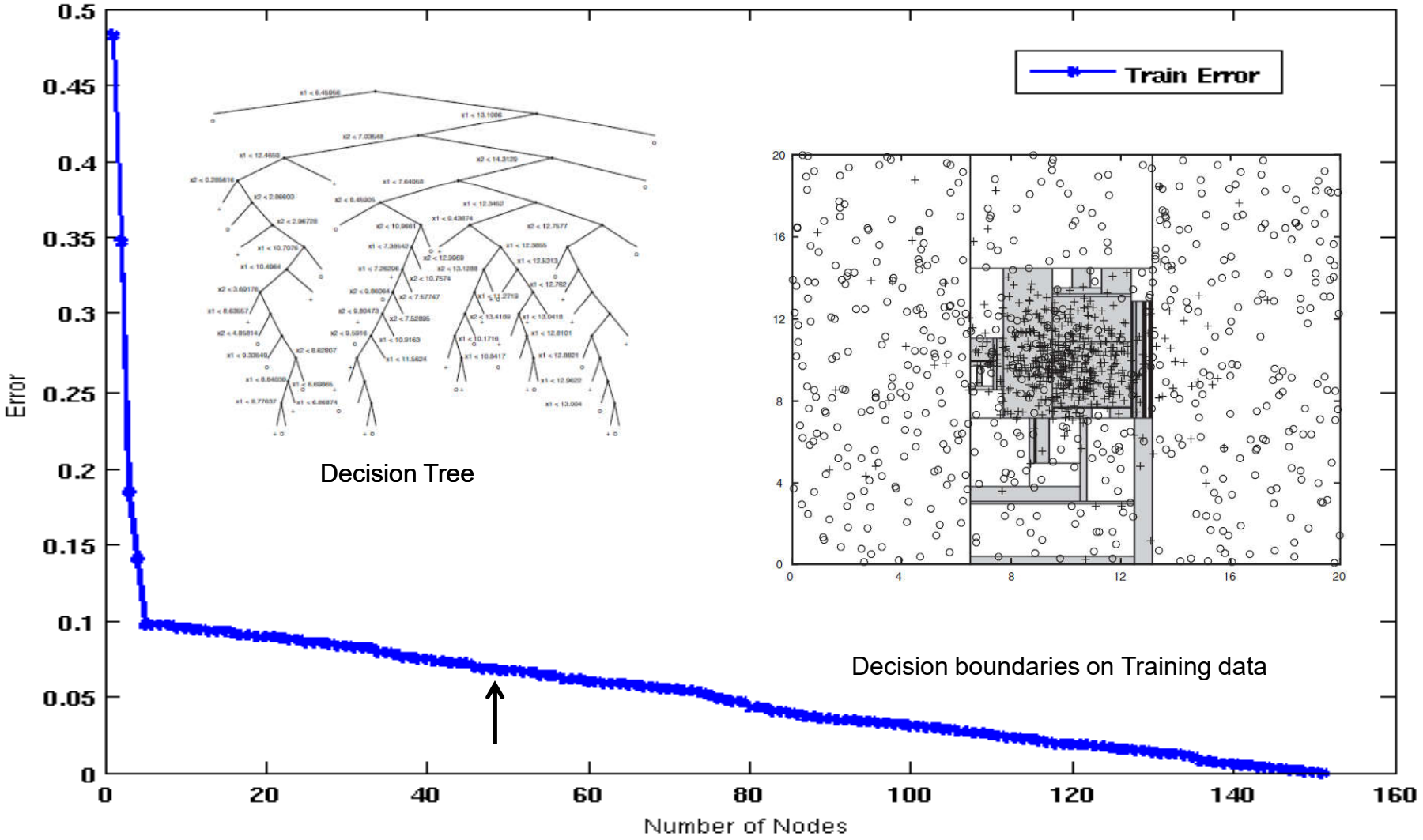
-
رسم خطا:
به این خطا میگیم خطای داده های اموزشی
محور x تعداد نودهای درخت است و محور y خطایی است که تا اون لحظه داشتیم مثلا با اضافه کردن نود اول 50 درصد خطا داشتیم روی دوتا کلاس و بعد هرچی نودها بیشتر میشه این خطا کاهش پیدا میکنه و از یه جایی به بعد اون خطا اون افت عجیب و غریب رو دیگه نداره و یک رفتار کاهشی خیلی کند پیدا میکنه

Decision Tree with 4 nodes



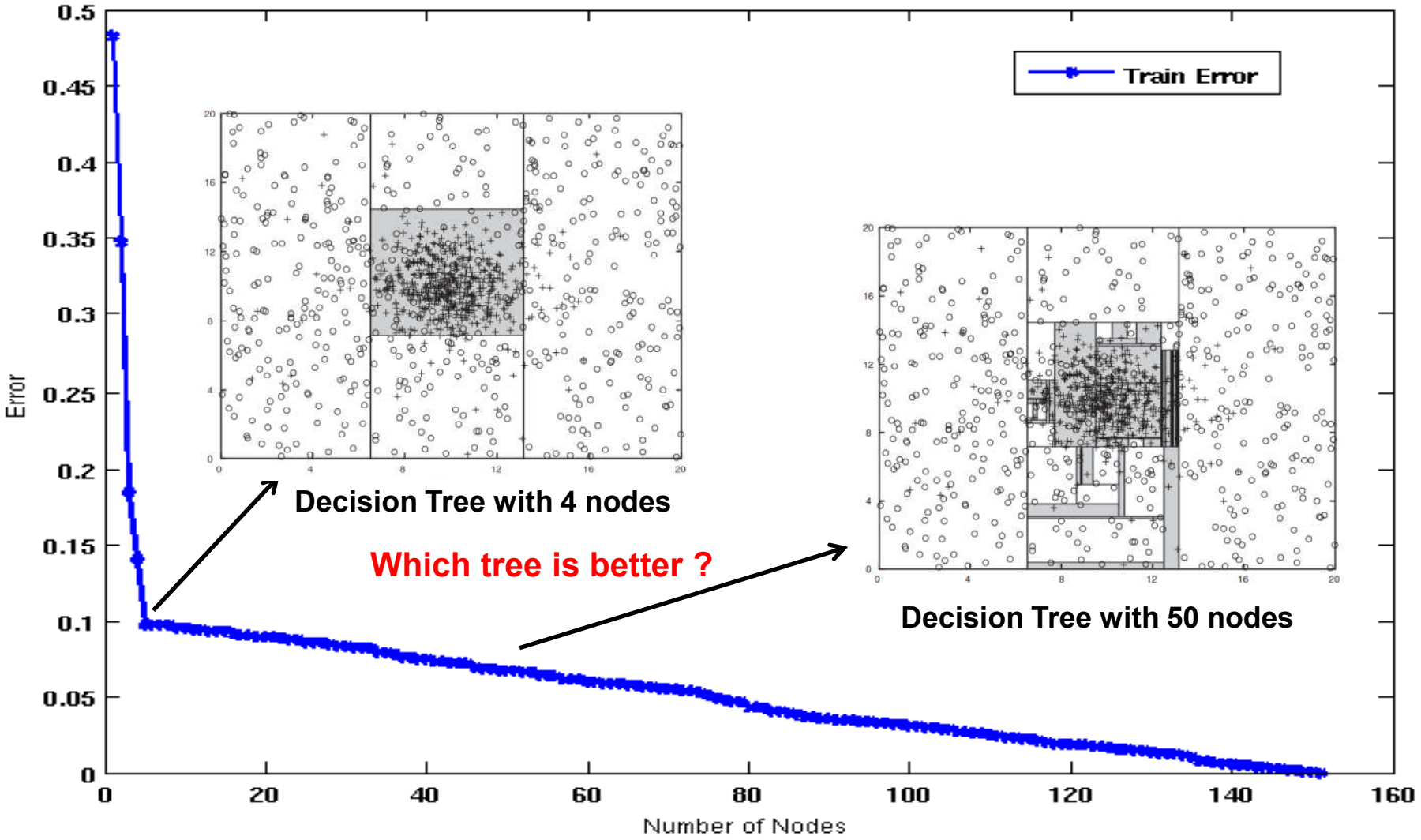
--

Decision Tree with 50 nodes



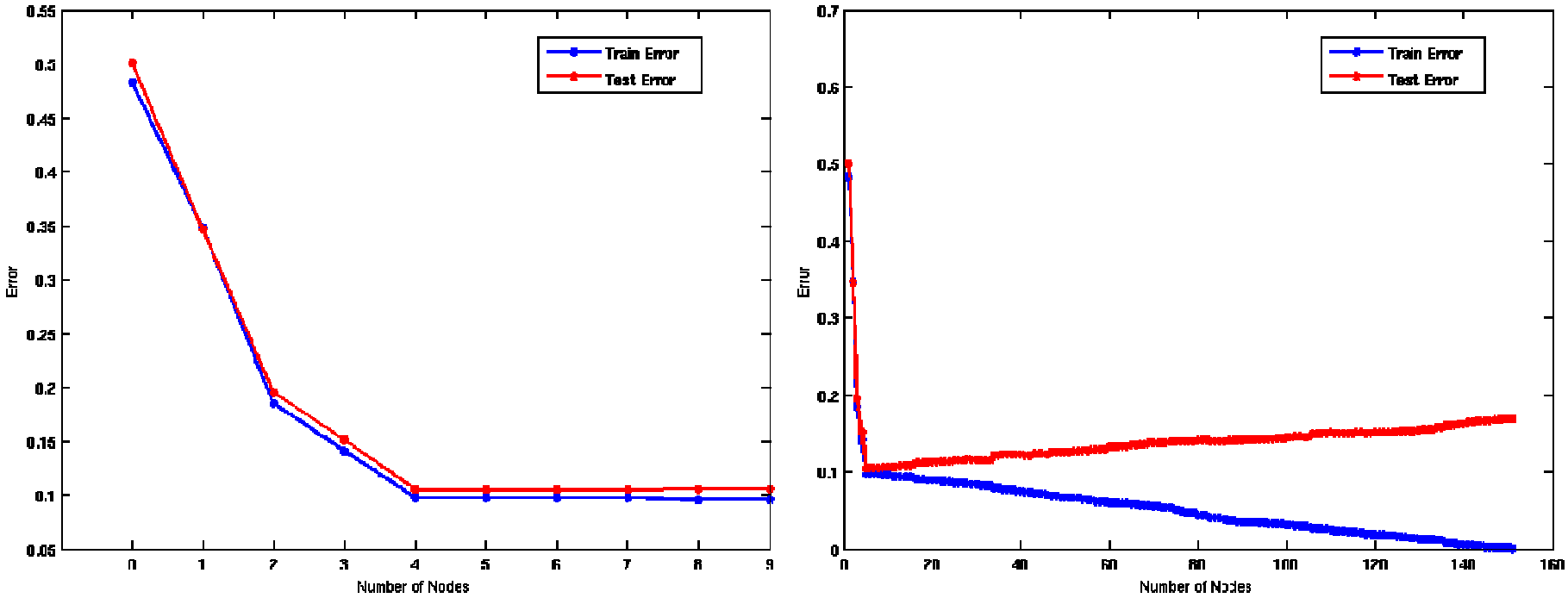
--

Which tree is better?



درخت اول بهتره یا درخت دوم؟
چیشد؟؟

Model Underfitting and Overfitting



•As the model becomes more and more complex, test errors can start increasing even though training error may be decreasing

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

Overfitting: when model is too complex, training error is small but test error is large

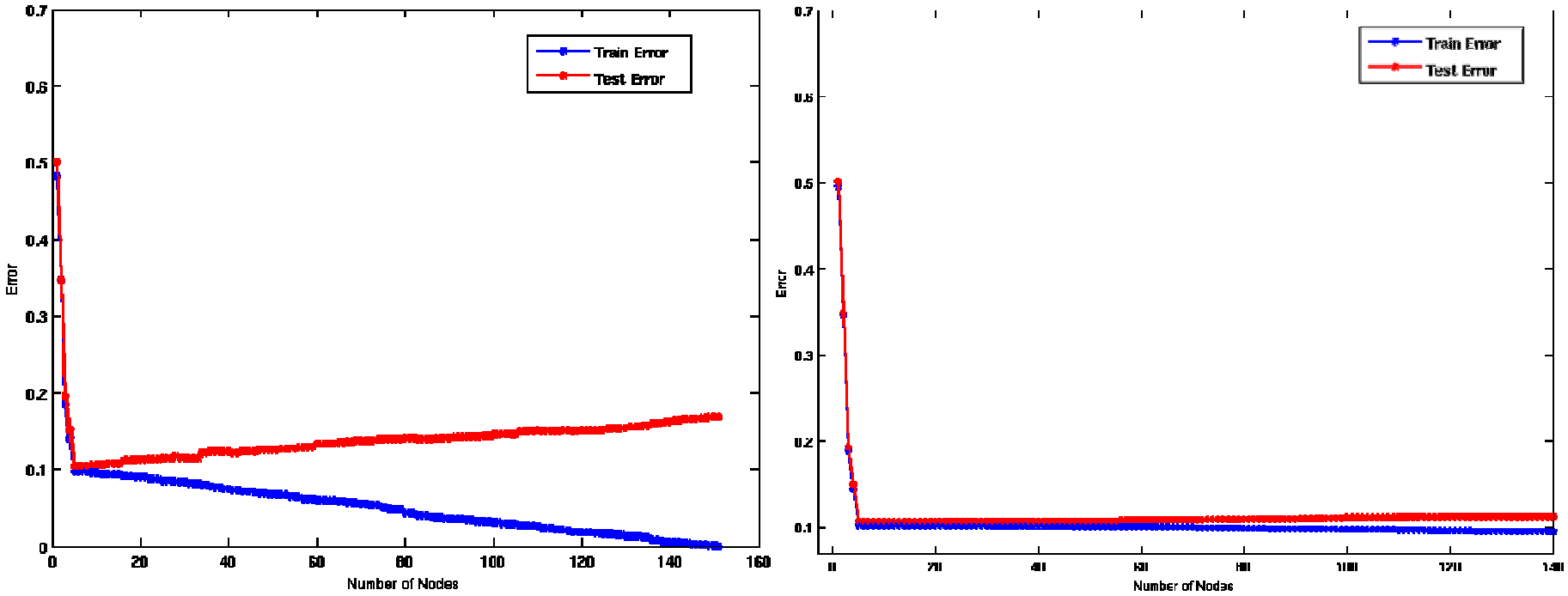
-

Overfitting: ینی مدل زیاد سعی کرده داده ها رو یاد بگیره ینی بیش از اندازه fit شده روی داده ها

Underfitting: وقتایی که دیتا هنوز خیلی خوب fit نشده روی داده ها --> اونجایی هست که خطا خیلی بالا است

حالا Underfitting و Overfitting چطوری میشه تشخیص داد؟ با کمک Training errors و Test errors

Model Overfitting – Impact of Training Data Size

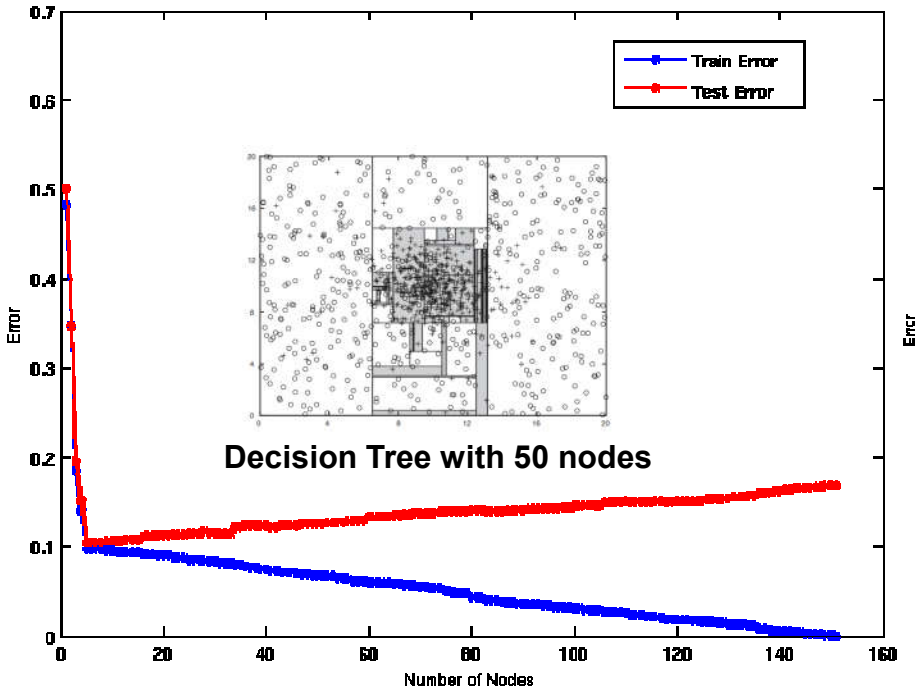


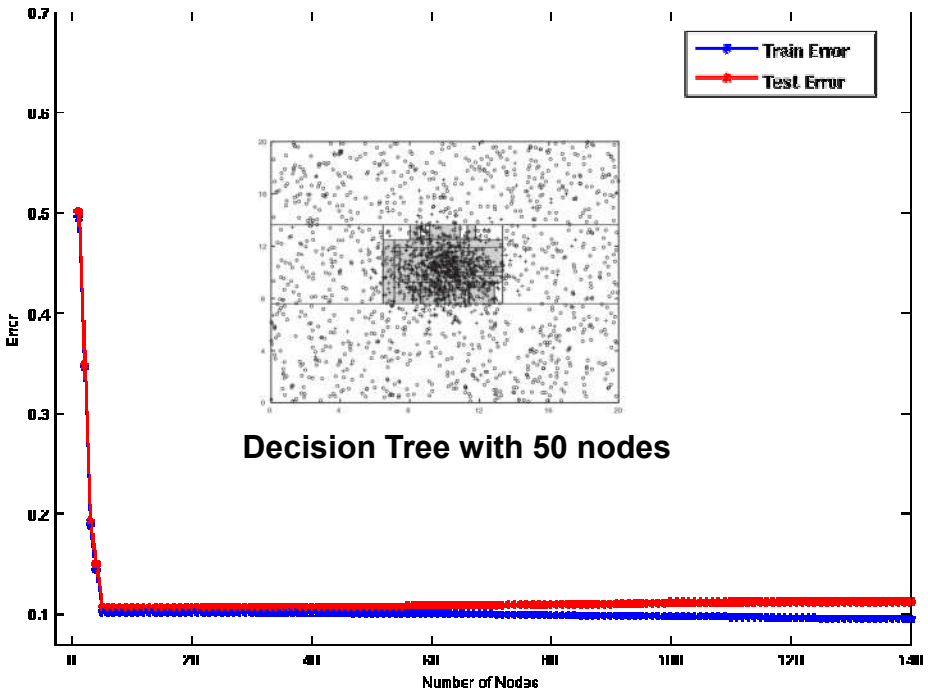
Using twice the number of data instances

- Increasing the size of training data reduces the difference between training and testing errors at a given size of model

-
تا 9 تا نود اول خطای اموزشی و خطای تست هر دو داره کم میشه ولی وقتی این اموزش داره ادامه پیدا میکنه از یه جایی به بعد خطای اموزش کمتر میشه ولی خطای تست داره بیشتر میشه --> اینجا میگیم Overfitting داره رخ میده و نباید بذاریم مدلمون اینقدر train بشه

Model Overfitting – Impact of Training Data Size





Using twice the number of data instances

- Increasing the size of training data reduces the difference between training and testing errors at a given size of model

ما دنبال این هستیم که نه Overfitting بشه و نه Underfitting بشه
بهترین مدل کجا میشه؟؟

یک اتفاق دیگه ای که Overfitting برامون تولید میکنه، داده های اموزشی هستن --> اگر دیتای اموزشی خیلی کم باشه Overfitting می تونه برامون اتفاق بیوفته --> پس اگر داده های اموزشی رو خیلی زیاد بکنیم احتمال Overfitting کاهش پیدا میکنه

Reasons for Model Overfitting

- Not enough training data

- High model complexity
  - Multiple Comparison Procedure

دلایل بیش از حد برازش مدل
+ داده های آموزشی کافی نیست
+ پیچیدگی مدل بالا
- روش مقایسه چندگانه

داده های بیشتر Overfitting کمتر

Effect of Multiple Comparison Procedure

- Consider the task of predicting whether stock market will rise/fall in the next 10 trading days

- Random guessing:

$$P(correct) = 0.5$$

- Make 10 random guesses in a row:

$$P(\#\,correct \geq 8) = \frac{\binom{10}{8} + \binom{10}{9} + \binom{10}{10}}{2^{10}} = 0.0547$$

Day 1	Up
Day 2	Down
Day 3	Down
Day 4	Up
Day 5	Down
Day 6	Down
Day 7	Up
Day 8	Up
Day 9	Up
Day 10	Down

Effect of Multiple Comparison Procedure

- Approach:
  - Get 50 analysts
  - Each analyst makes 10 random guesses
  - Choose the analyst that makes the most number of correct predictions

- Probability that at least one analyst makes at least 8 correct predictions

$$P(\#\,correct \geq 8) = 1 - (1 - 0.0547)^{50} = 0.9399$$

Effect of Multiple Comparison Procedure

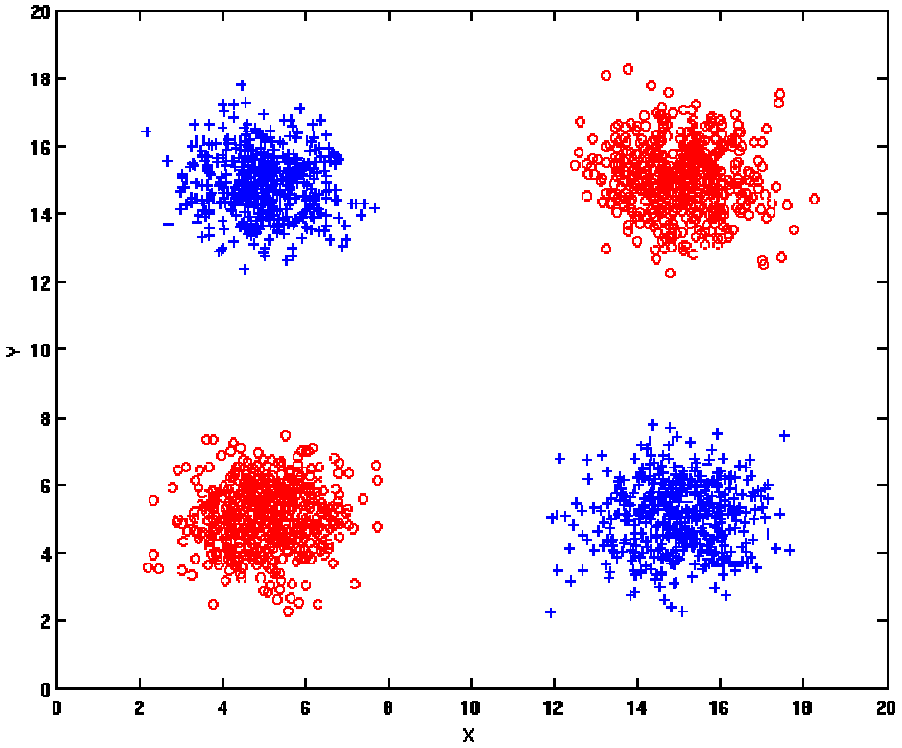
- Many algorithms employ the following greedy strategy:
  - Initial model: M
  - Alternative model: M' = M $\cup$ γ,
    where γ is a component to be added to the model
    (e.g., a test condition of a decision tree)
  - Keep M' if improvement, Δ(M,M') > α

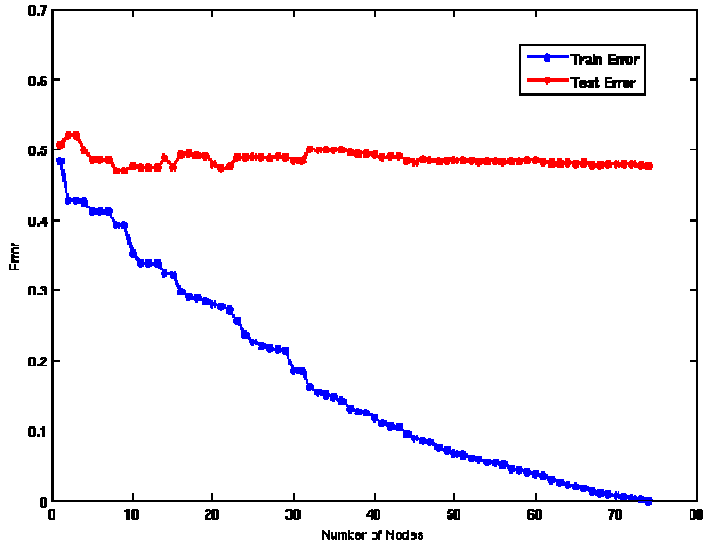
- Often times, γ is chosen from a set of alternative components, $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_k\}$

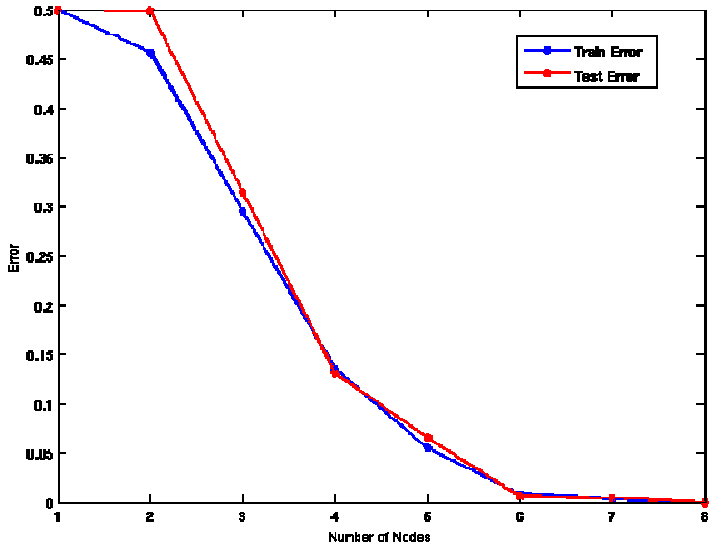
- If many alternatives are available, one may inadvertently add irrelevant components to the model, resulting in model overfitting

---

Effect of Multiple Comparison - Example







Use additional 100 noisy variables generated from a uniform distribution along with X and Y as attributes.

Use 30% of the data for training and 70% of the data for testing

Using only X and Y as attributes

از 100 متغیر نویز اضافی تولید شده از یک توزیع یکنواخت به همراه X و Y به عنوان ویژگی استفاده کنید. از 30 درصد داده ها برای آموزش و 70 درصد از داده ها برای آزمایش استفاده کنید

Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary

- Training error does not provide a good estimate of how well the tree will perform on previously unseen records

- Need ways for estimating generalization errors(How?)

Model Selection

- Purpose
  - ensure that model is not overly complex (to avoid overfitting)
  - Performed during model building

- Need to estimate generalization error
  - Using Validation Set
  - Incorporating Model Complexity

این مسئله:
یه جایی پیش میاد دوتا درخت داریم --> کدوم یکی از این درخت ها رو انتخاب بکنیم؟
مدلی انتخاب بشه که ساده تر است

Model Selection:

Using Validation Set

- Divide training data into two parts:
  - Training set:
    - use for model building
  - Validation set:
    - use for estimating generalization error
    - Note: validation set is not the same as test set

- Drawback:
  - Less data available for training

Validation Set: یکی بخشی از این دیتای اموزشی رو جدا میکنیم و دیگه نمیاریمش توی اموزش

Model Selection:

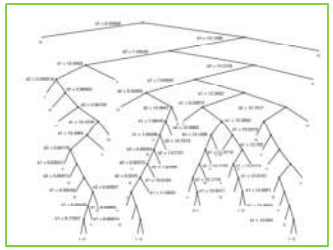
Incorporating Model Complexity

- Rationale: Occam's Razor
  - Given two models of similar generalization errors,

    one should prefer the simpler model over the more complex model

  - A complex model has a greater chance of being fitted accidentally

  - Therefore, one should include model complexity when evaluating a model



Gen. Error(Model) = Train. Error(Model, Train. Data) +

α x Complexity(Model)

-
وقتی یک مدلی ایجاد کردیم روی دیتای اموزش یک خطایی روی دیتای اموزشی به دست میاد به اسم Train. Error و کنارش بیاد پیچیدگی مدل رو در نظر بگیر با یک ضریبی --> خطا افزایش پیدا میکنه
خطای عمومی میشه کلا --> بحث over هم توش هست

Estimating the Complexity of Decision Trees

- **Pessimistic Error Estimate** of decision tree T with k leaf nodes:

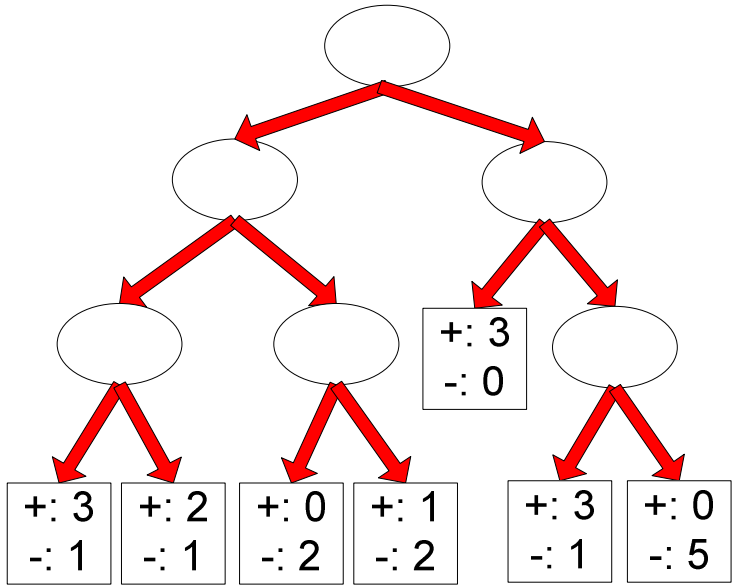
$$err_{gen}(T) = err(T) + \Omega \times \frac{k}{N_{train}}$$

  - err(T): error rate on all training records
  - Ω: trade-off hyper-parameter (similar to α)
    - Relative cost of adding a leaf node
  - k: number of leaf nodes
  - N_{train}: total number of training records

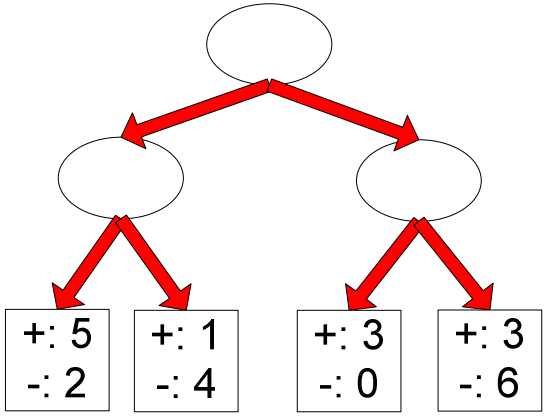
k میشه تعداد برگ های درخت
n تعداد داده های اموزشی

مدل پیچیده تعداد زیادی برگ داره نسبت به داده اموزشی پس err gen رو افزایش میده

Estimating the Complexity of Decision Trees: Example



Decision Tree, T_L



Decision Tree, T_R

$e(T_L) = 4/24$

$e(T_R) = 6/24$

$\Omega = 1$

$e_{gen}(T_L) = 4/24 + 1*7/24 = 11/24 = 0.458$

$e_{gen}(T_R) = 6/24 + 1*4/24 = 10/24 = 0.417$

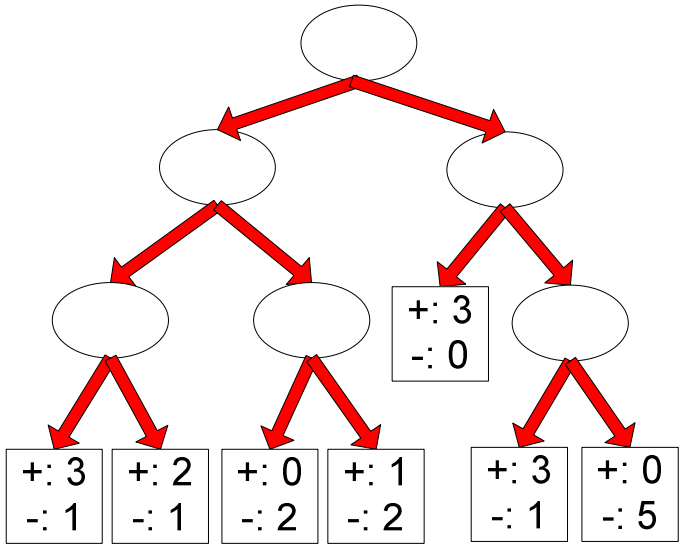
کدوم درخت رو انتخاب بکنیم؟

درخت سمت راستی بهتر است

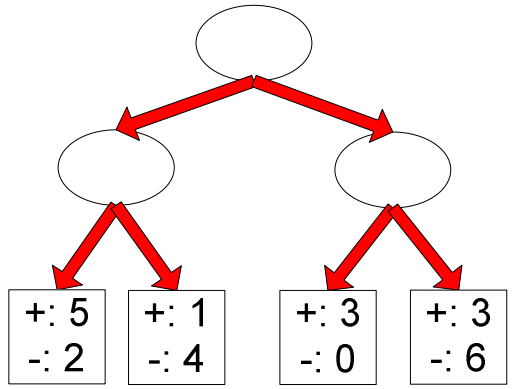
err gen کمک می کرد خطای عمومی یک clasifier رو بدون دیتای تست حساب بکنیم

Estimating the Complexity of Decision Trees

- Resubstitution Estimate:
  - Using training error as an optimistic estimate of generalization error
  - Referred to as optimistic error estimate



Decision Tree, T_L



Decision Tree, T_R

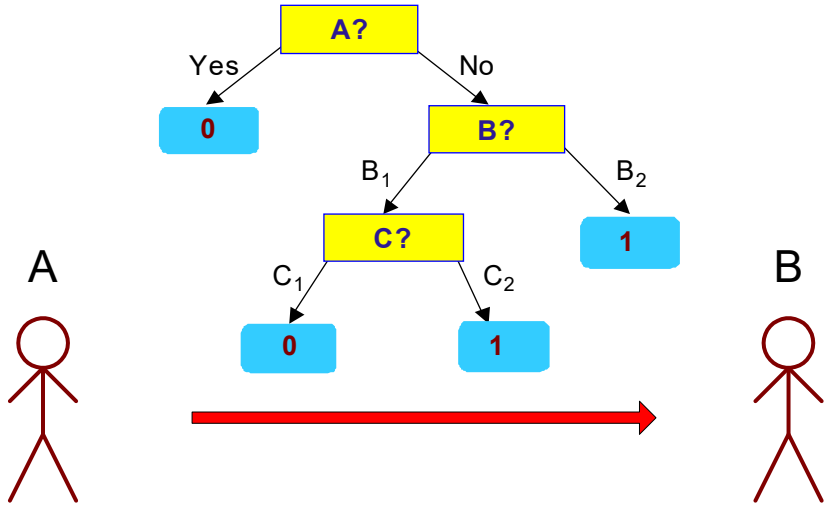
$e(T_L) = 4/24$

$e(T_R) = 6/24$

اسم های دیگه ای که بهش میگن

Minimum Description Length (MDL)

X	y
X_1	1
X_2	0
X_3	0
X_4	1
…	…
X_n	1

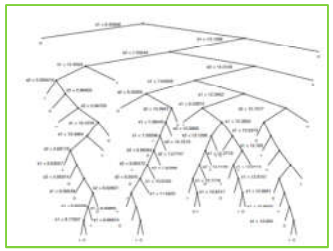


X	y
X_1	?
X_2	?
X_3	?
X_4	?
…	…
X_n	?

- Cost(Model,Data) = Cost(Data|Model) + α x Cost(Model)
  - Cost is the number of bits needed for encoding.
  - Search for the least costly model.
- Cost(Data|Model) encodes the misclassification errors.
- Cost(Model) uses node encoding (number of children) plus splitting condition encoding.

Model Selection for Decision Trees

- Pre-Pruning (Early Stopping Rule)
  - Stop the algorithm before it becomes a fully-grown tree
  - Typical stopping conditions for a node:
    - Stop if all instances belong to the same class
    - Stop if all the attribute values are the same
  - More restrictive conditions:
    - Stop if number of instances is less than some user-specified threshold
    - Stop if class distribution of instances are independent of the available features (e.g., using χ^2 test)
    - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).
    - Stop if estimated generalization error falls below certain threshold



-
برای اینکه درخت بزرگ نشه می تونیم:
هرس کردن کمک میکنه که پیچیدگی درخت کمتر بشه و از این چالش هایی که توی ove.. رخ می ده برامون جلوگیری بکنه
پیش هرس: چی باعث میشه که درخت دیگ توسعه پیدا نکنه --> حساس تر بشیم پس: 3 تا چیز برای توقف داشتیم و حساس تر روی اینا میشه:
4 ایده:
1: یک تری شلد یا استانه بذاریم و از یه جایی به بعد دیگه نذاریم شاخه توسعه پیدا بکنه مثلا بگیم ما انتظار داریم که توی هر شاخه به دقت 70 برسیم و دیگه جلوتر نریم پس یک تری شلد برای ناخالصی می خوایم بذاریم
2: توی بعضی از شاخه ها که داره توسعه پیدا میکنه اگر توی یک شاخه تعداد داده ها که قرار گرفتن که خیلی کم شد دیگه نرو ویژگی اضافه کن واین شاخه رو گسترش نده ینی برای ده تا نمونه که توی یک شاخه گیر کرده لازم نیست ویژگی اضافه کنیم که اینجا یک عدد تعریف می کنیم و میگیم توی هر شاخه دیگه دست کم 8 تا باشه کمتر نباشه
3: استقلال --> اگر به جایی رسیدیم که ویژگی های که باقی مونده توی یک شاخه هیچ ربطی به کلاس ما ندارن طبق شاخص های اماری --> دیگه اضافشون نکنیم؟؟؟
4: شاید درخت رو با معیار جینی ساختیم و به یک شرایطی که رسیدیم ؟؟؟

Model Selection for Decision Trees

- Post-pruning
  - Grow decision tree to its entirety
  - Subtree replacement
    - Trim the nodes of the decision tree in a bottom-up fashion
    - If generalization error improves after trimming, replace sub-tree by a leaf node
    - Class label of leaf node is determined from majority class of instances in the sub-tree

-
پیش هرس مال قبل از ove.. است
پس هرس:
با یکسری روش هایی بریم درخت رو هرس بکنیم و این هرس کردن خیلی به جلوگیری ازove.. و کاهش پیچیدگی کمک میکنه
درخت رشد بکنه --> بعد با یک شاخصی بریم سراغ یکسری شاخه ها و روی اون شاخه ها حساس شده و اگر بدرد نمی خورن اون شاخه رو حذف بکنی
از بالا به پایین درخت توسعه پیدا می کرد ولی برای هرس کردن معکوس این عمل می کنیم
کجا رو انتخاب بکنیم برای هرس کردن:
مثلا جایی رو هرس بکنیم که err gen رو کاهش میده مثلا یک بار err gen رو روی کل درخت حساب بکنیم و یک بار هم بگیم اگر این شاخه رو حذف بکنیم چند میشه و اگر کاهش یافت شاخه رو حذف میکنیم --> 1
Class label :2

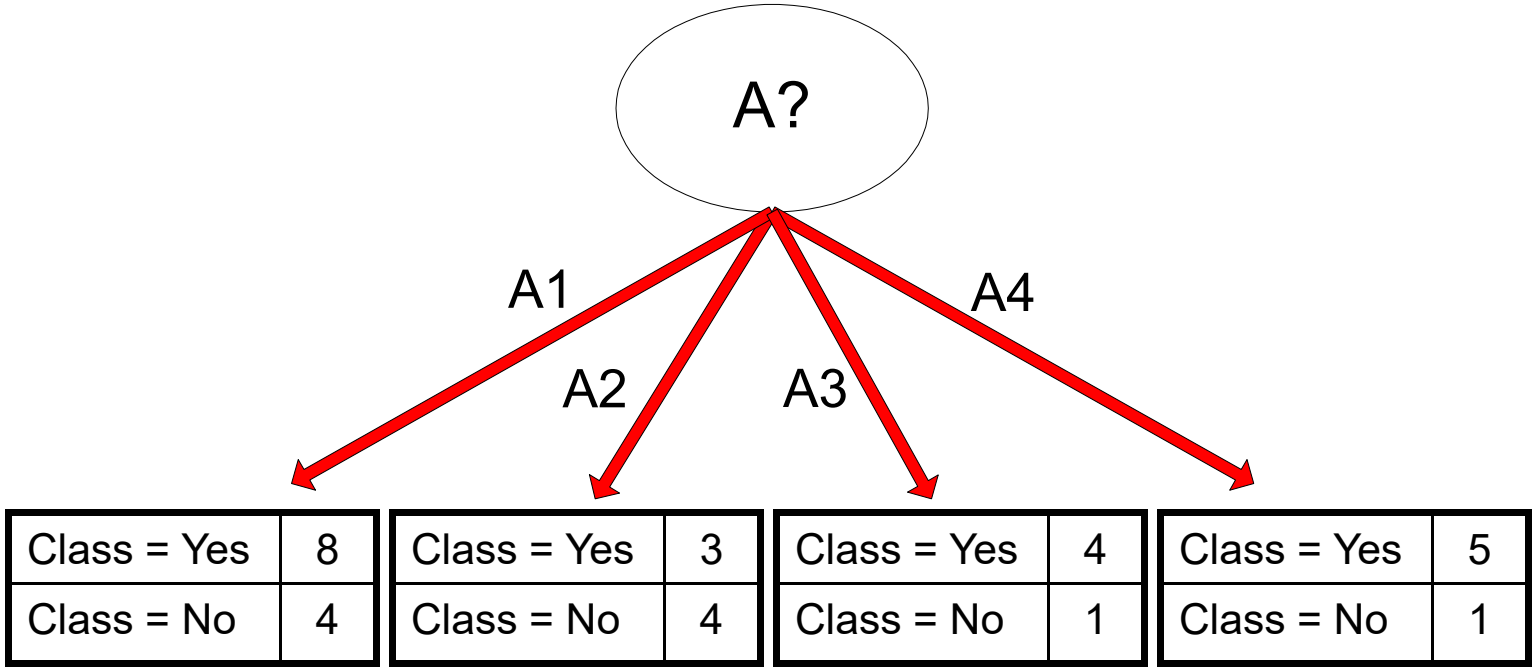
Example of Post-Pruning

Training Error (Before splitting) = 10/30

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

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

قبل A:



Class = Yes	8
Class = No	4

Class = Yes	3
Class = No	4

Class = Yes	4
Class = No	1

Class = Yes	5
Class = No	1

--

Example of Post-Pruning

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

Training Error (Before splitting) = 10/30

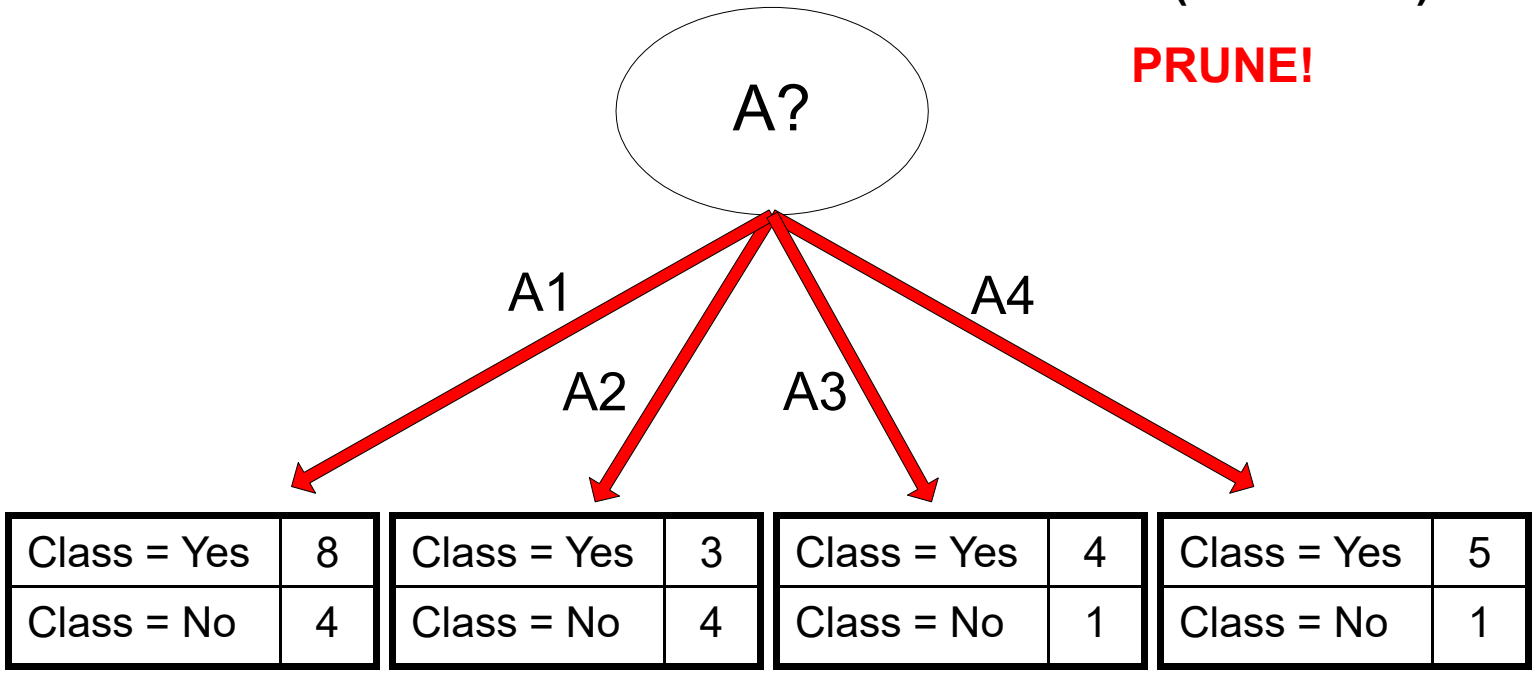
Pessimistic error = (10 + 0.5)/30 = 10.5/30

Training Error (After splitting) = 9/30

Pessimistic error (After splitting)

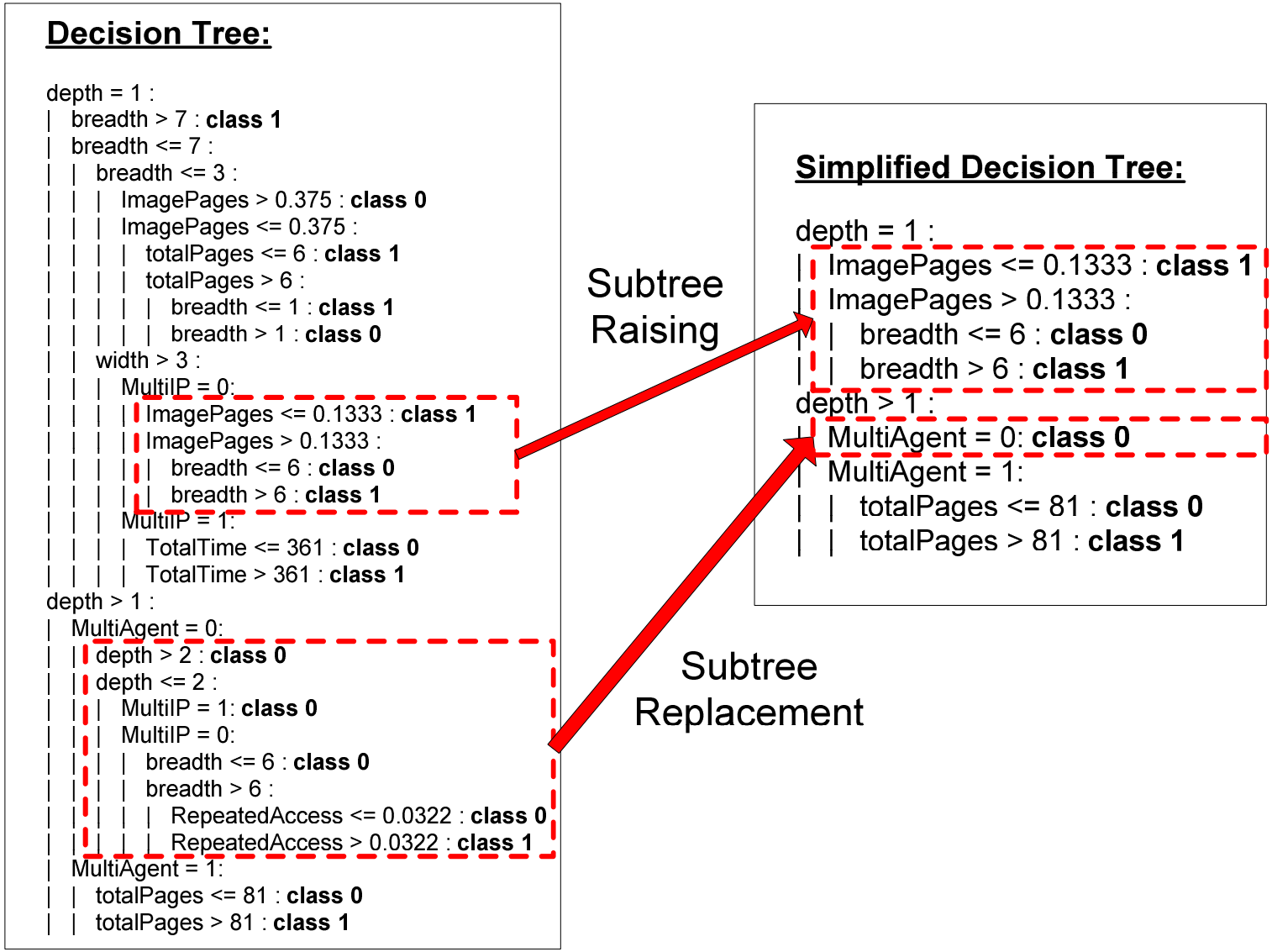
= (9 + 4 × 0.5)/30 = 11/30

PRUNE!



ساخت درخت از بالا به پایین است
هرس کردن از پایین به بالا

Examples of Post-pruning



--

Model Evaluation

Purpose:

- To estimate performance of classifier on previously unseen data (test set)

- Holdout
- Cross validation

--

Model Evaluation:Holdout

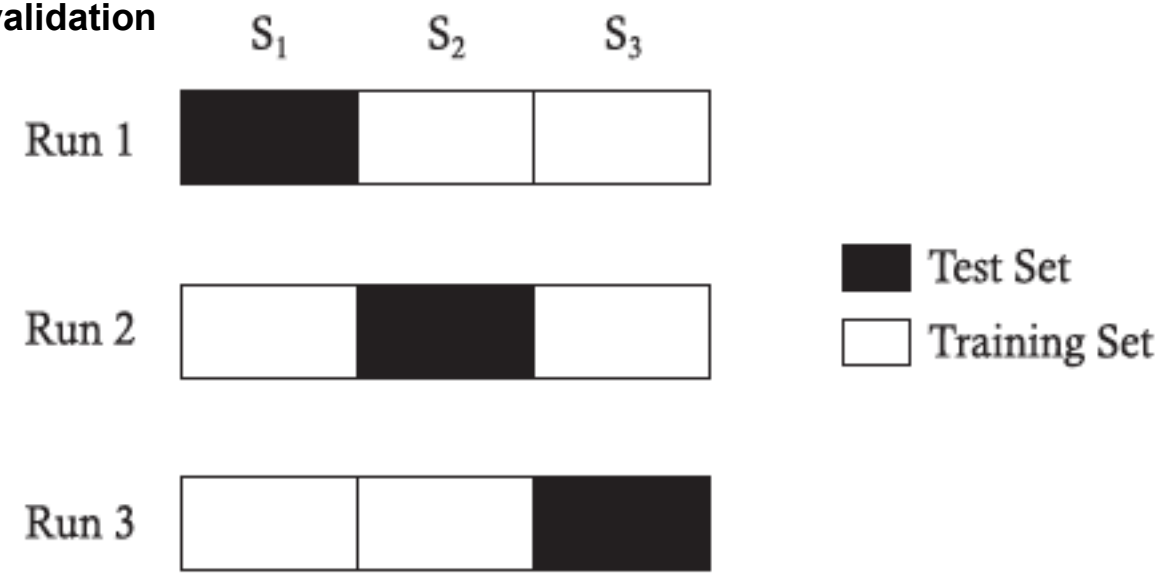
- Holdout
  - Reserve k% for training and (100-k)% for testing
  - Random subsampling: repeated holdout

Holdout:
وقتی با دیتایی برخورد کردیم بیا k درصد برای اموزش در نظر بگیر و مابقی رو برای تست
نکته: Validation Set یک بخشی از همین k درصده است

Model Evaluation:Cross-validation

- Cross validation
  - Partition data into k disjoint subsets
  - k-fold: train on k-1 partitions, test on the remaining one
  - Leave-one-out: k=n

3-fold cross-validation



-
: Cross-validation
داده ها رو به k تا دسته تقسیم بکن همون اول کار و هر دفعه بیا k-1 رو بذار برای اموزش و بقیه قست باقی مانده برای تست و حواست باشه اینا باهم overlap نداشته باشن
توی قبلی ممکن بود overlap رخ بده چون داشتیم تصادفی برمیداشتیم ولی اینجا این اتفاق نمی افته چون هر دسته ای یا تست است یا ترین
حالت حدی داره این روش که بهش Leave-one-out و موقعی این شرایط پیش میاد که k تا دسته برابر با تعداد نمونه ها باشه
نکته: وقت هایی که تعداد داده های اموزشی کمه می گن یک دیتا رو بذار تست و بقیه نمونه ها رو بذار واسه ترین مثلا اگر 100 تا داریم 99 تا ترین و یک نمونه واسه تست : Leave-one-out

Variations on Cross-validation

- Repeated cross-validation
  - Perform cross-validation a number of times
  - Gives an estimate of the variance of the generalization error
- Stratified cross-validation
  - Guarantee the same percentage of class labels in training and test
  - Important when classes are imbalanced and the sample is small
- Use nested cross-validation approach for model selection and evaluation

وقت هایی که داده ها بالانس نیستن ینی توازن مثبت و منفی با هم یکی نباشه و زمانی که k-fold می زنیم این احتمال وجود داره که همش از یک کلاس باشه از بدشانسی مثلا 80 مثبت بود و 20 منفی و کاری که می کنن این است که می گن این نسبت 80 به 20 توی هر دو جا رعایت بشه --> ینی نمونه برداری به نحوی انجام میده که هر fold که به عنوان تست شد نسبت کلاس مثبت و منفی توش رعایت بشه و یه کاری می کنیم که توی داده های اموزشی و داده های تست مطمئن باشیم که از هر دوتا کلاس به یک نسبت داشته باشیم