# Mining High-Speed Data Streams

Pedro Domingos

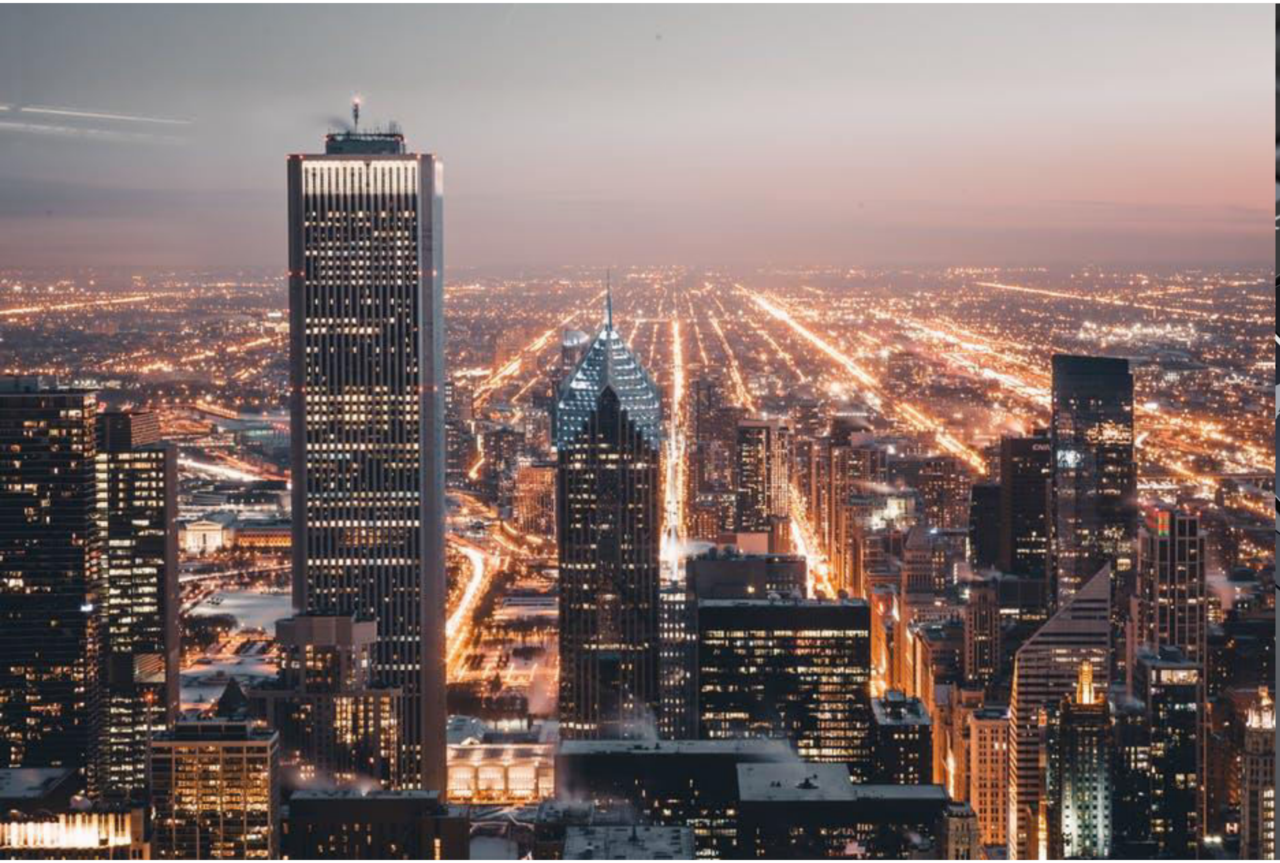
University of Washington

Geoff Hulten

University of Washington

Davide Gallitelli

Politecnico di Torino – TELECOM ParisTech
@DGallitelli95



# **Huge** and **Fast** data streaming

Limited by:

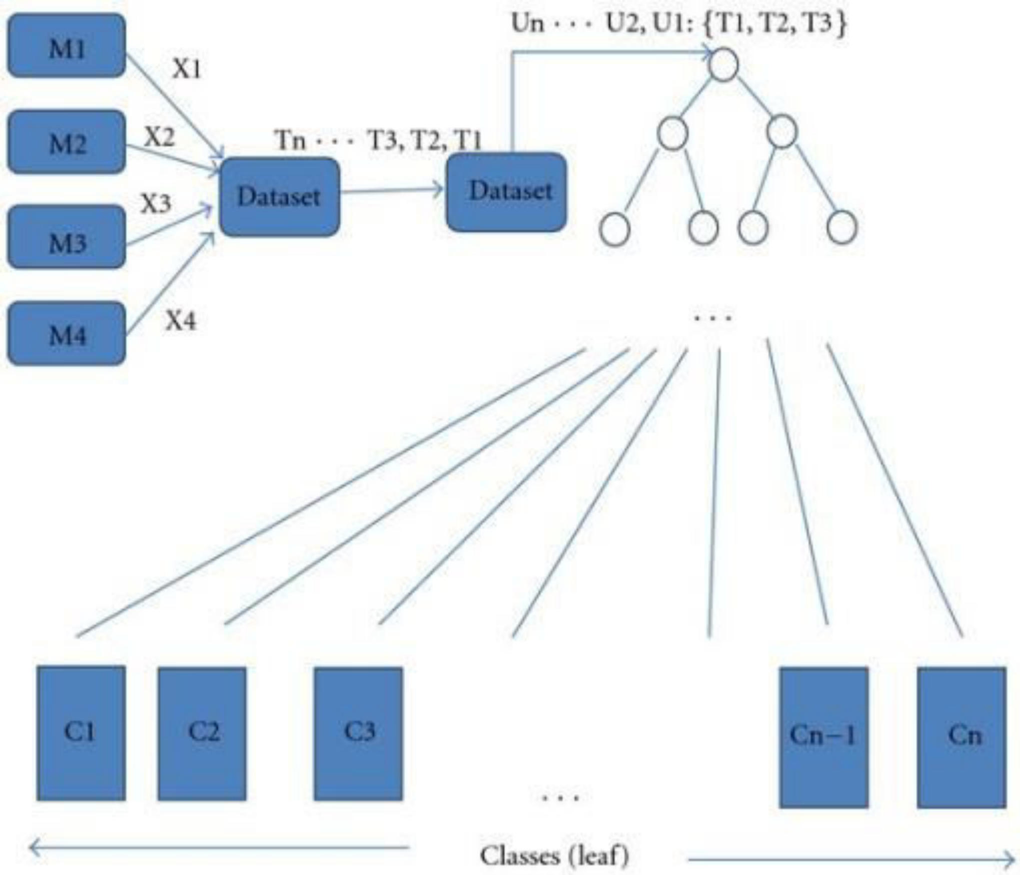
- Time
- Memory
- Sample Size

SPRINT

Tested on up to a few million examples. *Less than a day's worth!*

KDD systems operating **continuously** and **indefinitely**

# VERY
# FAST
# DECISION
# TREE

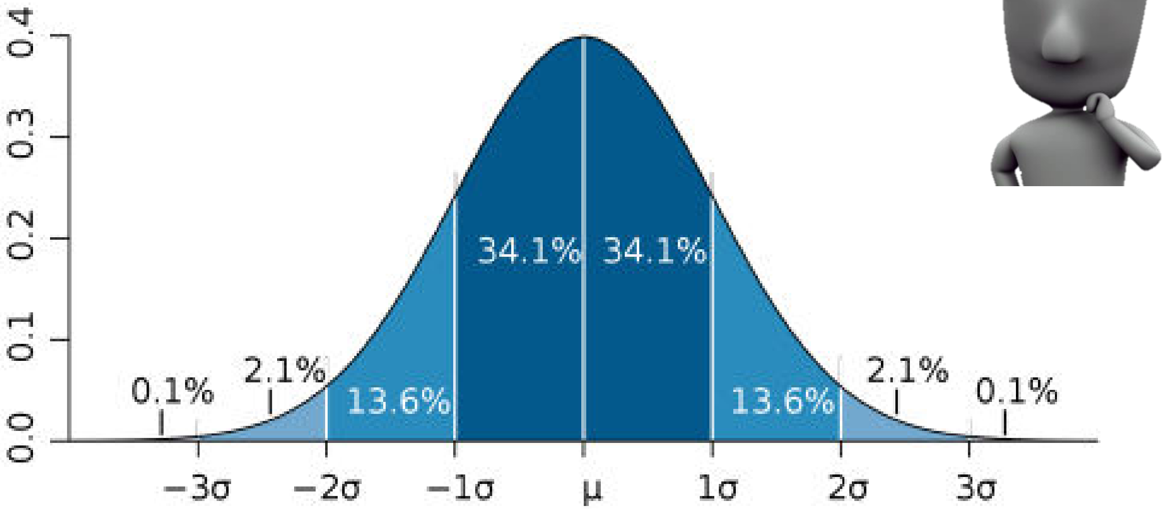


# Hoeffding Decision Tree

- Classical DT learners are limited by main memory size
- Probably, not all examples are needed to find the best attribute at a node
- How to decide how many are necessary? ***Hoeffding Bound***!

*«Suppose we have made $n$ independent observations of a variable $r$ with domain $R$, and computed their mean $\bar{r}$. The Hoeffding bound states that, with probability $1 - \delta$, the true mean of the variable is at least $\bar{r} - \epsilon$»*

$$\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}$$



# How many examples are enough?

- Let $G(X_i)$ be the heuristic measure of choice (*Information Gain*, *Gini Index*)
- $X_a$ : the attribute with the highest attribute evaluation value after *n* examples
- $X_b$ : the attribute with the second highest split evaluation function value after *n* examples
- We can compute

$$\Delta\bar{G} = \bar{G}(X_a) - \bar{G}(X_b) > \epsilon$$

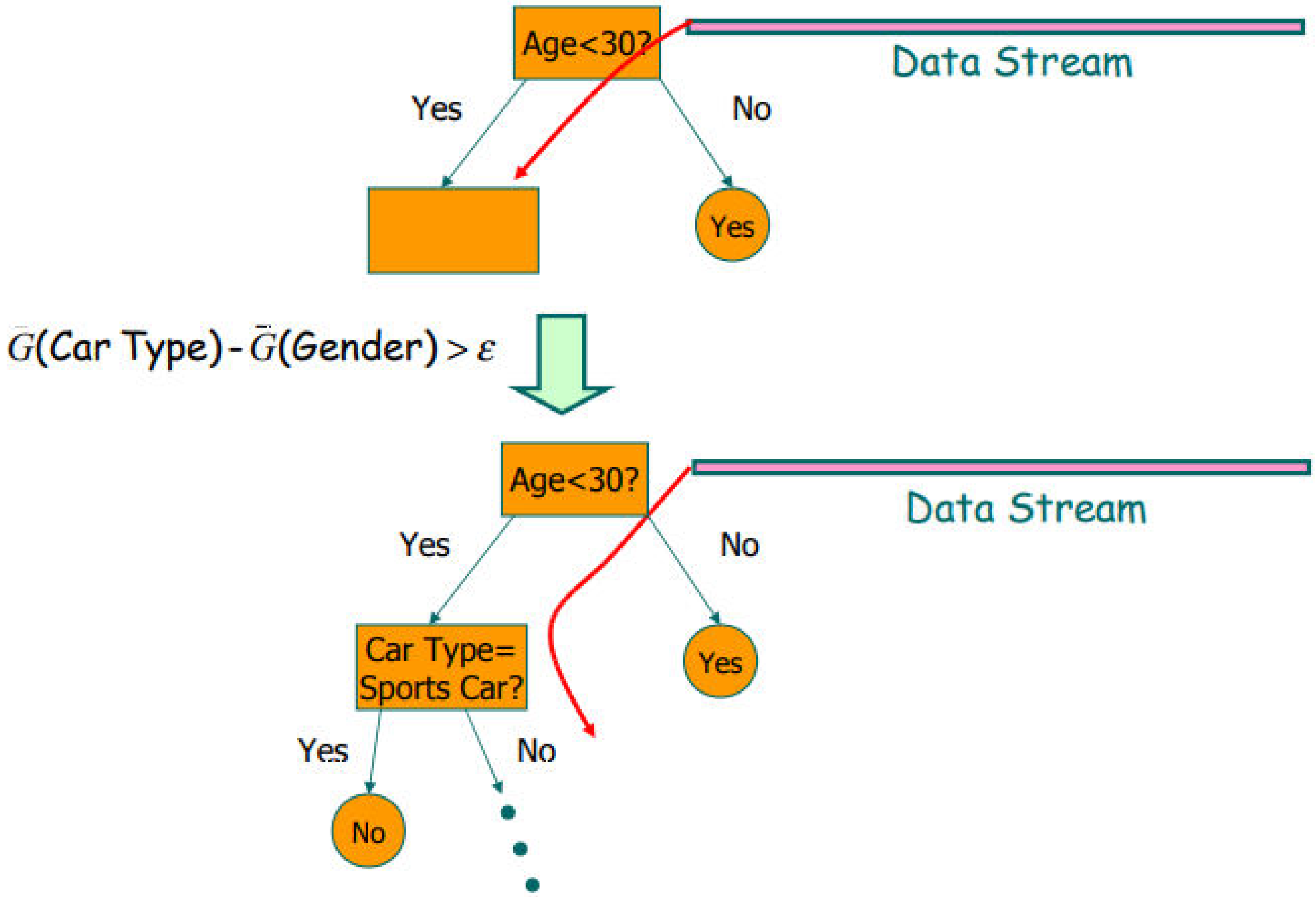
- Thanks to Hoeffding Bound, we can infer that:
  - $\Delta G \geq \Delta\bar{G} - \epsilon > 0$ with probability $1 - \delta$, where $\Delta G$ is the true difference in heuristic measure
  - This means that we can split the tree using $X_a$, and the succeeding examples will be passed to the new leaves (incremental approach)

# HT Algorithm

- Compute the heuristic measure for the attributes and determine the best two attributes
- At each node chack for the condition

$$\Delta\bar{G} = \bar{G}(X_a) - \bar{G}(X_b) > \epsilon$$

- If *true*, create child nodes based on the test at the node; else, get more examples from stream.



# In a nutshell

- Learning in Hoeffding tree is constant time per example (instance) and this means Hoeffding tree is suitable for data stream mining.
- Requires each example to be read *at most once* (incrementally built).
- With high probability, a Hoeffding tree is asymptotically identical to the decision tree built by a batch learner.

$$E[\Delta_i(HT_\delta, DT_*)] \leq \frac{\delta}{p}$$

✓

- Independent of the probability distribution generating the observations
- Built incrementally by sequential reading
- Make class predictions in parallel

✗

- What happens with ties?
- Memory used with tree expansion
- Number of candidate attributes

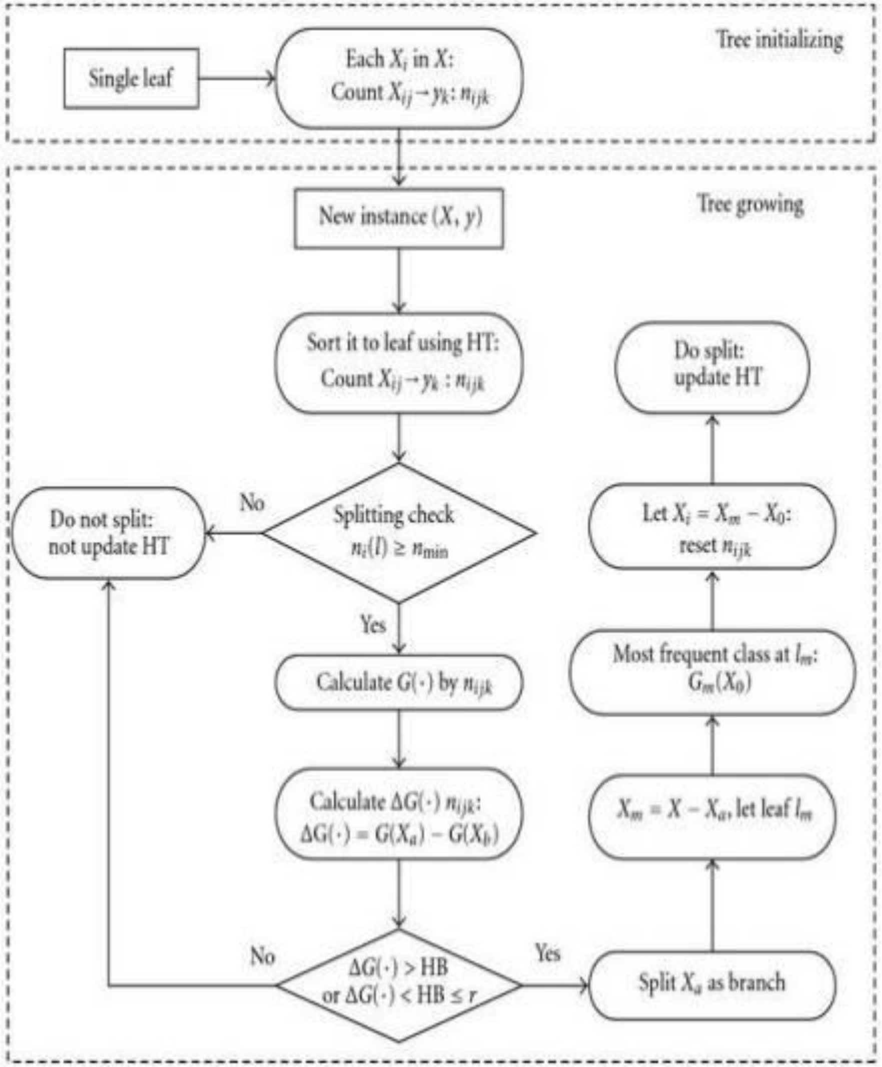
goo.gl/gBnm9h

goo.gl/QvZMC7

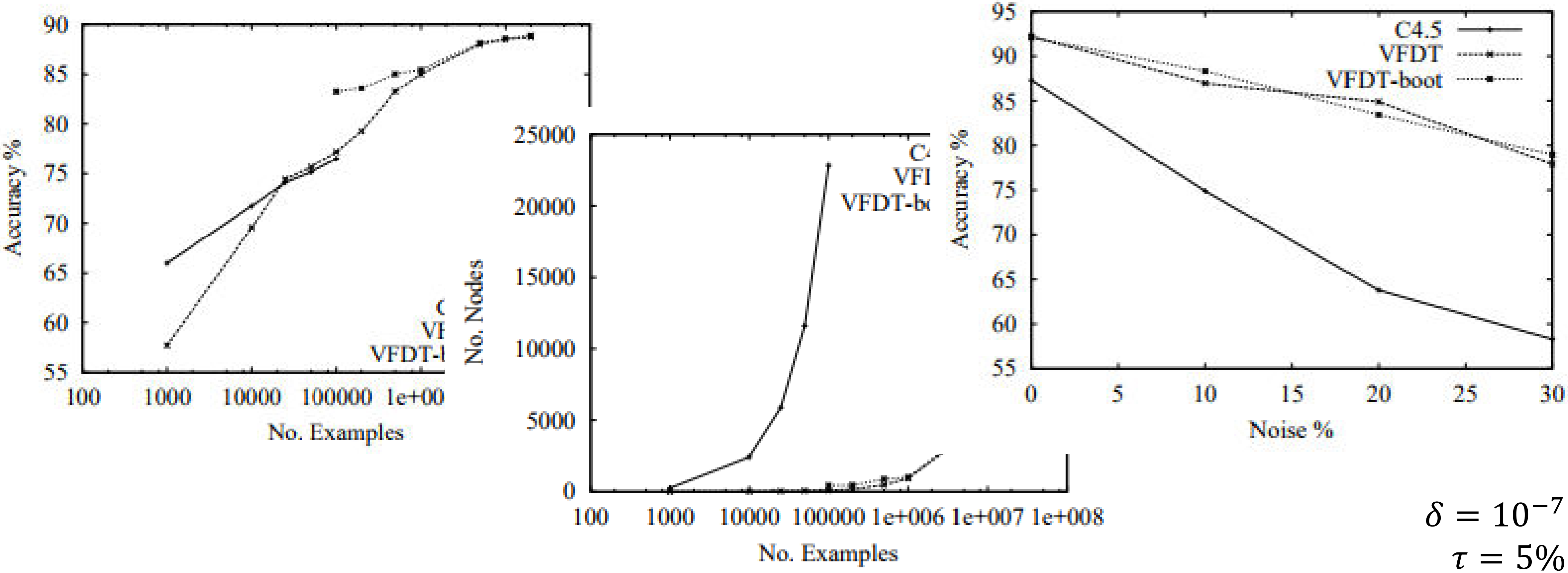
# VFDT

# VFDT (Very Fast Decision Tree)

- Hoeffding tree algorithm implementation is VFDT
- VFDT includes refinements to the HT algorithm:
    - Tie-braking algorithm
    - Recompute *G* after a user-defined #examples
    - Deactivation of inactive leaves
    - Drop of unpromising early attributes (if $\Delta G > \epsilon$)
    - *Bootstrap* with traditional learner on a small subset of data
    - Rescan of previously-seen examples


Tree initializing
Single leaf
Each $X_i$ in $X$: Count $X_{ij} \to y_k$: $n_{ijk}$
Tree growing
New instance $(X, y)$
Sort it to leaf using HT: Count $X_{ij} \to y_k$ : $n_{ijk}$
Splitting check $n_i(l) \geq n_{\min}$
No
Do not split: not update HT
Yes
Calculate $G(\cdot)$ by $n_{ijk}$
Calculate $\Delta G(\cdot)$ $n_{ijk}$: $\Delta G(\cdot) = G(X_a) - G(X_b)$
$\Delta G(\cdot) > \text{HB}$ or $\Delta G(\cdot) < \text{HB} \leq r$
No
Yes
Split $X_a$ as branch
$X_m = X - X_a$, let leaf $l_m$
Most frequent class at $l_m$: $G_m(X_0)$
Let $X_i = X_m - X_0$: reset $n_{ijk}$
Do split: update HT

# Comparison with C4.5



$\delta = 10^{-7}$

$\tau = 5\%$

$n_{min} = 200$

# A VFDT application : Web Data

- Mining the stream of Web page requests emanating from the whole University of Washington main campus.
- Useful to improve Web Caching, by predicting which hosts and pages will be requested in the near future.

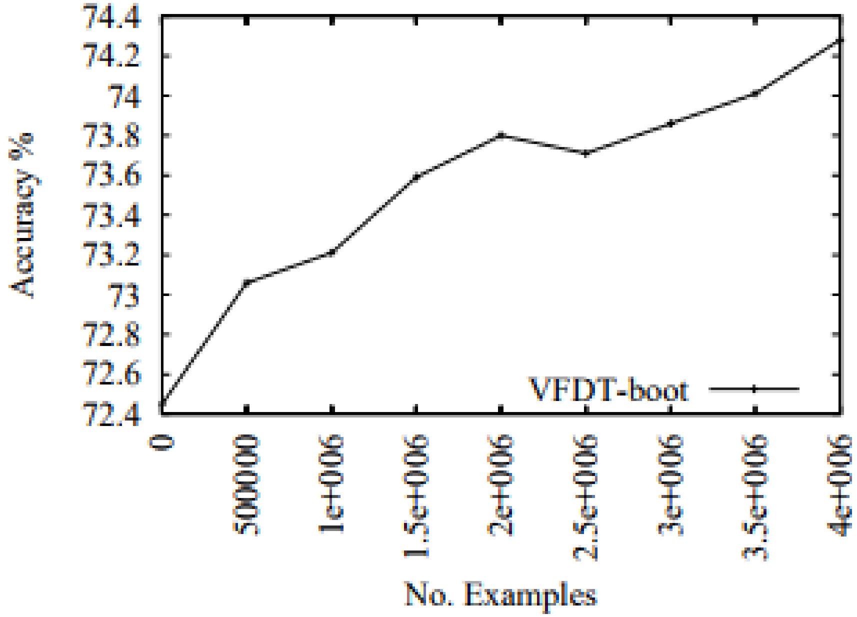


**Figure 7: Performance on Web data.**

# Future Work

- Test other applications (such as *Intrusion detection*)
- Use of *non-discretized numeric attributes*
- Use of *post-pruning*
- Use of adaptive δ
- Compare with other incremental algorithms (ID5R or SLIQ/SPRINT)
- Adapt to time-changing domains (*concept drift*)
- *Parallelization*

# QUESTIONS?

# THANK YOU!