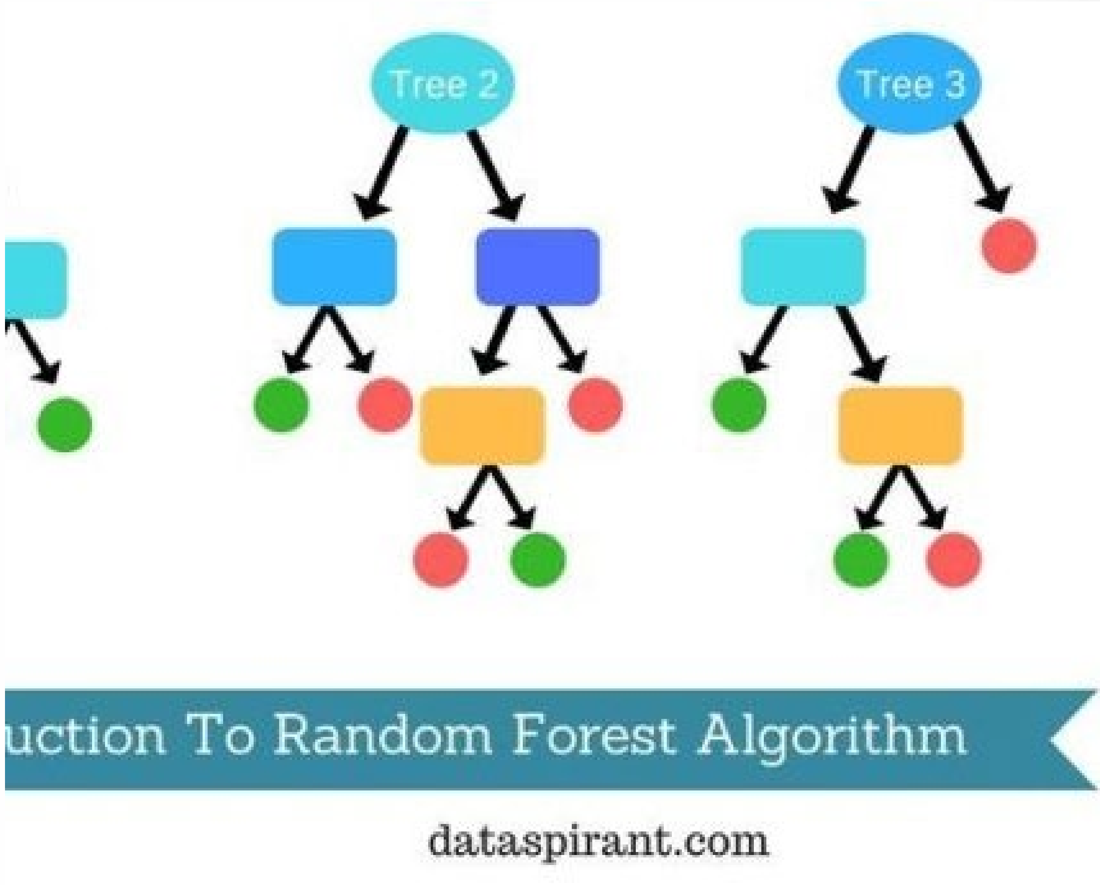
TREE-BASED ENSEMBLE LEARNING:

RANDOM FOREST…AND ITS VARIANTS

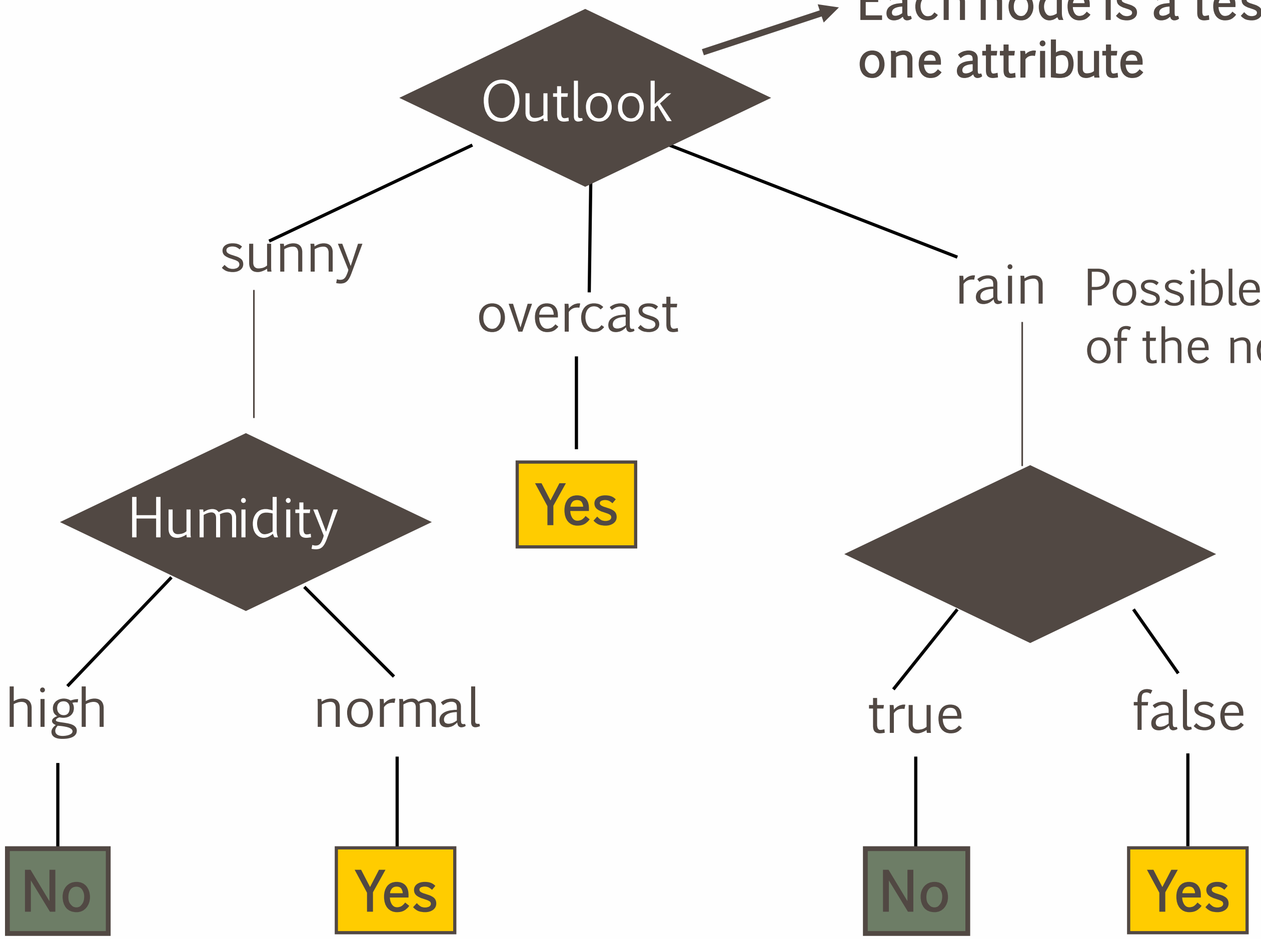
Chih-Chung Hsu
Associate Professor
ACVLab, Institute of Data Science
National Cheng Kung University



Decision trees

- Non-linear classifier
- Easy to use
- Easy to interpret
- Succeptible to overfitting but can be avoided.

Anatomy of a decision tree

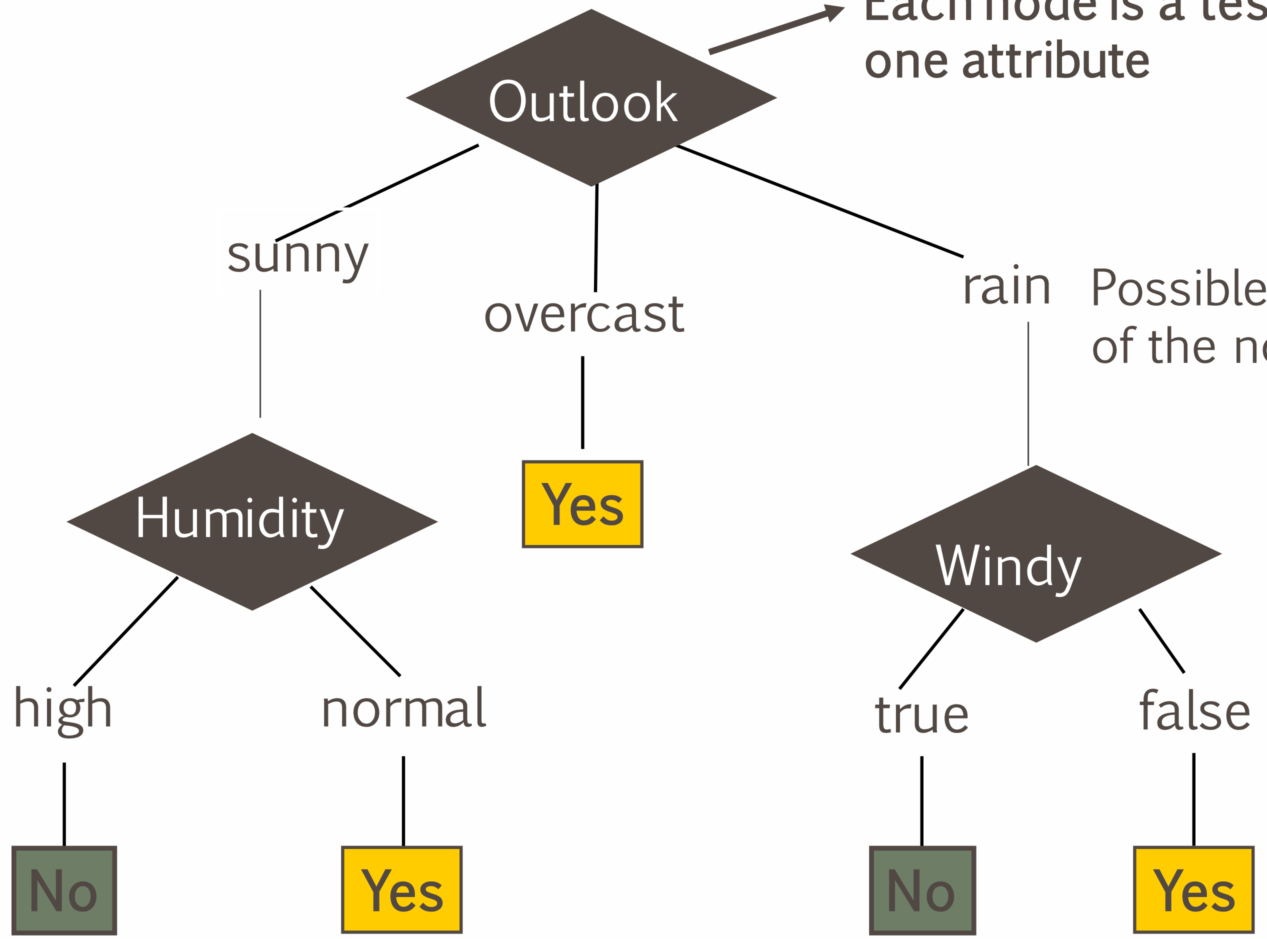


Each node is a test on one attribute

Possible attribute values of the node

Leafs are the decisions

Anatomy of a decision tree

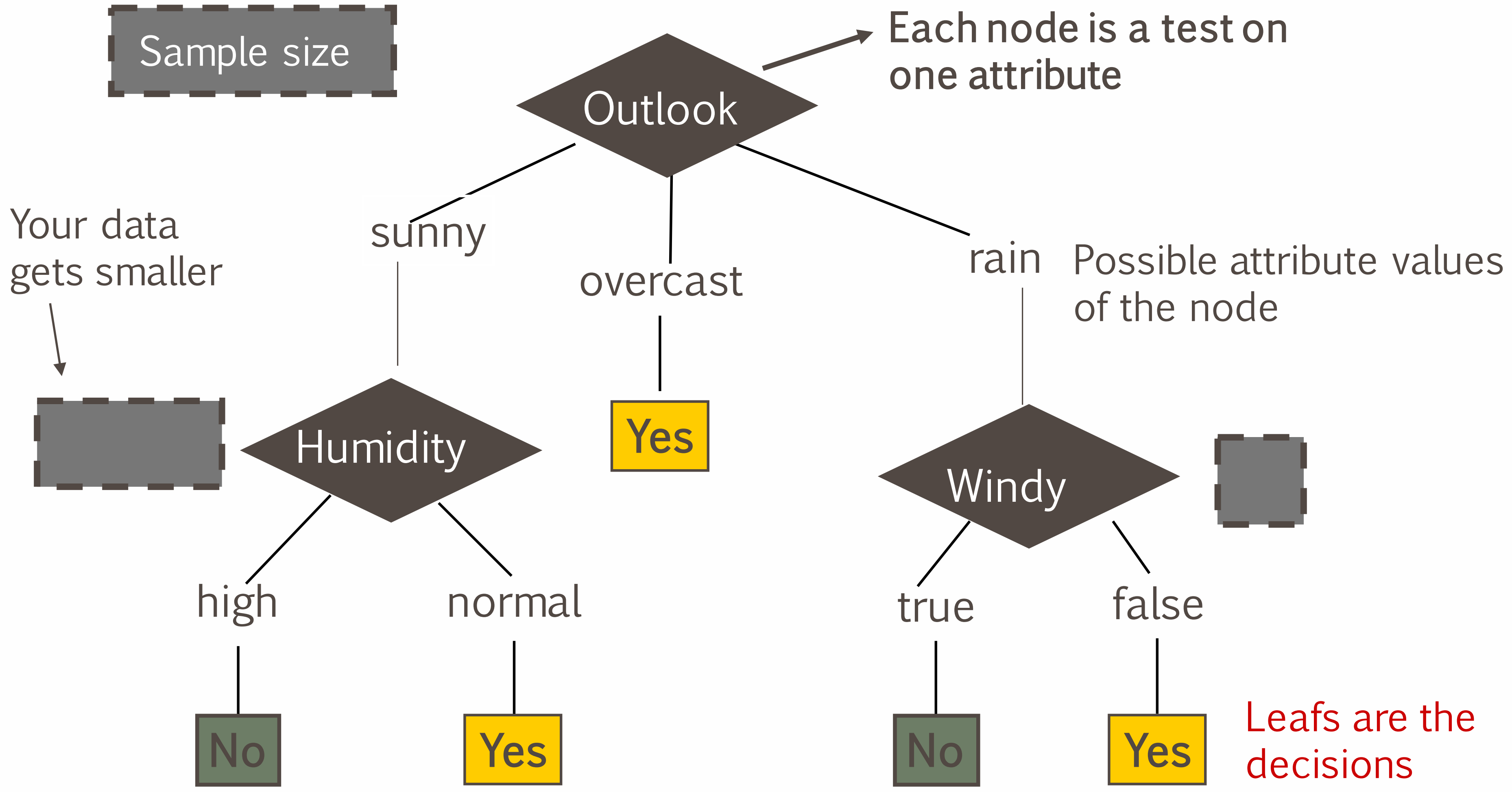


Each node is a test on one attribute

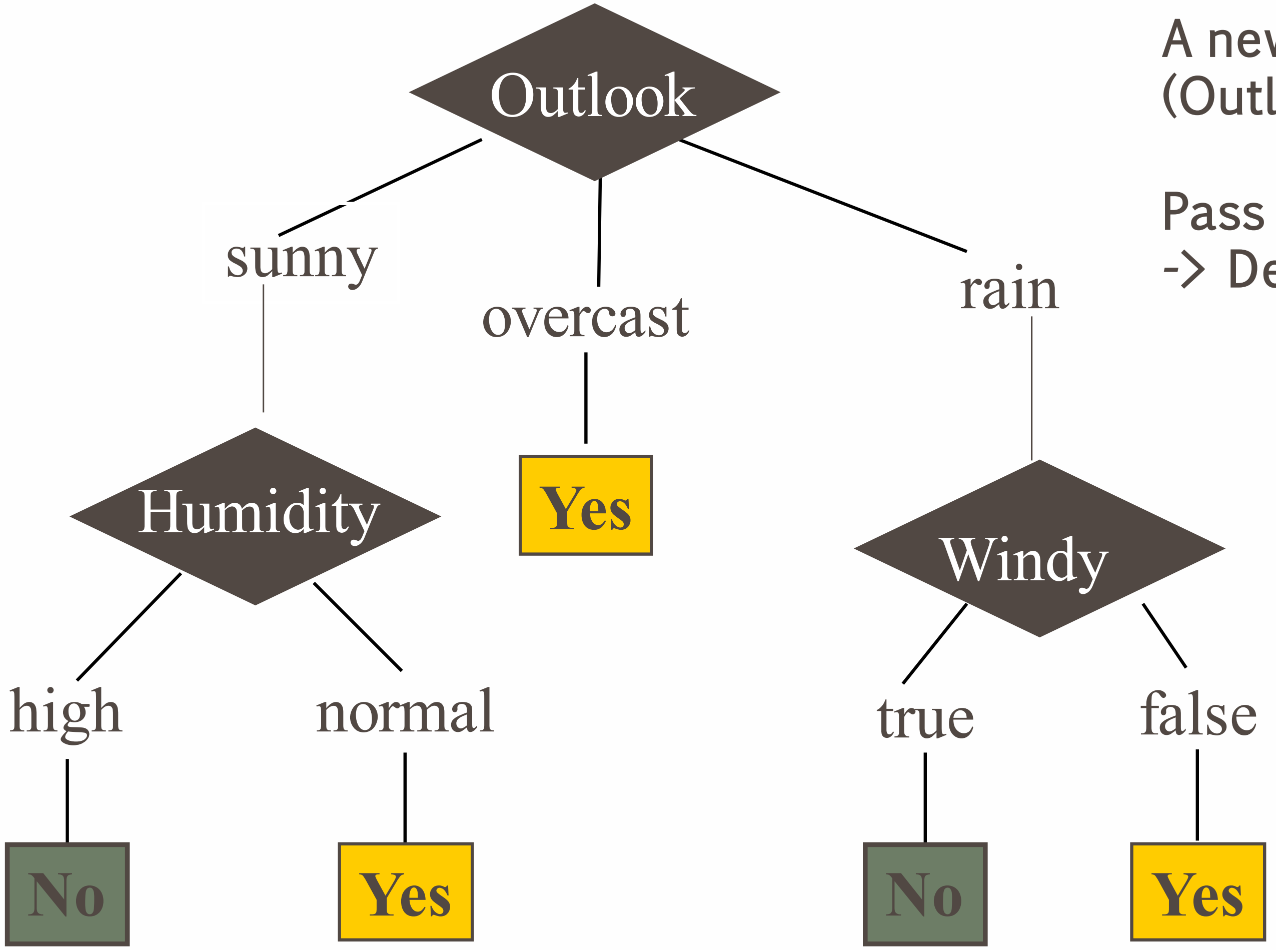
Possible attribute values of the node

Leafs are the decisions

Anatomy of a decision tree



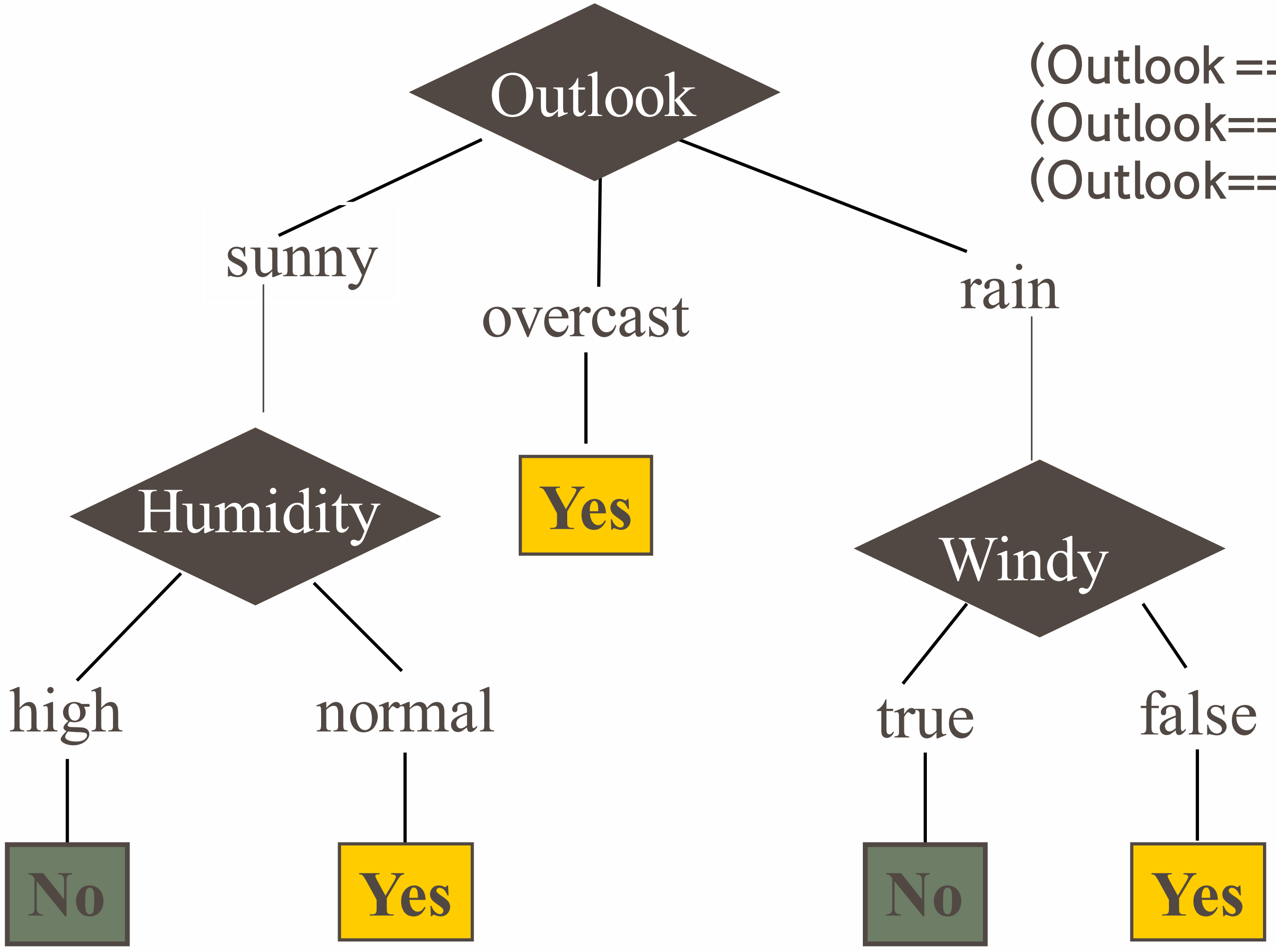
To ‘play tennis’ or not.



A new test example:
(Outlook==rain) and (not Windy==false)

Pass it on the tree
-> Decision is yes.

To ‘play tennis’ or not.



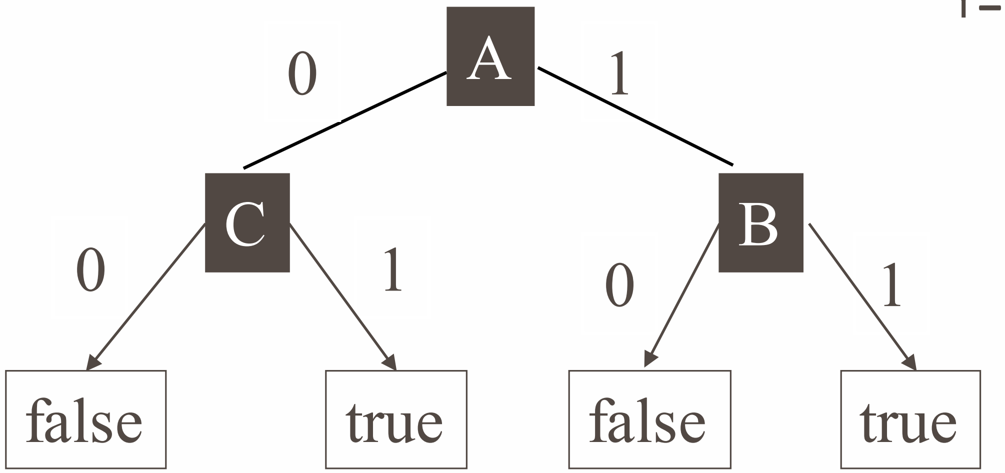
(Outlook ==overcast) -> yes
(Outlook==rain) and (not Windy==false) ->yes
(Outlook==sunny) and (Humidity=normal) ->yes

Decision trees

- Decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances.

(Outlook ==overcast)
  OR
((Outlook==rain) and (not Windy==false))
  OR
((Outlook==sunny) and (Humidity=normal))
  => yes play tennis

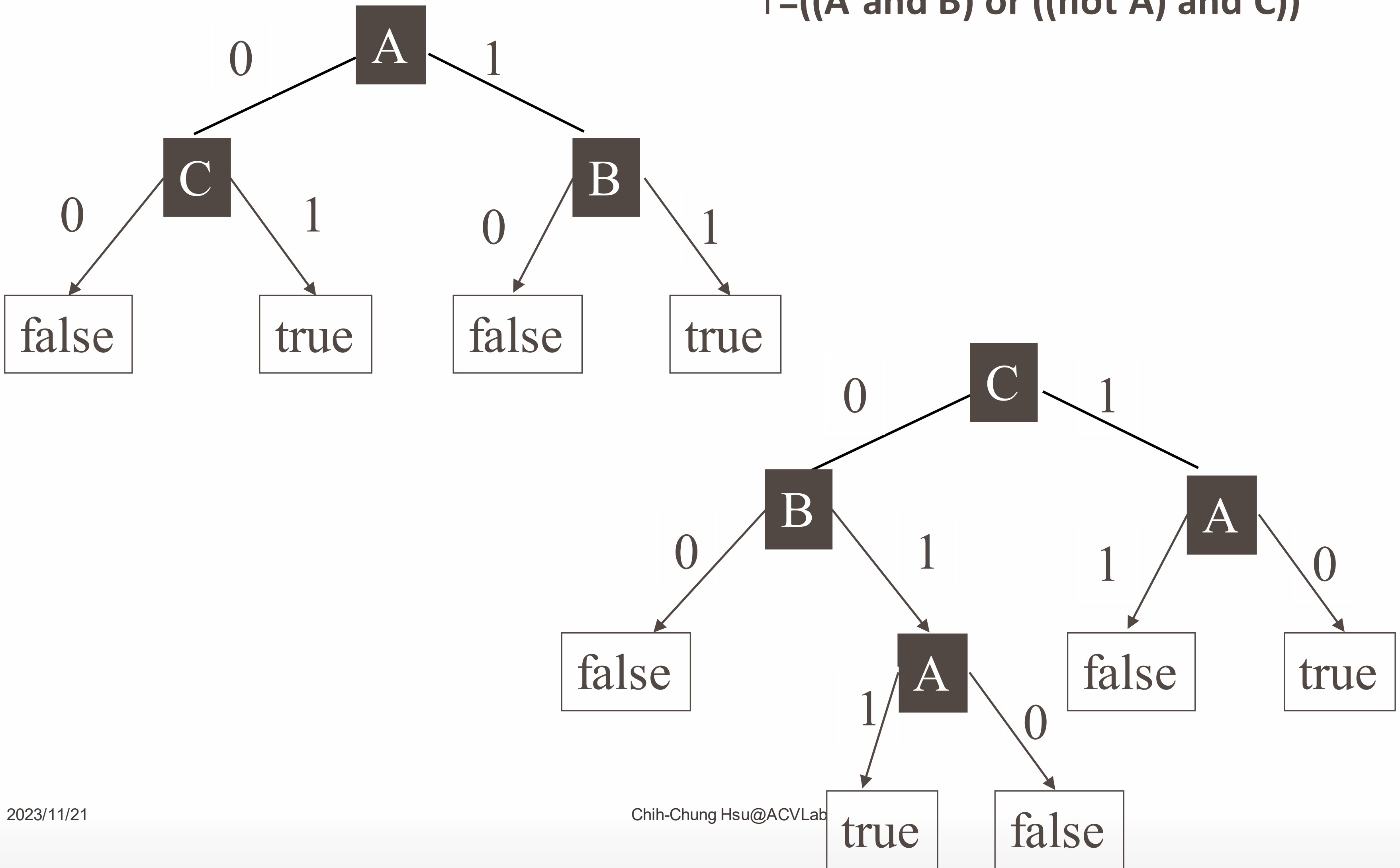
Representation



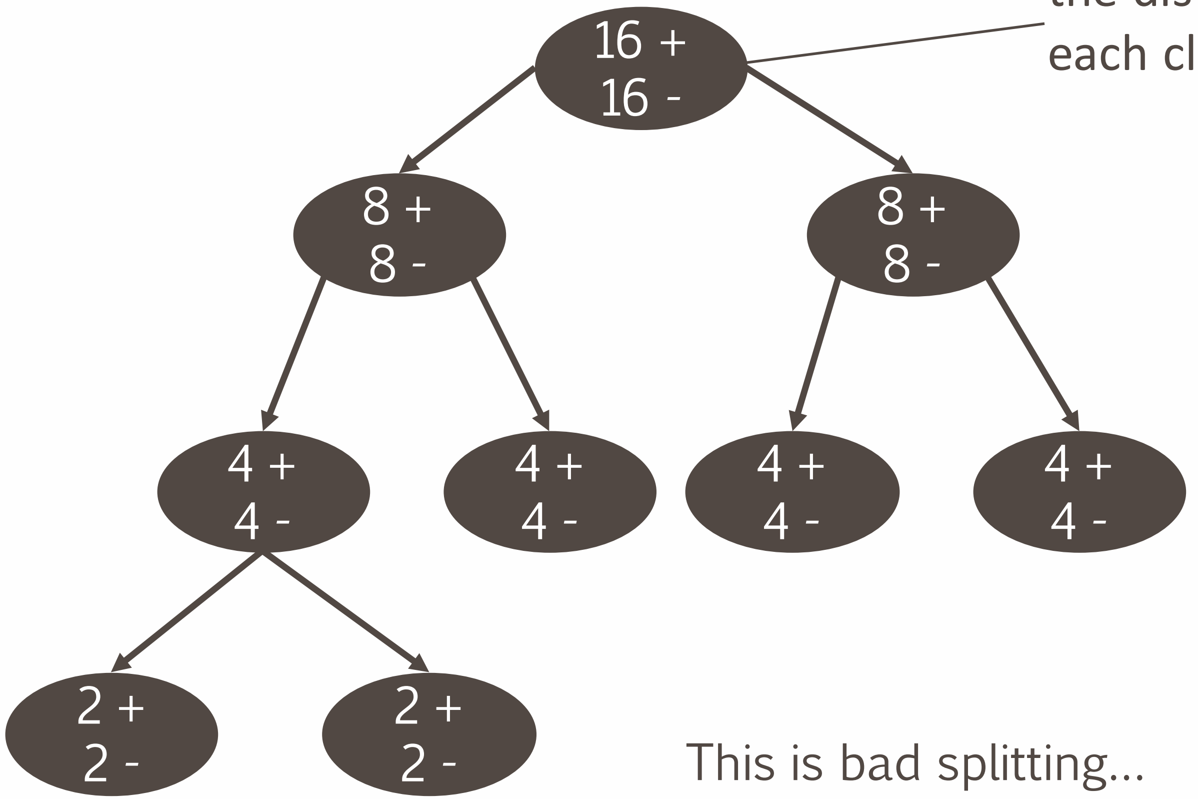
Y=((A and B) or ((not A) and C))

Same concept different representation

Y=((A and B) or ((not A) and C))



Which attribute to select for splitting?



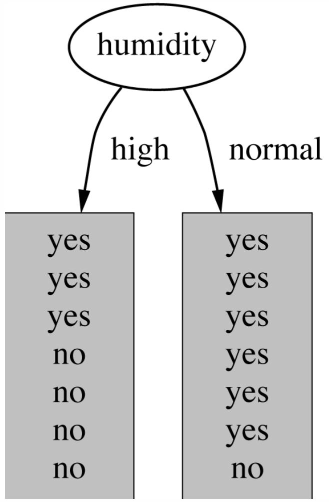
the distribution of each class (not attribute)

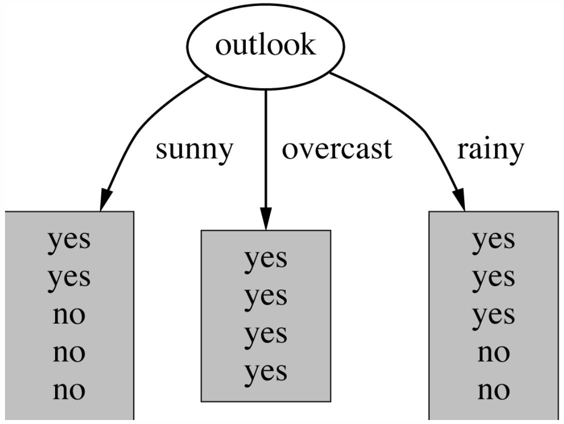
This is bad splitting...

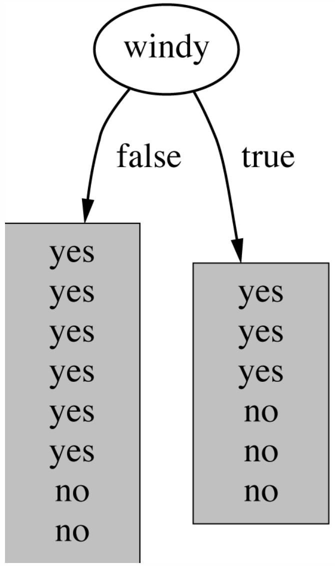
How do we choose the test ?

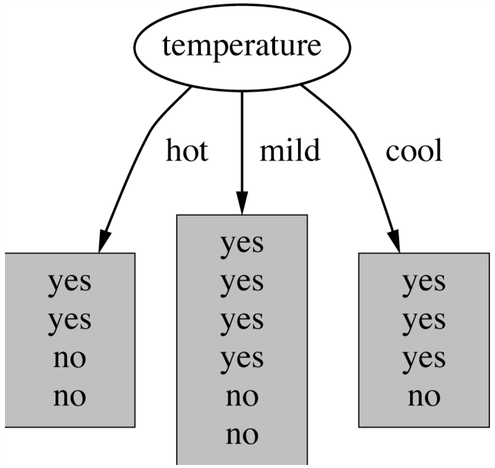
Which attribute should be used as the test?

Intuitively, you would prefer the one that *separates* the training examples as much as possible.









Information Gain

- Information gain is one criteria to decide on the attribute.

Example

Attributes Labels

X1	X2	Y	Count
T	T	+	2
T	F	+	2
F	T	-	5
F	F	+	1

Which one do we choose?
X1 or X2?

$$IG(X1,Y) = H(Y) – H(Y|X1)$$

$$H(Y) = - (5/10) \log(5/10) -5/10\log(5/10) = 1$$

$$H(Y|X1) = P(X1=T)H(Y|X1=T) + P(X1=F) H(Y|X1=F)$$

$$= 4/10 (1\log 1 + 0 \log 0) +6/10 (5/6\log 5/6 +1/6\log1/6)$$

$$= 0.39$$

Information gain (X1,Y)= 1-0.39=0.61

Which one do we choose?

X1	X2	Y	Count
T	T	+	2
T	F	+	2
F	T	-	5
F	F	+	1

Information gain (X1,Y)= 0.61

Information gain (X2,Y)= 0.12

Pick the variable which provides the most information gain about Y

Pick X1

Recurse on branches

X1	X2	Y	Count
T	T	+	2
T	F	+	2
F	T	-	5
F	F	+	1

One branch

The other branch

Notice

- The number of possible values influences the information gain.

- The more possible values, the higher the gain (the more likely it is to form small, but pure partitions)

Purity (diversity) measures

- Purity (Diversity) Measures:
- – Gini (population diversity)
- – Information Gain
- – Chi-square Test

Overfitting

- You can perfectly fit to any training data
  - Zero bias, high variance

- Two approaches:
  - 1. Stop growing the tree when further splitting the data does not yield an improvement
  - 2. Grow a full tree, then prune the tree, by eliminating nodes.

Bagging

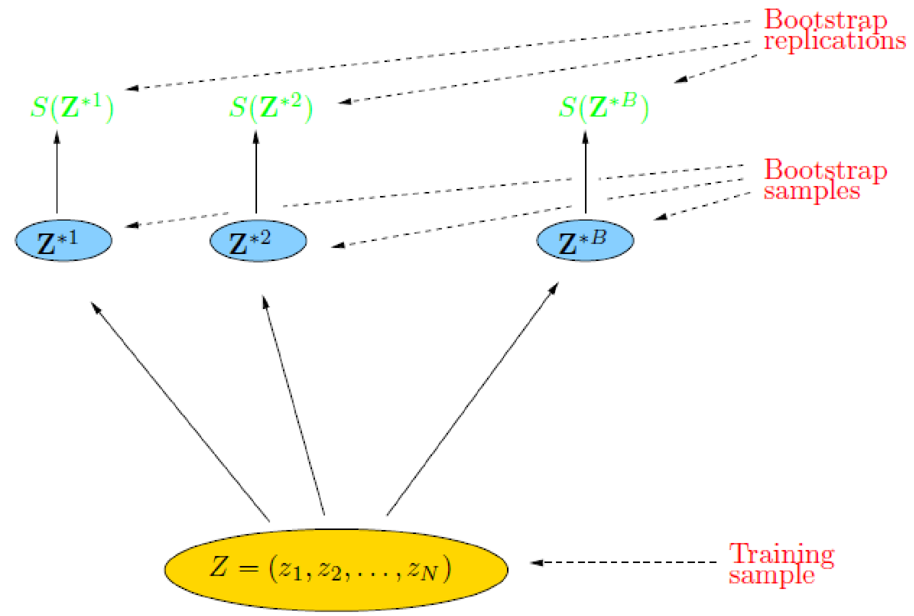
- Bagging or bootstrap aggregation
  - a technique for reducing the variance of an estimated prediction function.

- For classification, a committee of trees each
  - cast a vote for the predicted class.

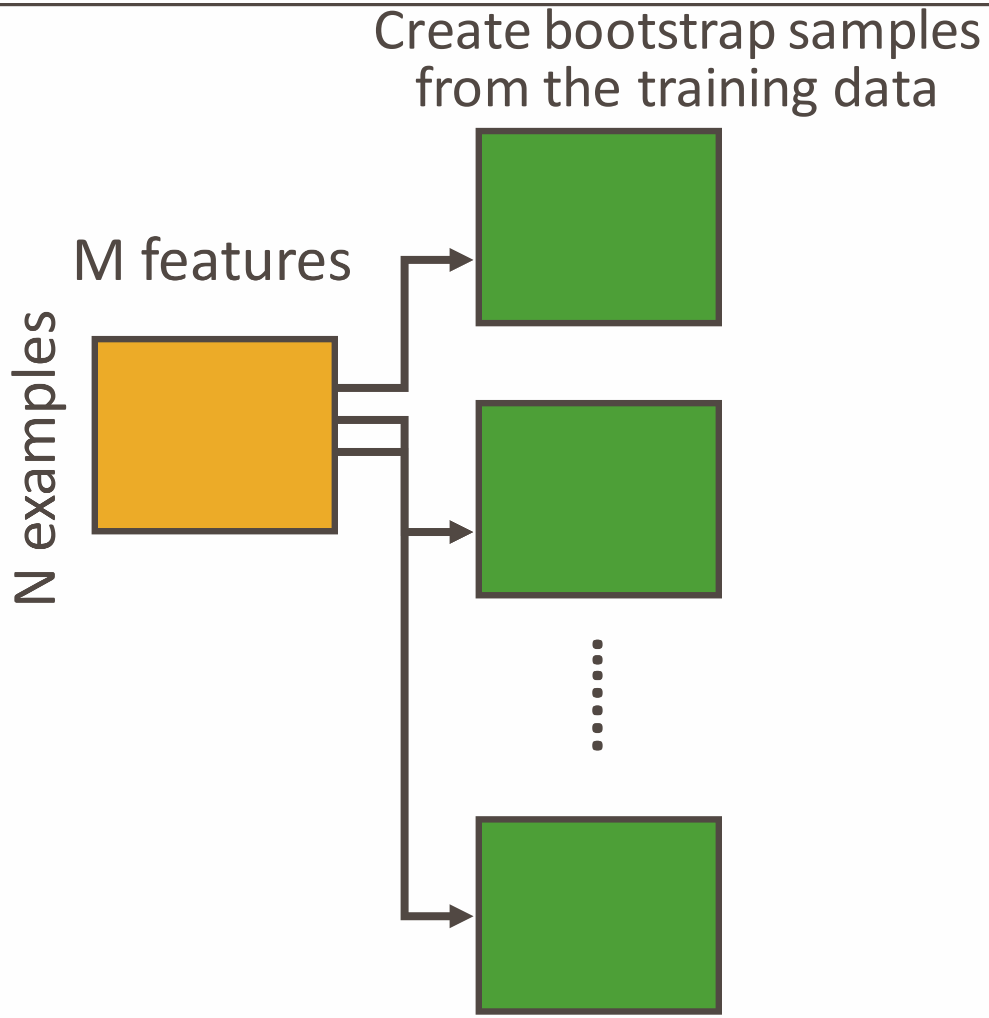
Bootstrap

The basic idea:

randomly draw datasets *with replacement* from the training data, each sample *the same size as the original training set*

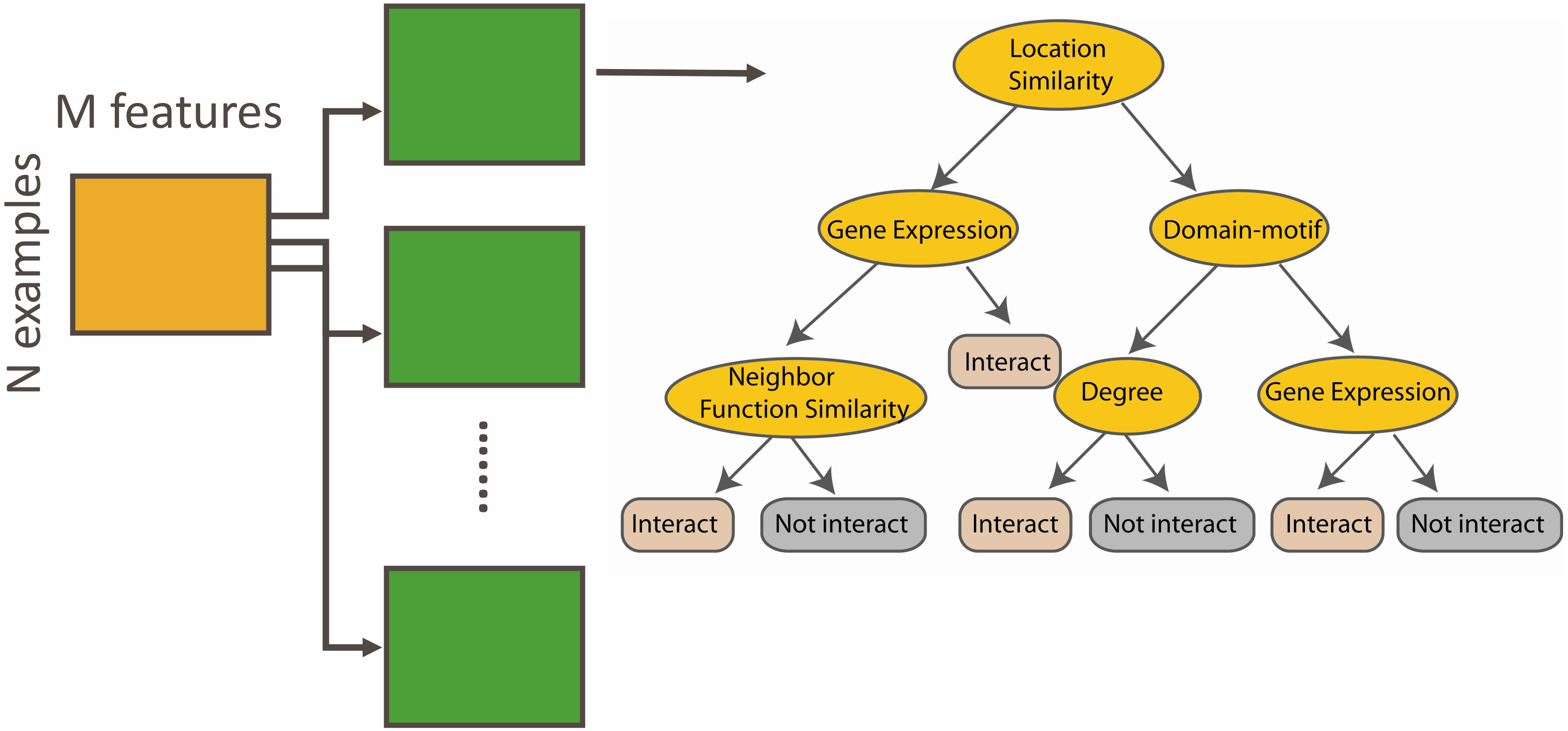


Bagging

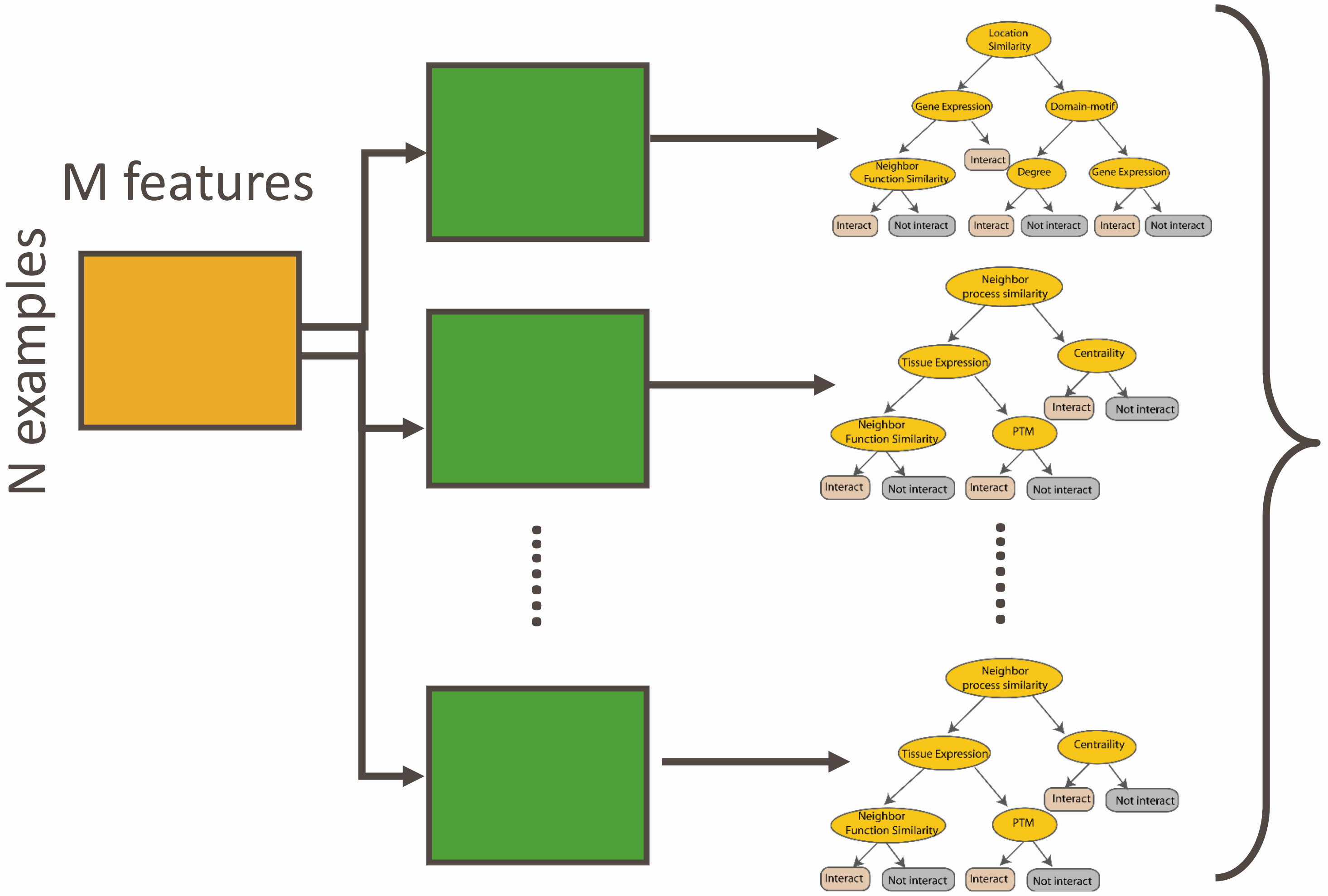

Create bootstrap samples
from the training data
M features
N examples

Random Forest Classifier

Construct a decision tree



Random Forest Classifier



M features

N examples

Take the majority vote

Bagging

$\mathbf{Z} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$

Z^{*b} where = 1,.., B..

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x).$$

The prediction at input x when bootstrap sample *b* is used for training

http://www-stat.stanford.edu/~hastie/Papers/ESLII.pdf (Chapter 8.7)

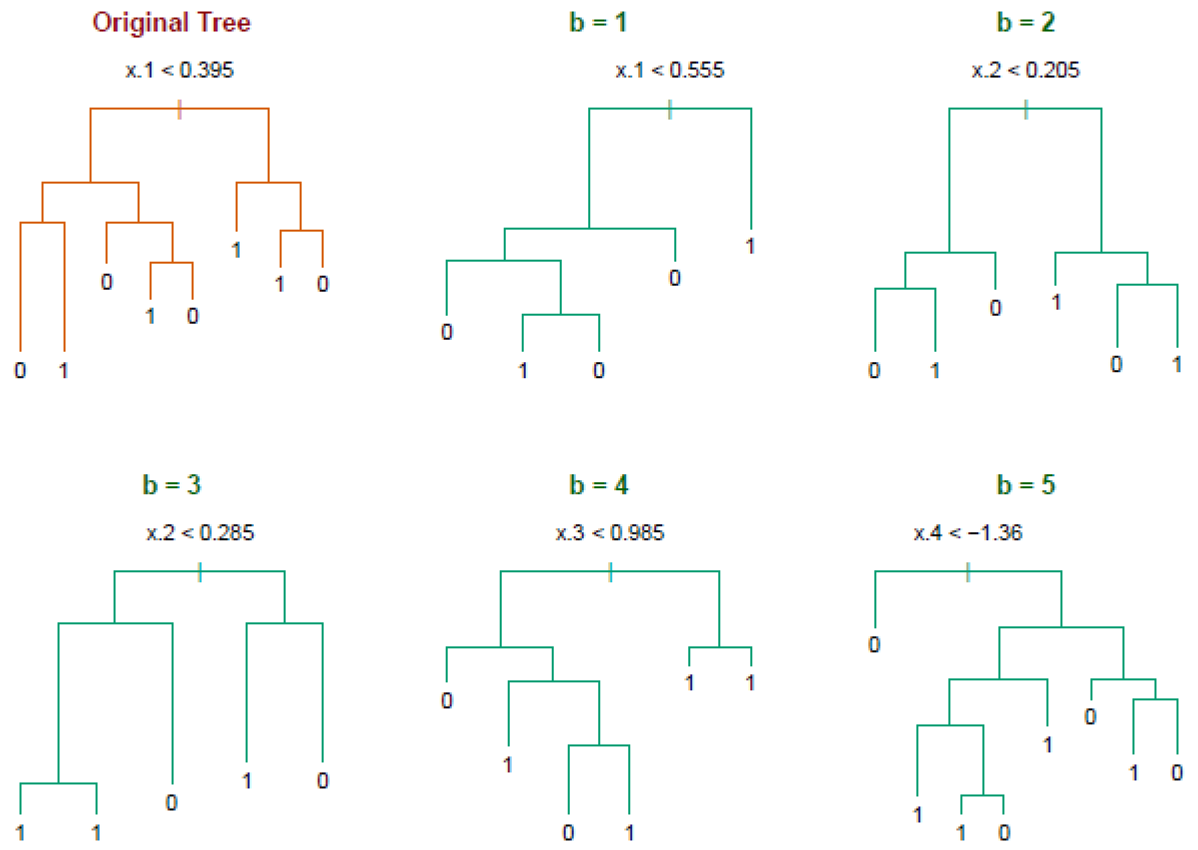
Bagging : an simulated example

- Generated a sample of size N = 30, with two classes and p = 5 features, each having a standard Gaussian distribution with pairwise Correlation 0.95.

- The response Y was generated according to
  - $\Pr(Y = 1|x1 \leq 0.5) = 0.2$,
  - $\Pr(Y = 0|x1 > 0.5) = 0.8$.

Bagging

Notice the bootstrap trees are different than the original tree



Bagging

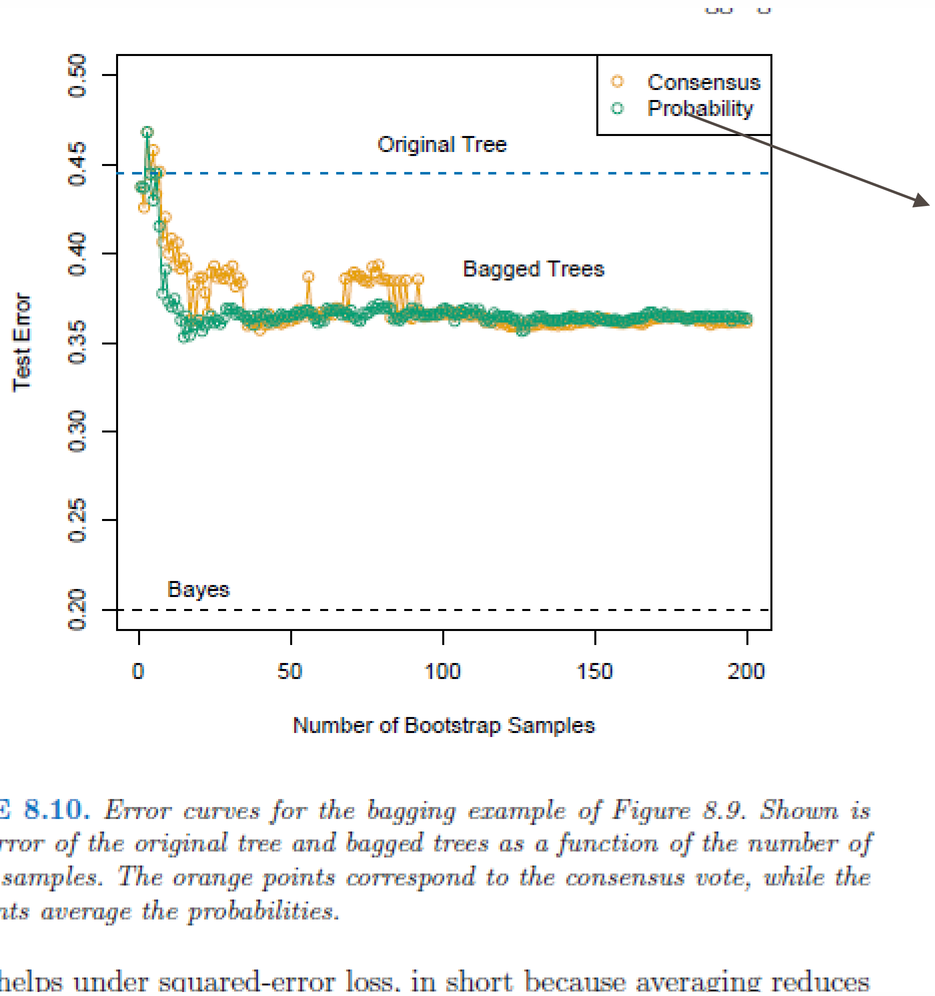


FIGURE 8.10. *Error curves for the bagging example of Figure 8.9. Shown is the test error of the original tree and bagged trees as a function of the number of bootstrap samples. The orange points correspond to the consensus vote, while the green points average the probabilities.*

bagging helps under squared-error loss, in short because averaging reduces

Treat the voting Proportions as probabilities

Hastie

http://www-stat.stanford.edu/~hastie/Papers/ESLII.pdf Example 8.7.1

Random forest classifier

- Random forest classifier, an extension to
- bagging which uses *de-correlated* trees.

MODEL ENSEMBLE

TRICKS

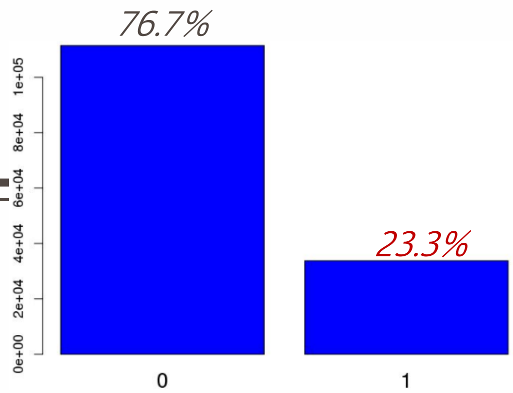
source: https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf
reading moderator: Ondrej Kaššák

Kaggle Competition

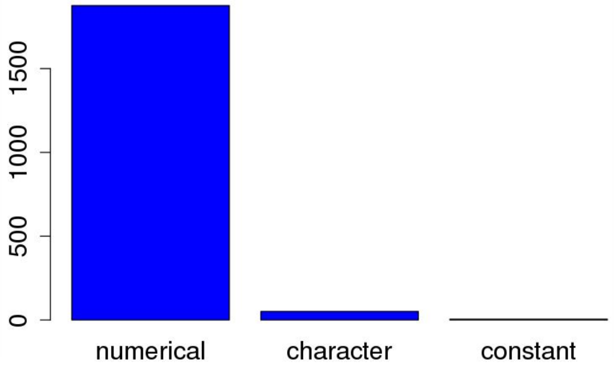
- Objective: Predict whether customers will respond to a direct mail loan offer
- Customers: 145,231
- Independent variables: 1932
- “Anonymous” features
- Dependent variable:
  - target = 0: DID NOT RESPOND
  - target = 1: RESPONDED
- Training sets: 96,820 obs.
- Testing sets: 48,411 obs.

Dataset facts

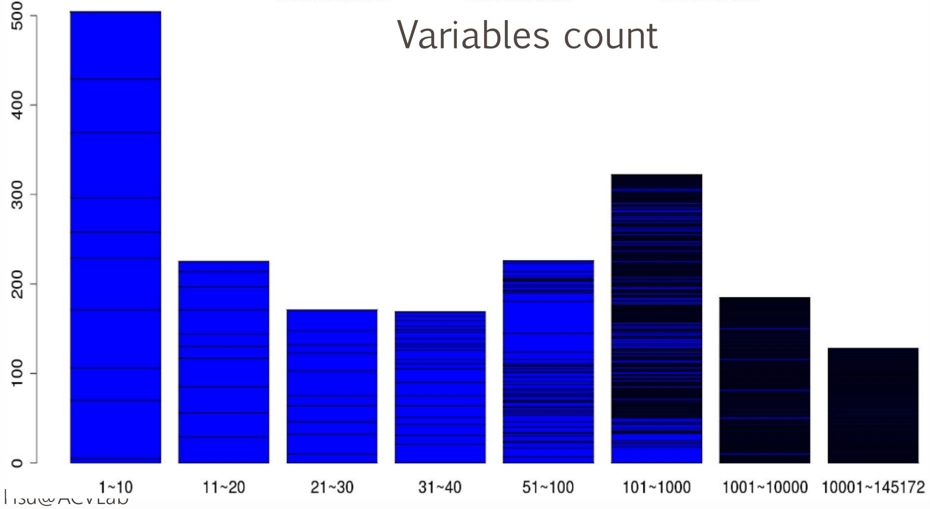
- Target=0 obs.: 111,458
- Target=1 obs.: 33,773
- Numerical variables: 1,876
- Character variables: 51
- Constant variables: 5
- Variable level counts:
  - 67.0% columns have
  - levels <= 100



Class 0 and 1 count



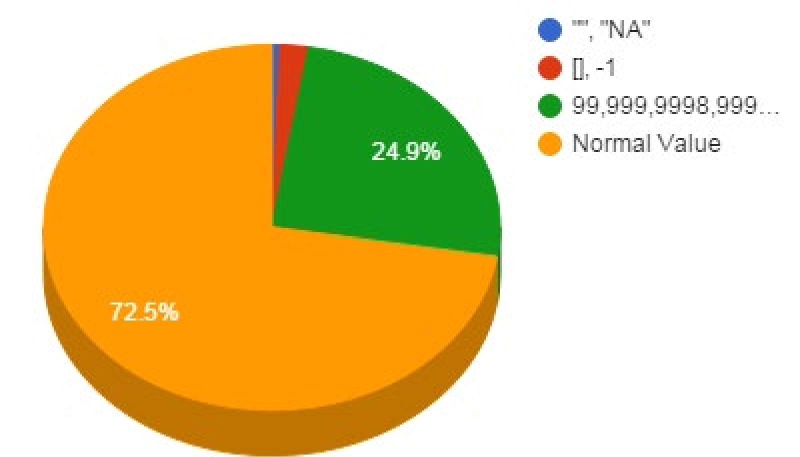
Variables count

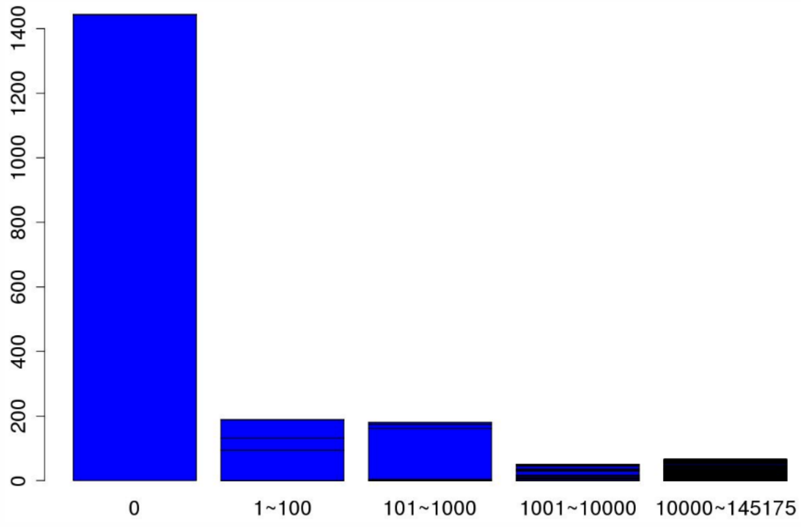


Count of levels for each column

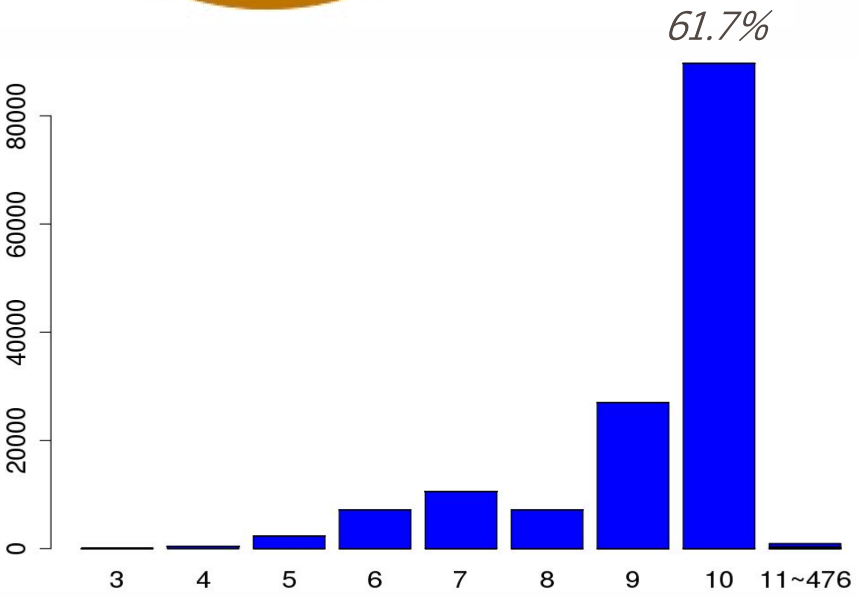
Missing values

- “”, “NA”: 0.6%
- “[]”, -1: 2.0%
- -99999, 96, …, 999, …, 99999999: 24.9%
- 25.3% columns have missing values





Count of NAs in each column

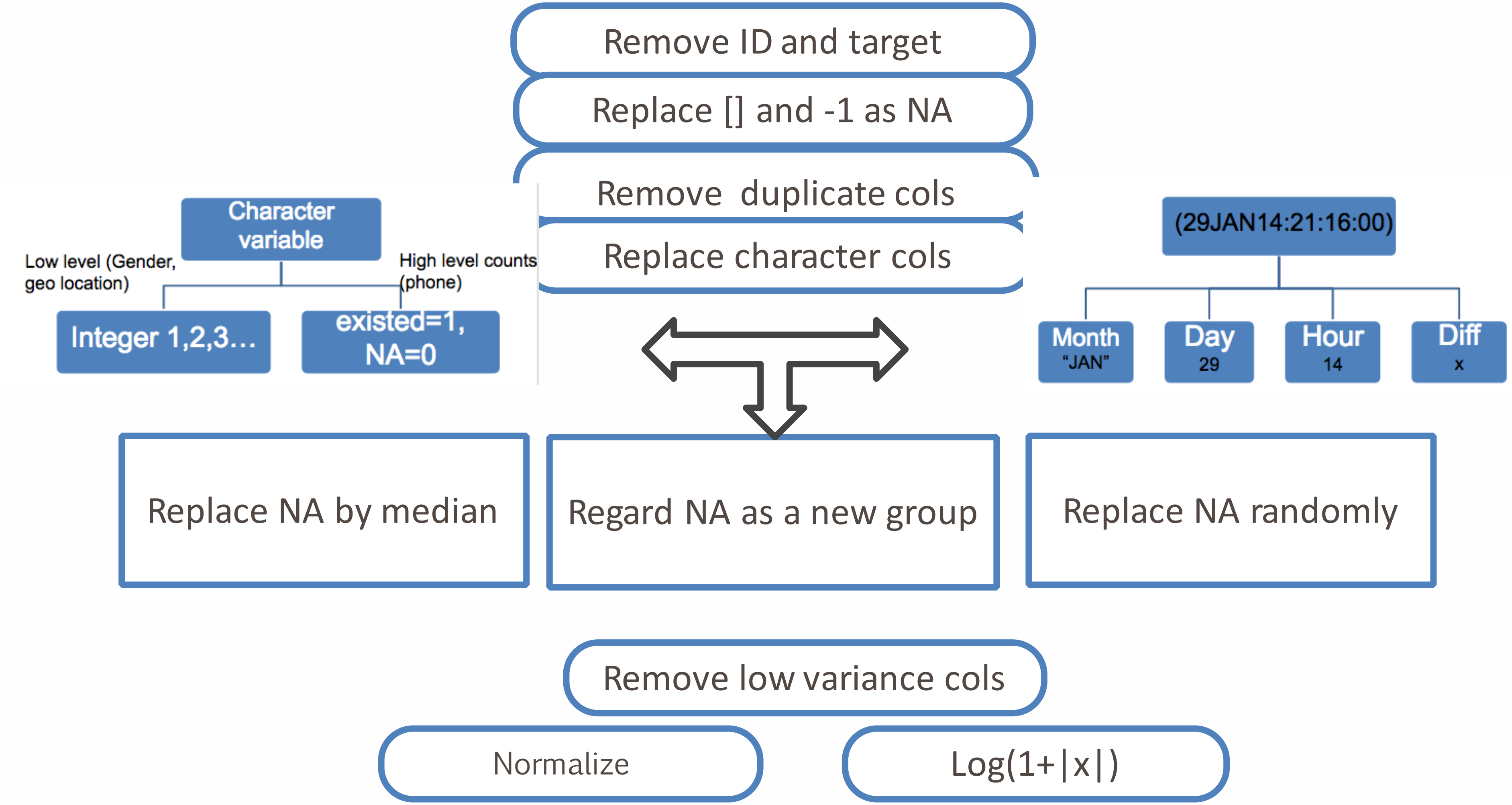


Count of NAs in each row

Challenges for classification

- Huge Dataset (145,231 X 1932)
- “Anonymous” features
- Uneven distribution of response variable
- 27.6% of missing values
- Deal with both numerical and categorical variables
- Undetermined portion of Categorical variables
- Data pre-processing complexity

Data preprocessing



Remove ID and target

Replace [] and -1 as NA

Remove duplicate cols

Replace character cols

Character variable

Low level (Gender, geo location)

High level counts (phone)

Integer 1,2,3…

existed=1, NA=0

(29JAN14:21:16:00)

Month "JAN"

Day 29

Hour 14

Diff x

Replace NA by median

Regard NA as a new group

Replace NA randomly

Remove low variance cols

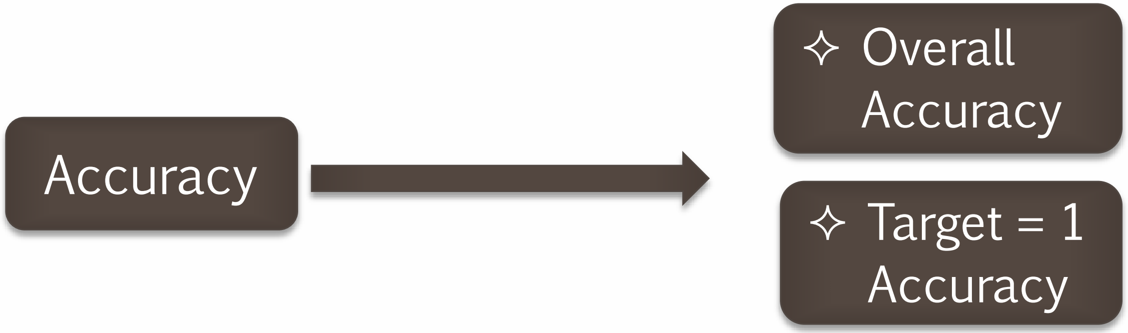
Normalize

Log(1+|x|)

Methodology

- K Nearest Neighbor (KNN)
- Support Vector Machine (SVM)
- Logistic Regression
- Random Forest
- XGBoost (eXtreme Gradient Boosting)
- Stacking

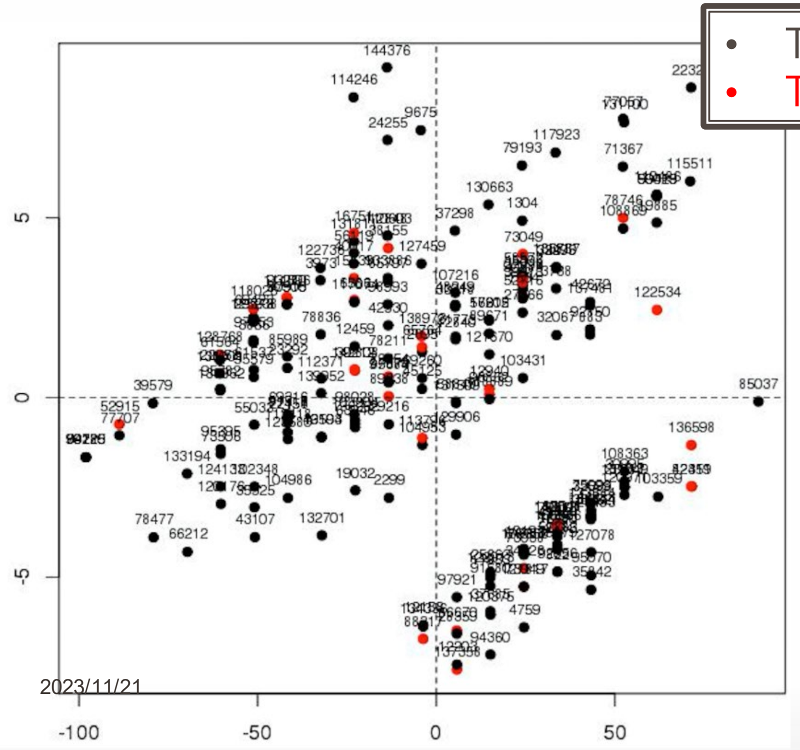
K Nearest Neighbor (KNN)

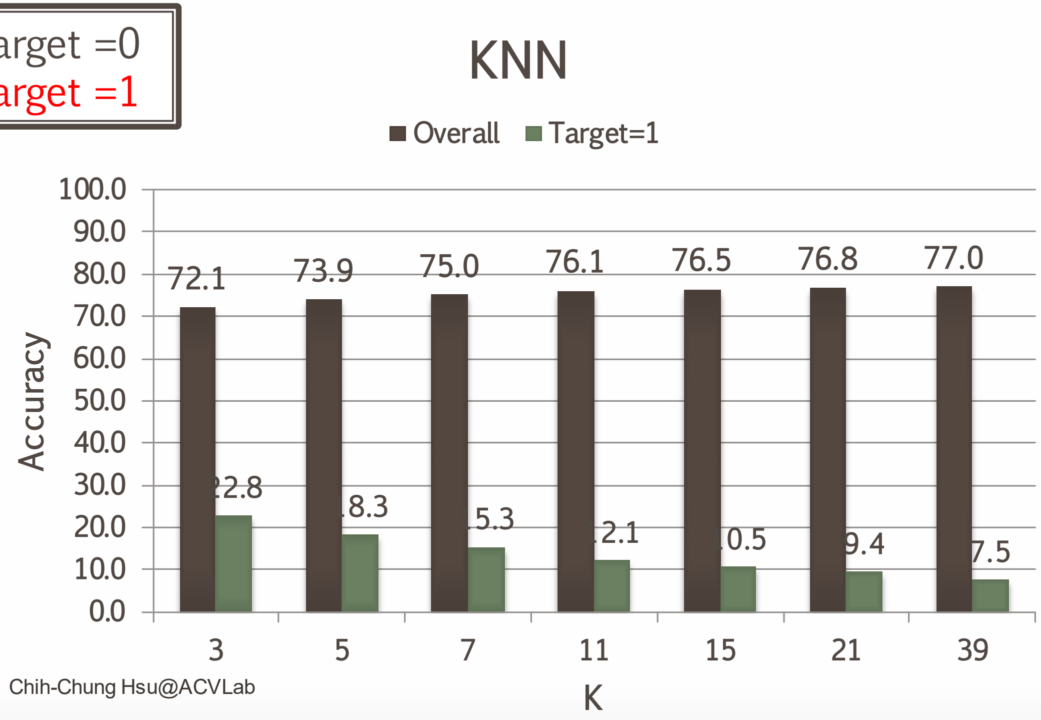


Accuracy

✧ Overall Accuracy

✧ Target = 1 Accuracy





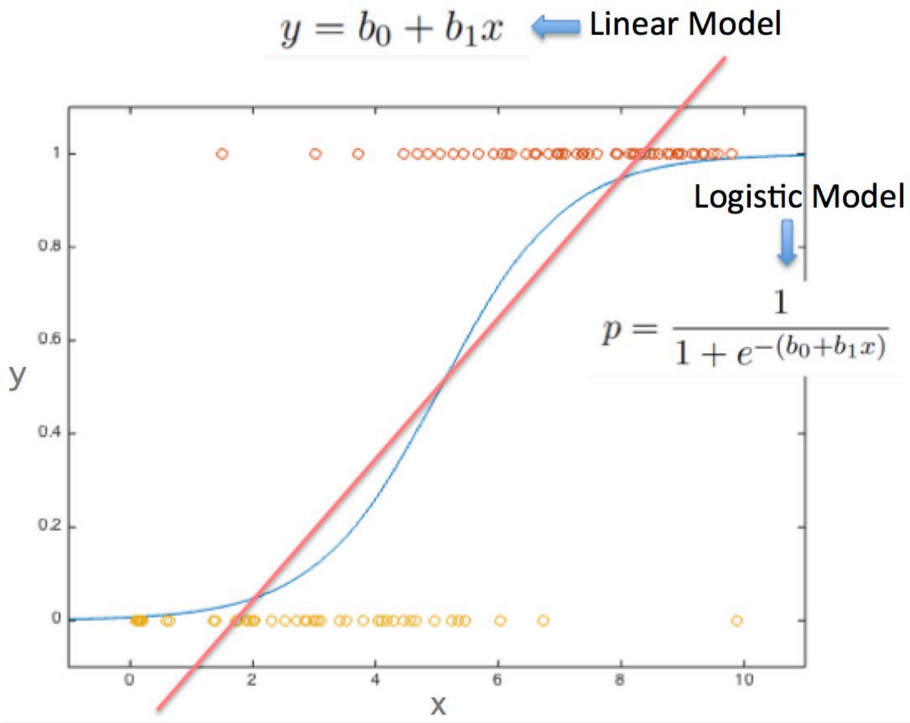
Support Vector Machine (SVM)

- Expensive; takes long time for each run
- Good results for numerical data

Confusion matrix		Prediction	
Truth		0	1
	0	19609	483
	1	5247	803

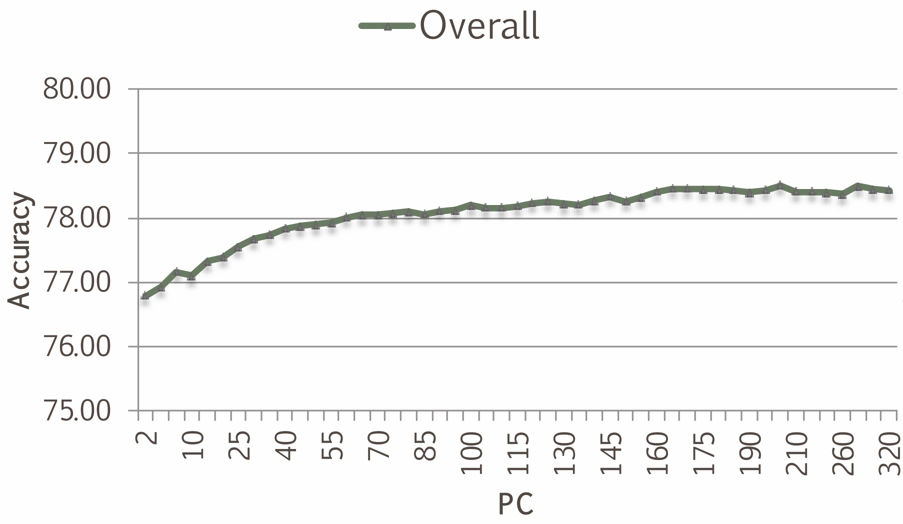
	Accuracy
Overall	78.1%
Target = 1	13.3%
Target = 0	97.6%

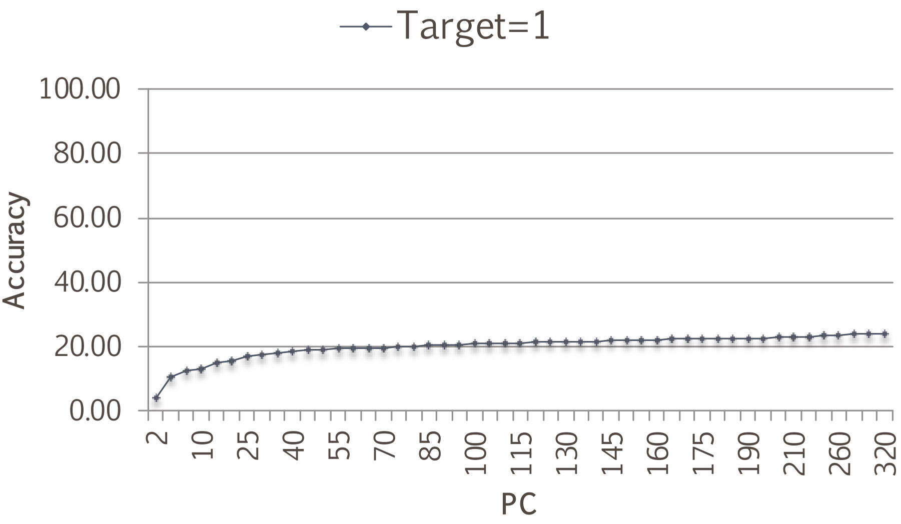
Logistic Regression



- Logistic regression is a regression model where the dependent variable is categorical.
- Measures the relationship between dependent variable and independent variables by estimating probabilities

Logistic Regression





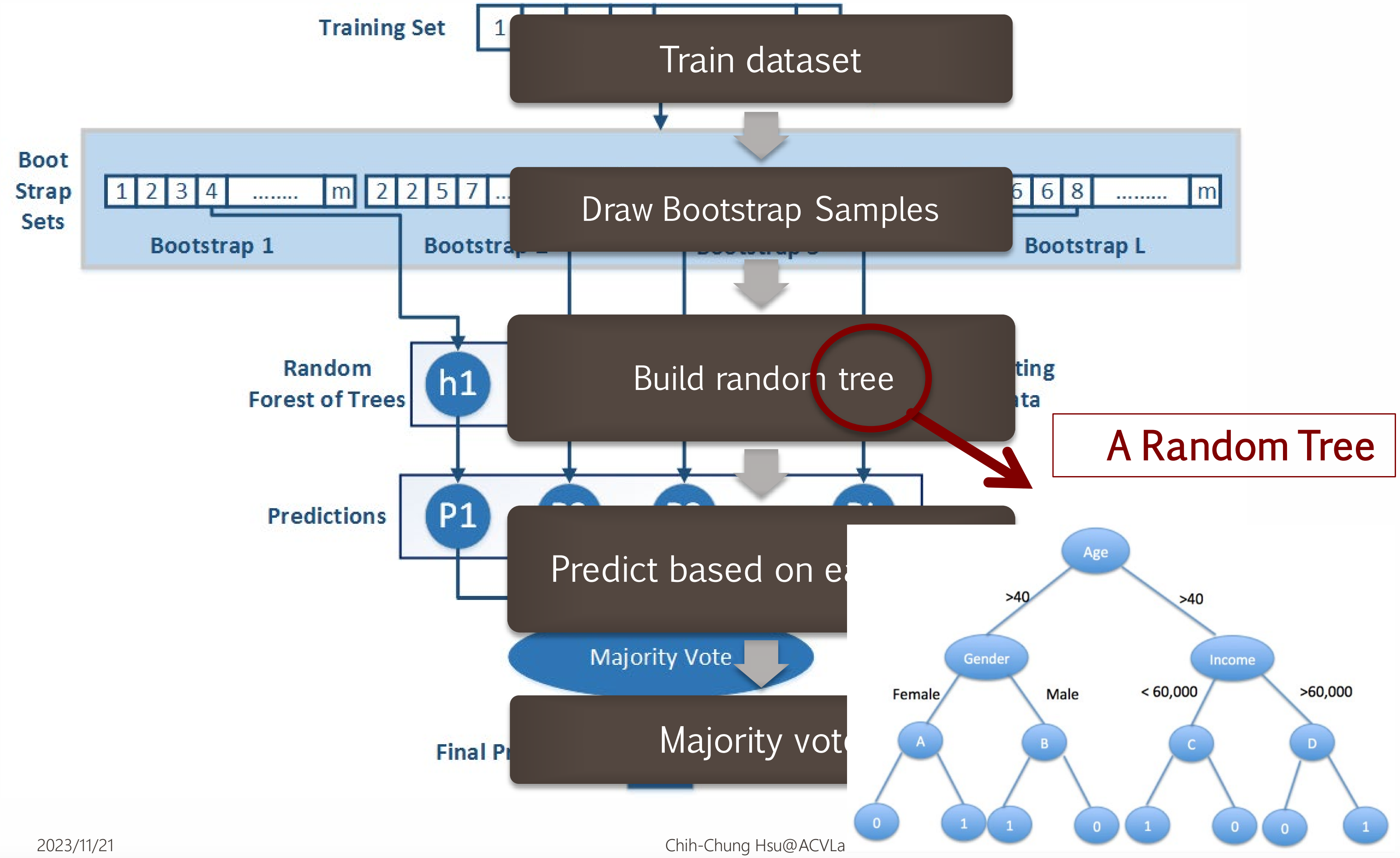
Confusion matrix		Prediction	
		0	1
Truth	0	53921	3159
	1	12450	4853

	Accuracy
Overall	79.2 %
Target = 1	28.1 %
Target = 0	94.5 %

Random Forest

- Machine learning ensemble algorithm
- -- Combining multiple predictors
- Based on tree model
- For both regression and classification
- Automatic variable selection
- Handles missing values
- Robust, improving model stability and accuracy

Random Forest



Train dataset

Draw Bootstrap Samples

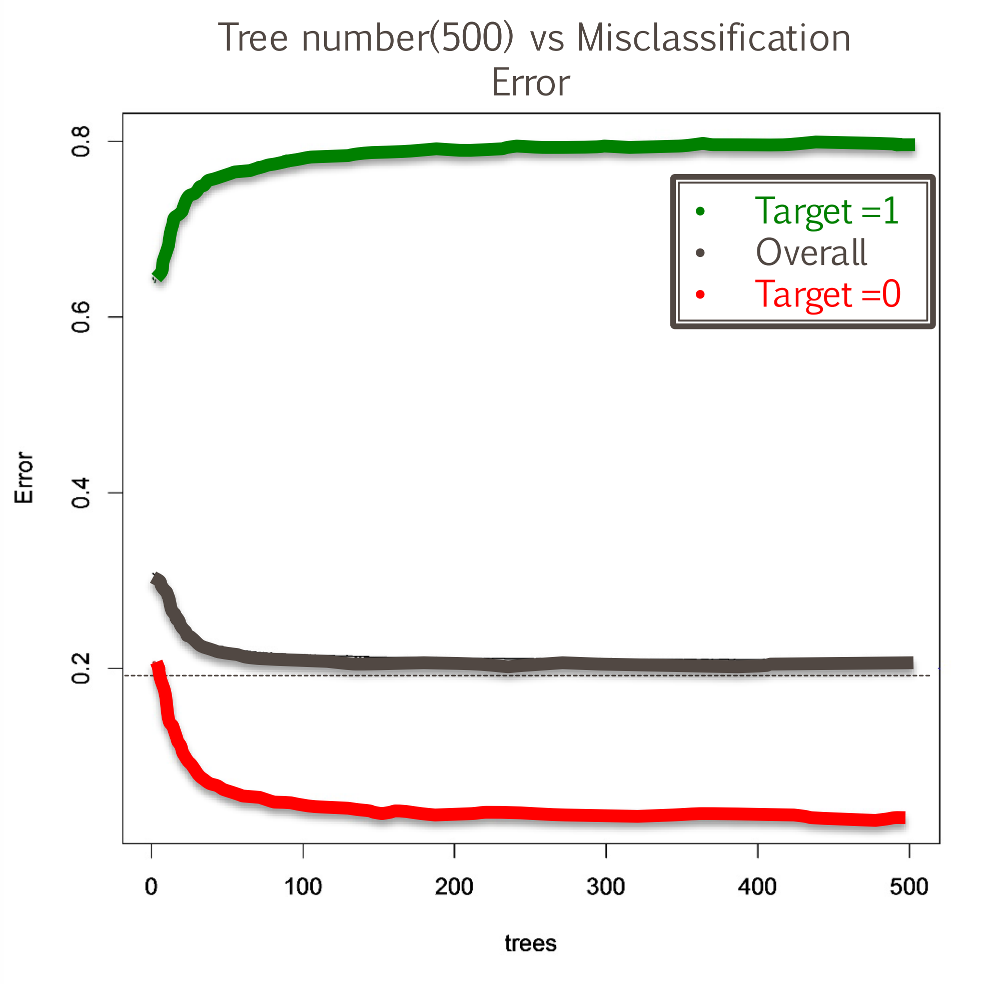
Build random tree

Predict based on e

Majority vot

A Random Tree

Random Forest



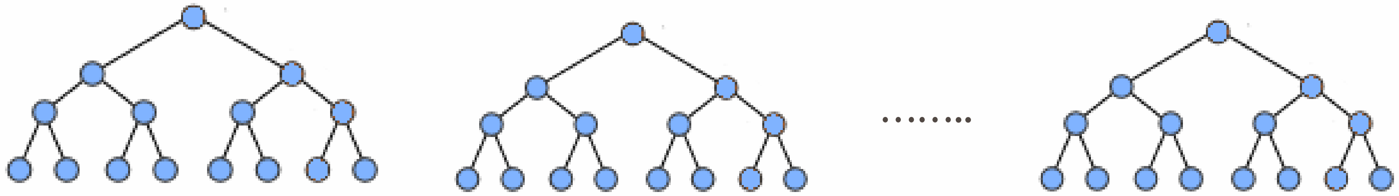
Confusion matrix		Prediction	
		0	1
Truth	0	36157	1181
	1	8850	2223

	Accuracy
Overall	79.3%
Target = 1	20.1%
Target = 0	96.8%

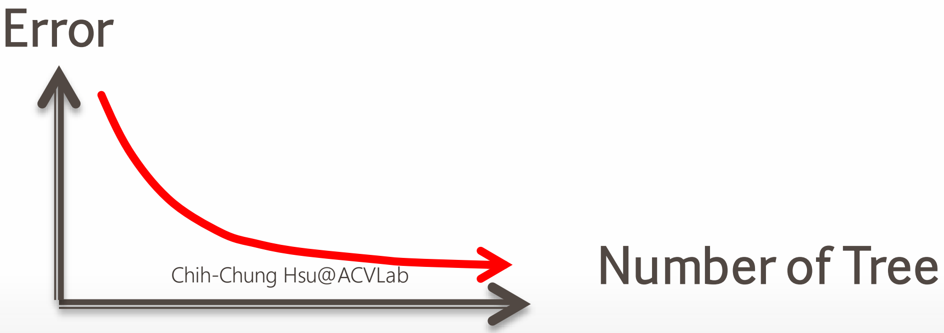
XGBoost

- Additive tree model: add new trees that complement the already-built ones
- Response is the optimal linear combination of all decision trees
- Popular in Kaggle competitions for efficiency and accuracy

Additive tree model

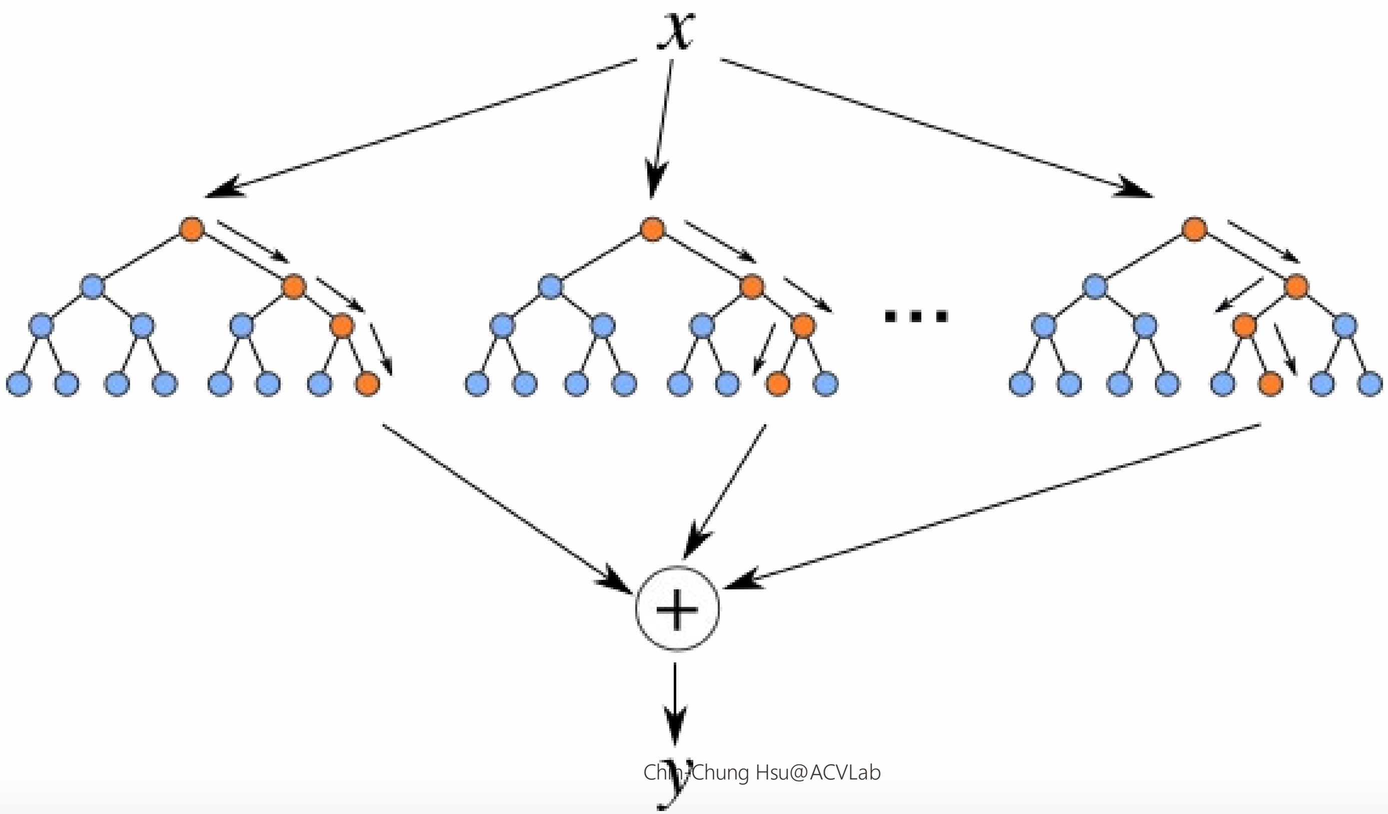


Greedy Algorithm

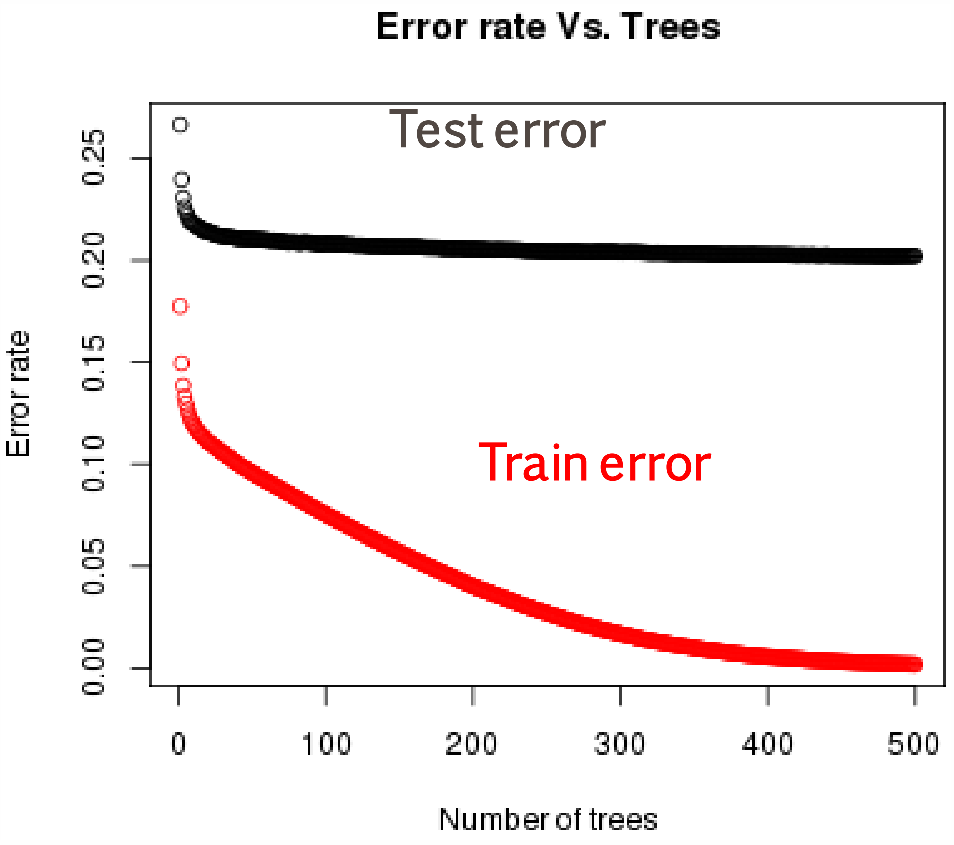


XGBoost

- Additive tree model: add new trees that complement the already-built ones
- Response is the optimal linear combination of all decision trees
- Popular in Kaggle competitions for efficiency and accuracy



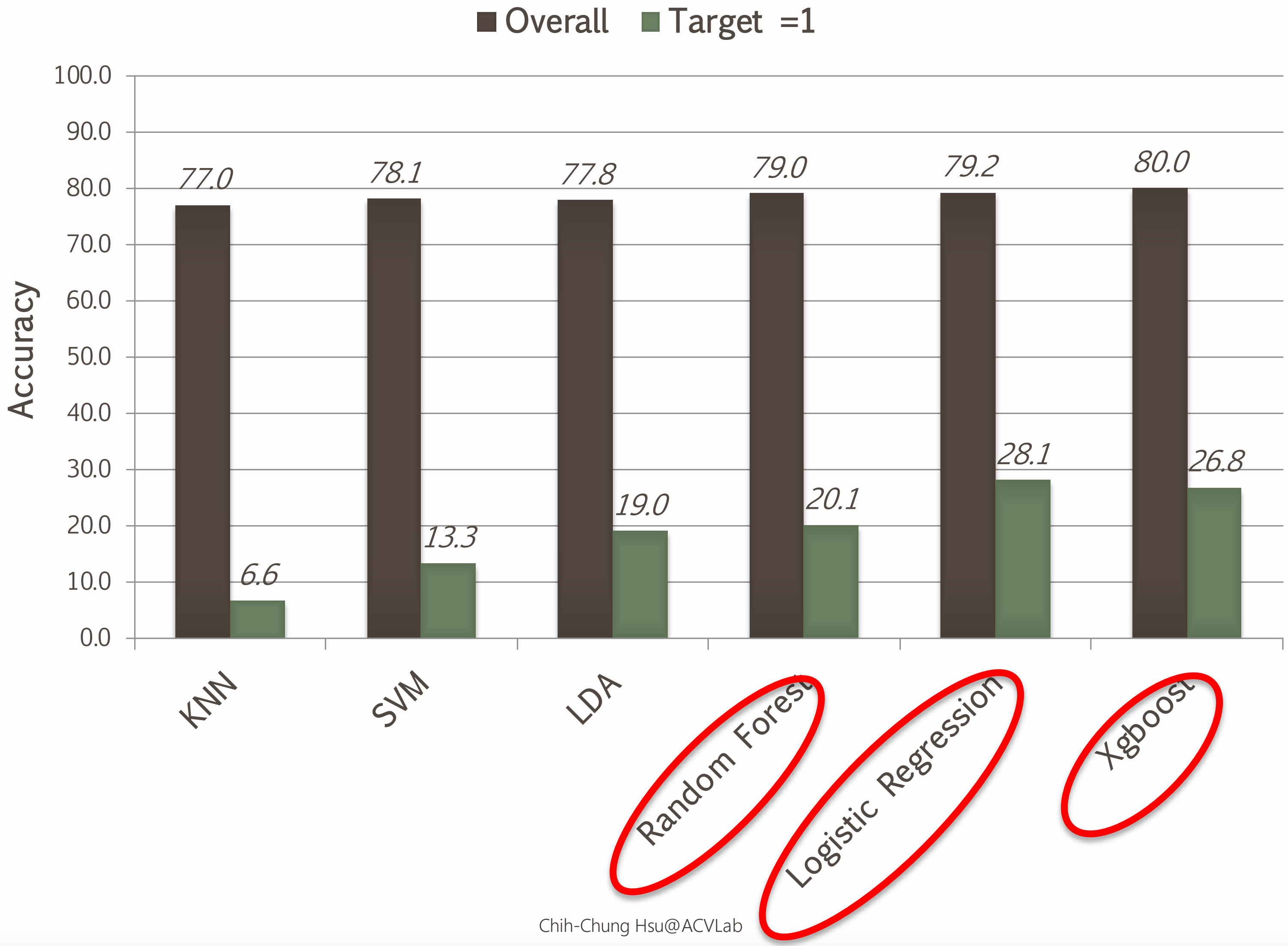
XGBoost



Confusion matrix		Prediction	
		0	1
Truth	0	35744	1467
	1	8201	2999

	Accuracy
Overall	80.0%
Target = 1	26.8%
Target = 0	96.1%

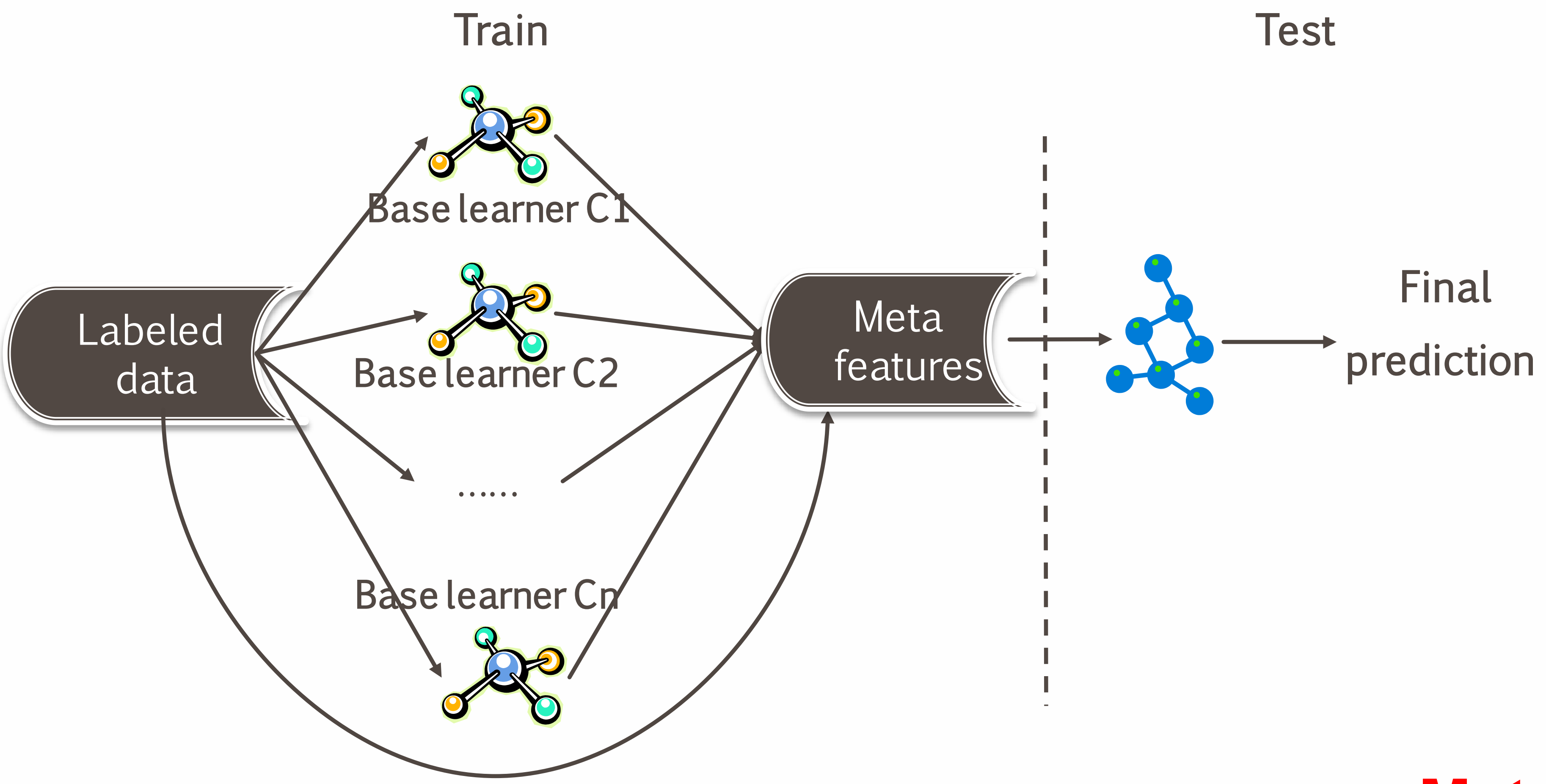
Methods Comparison



Winner or Combination ?

Stacking

- Main Idea: Learn and combine multiple classifiers



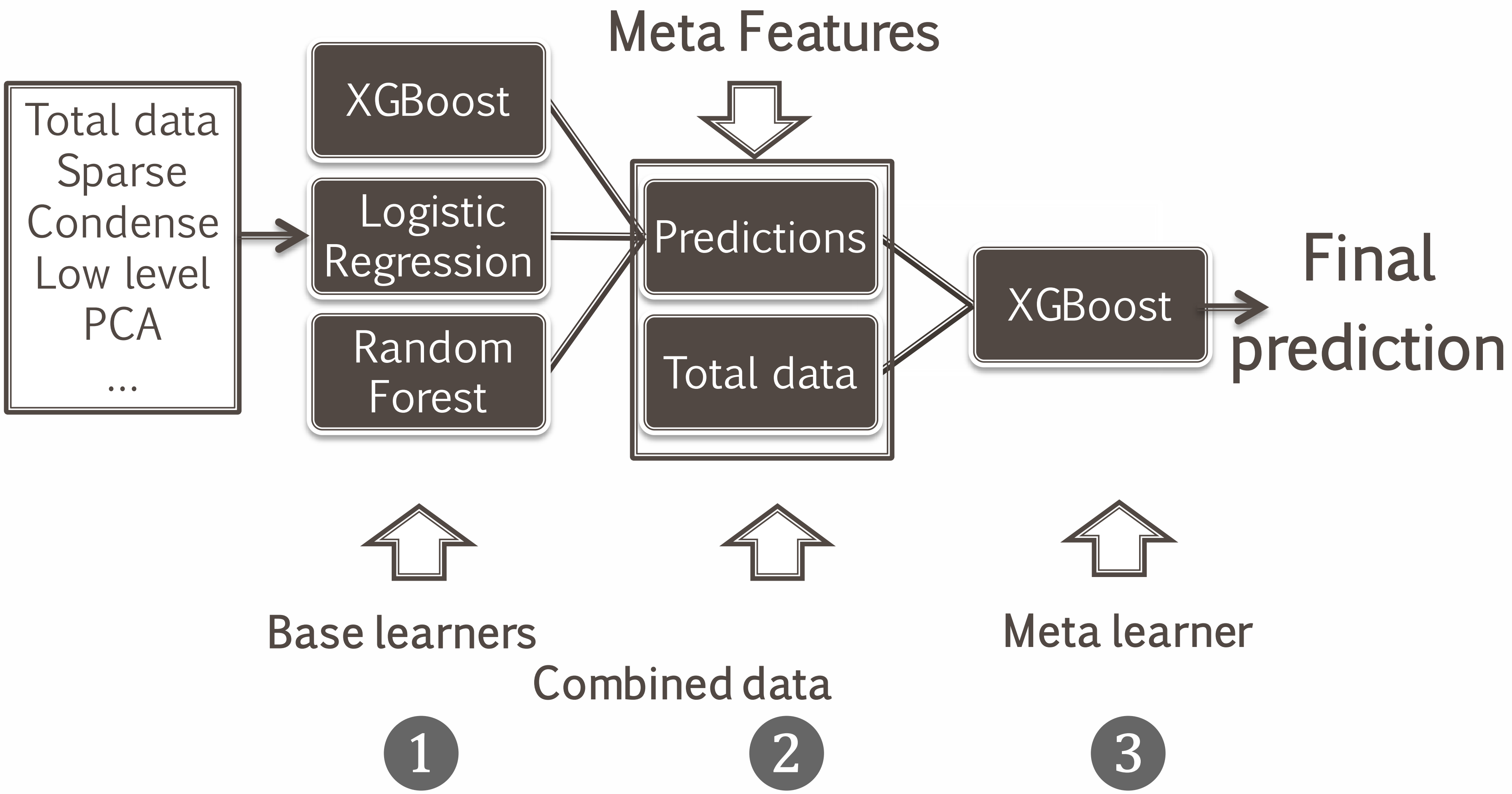
Meta learner

Generating Base and Meta Learners

- Base model—efficiency, accuracy and diversity
  - Sampling training examples
  - Sampling features
  - Using different learning models
-
- Meta learner
  - Majority voting
  - Weighted averaging
  - Kmeans
  - Higher level classifier — Supervised(XGBoost)

Unsupervised

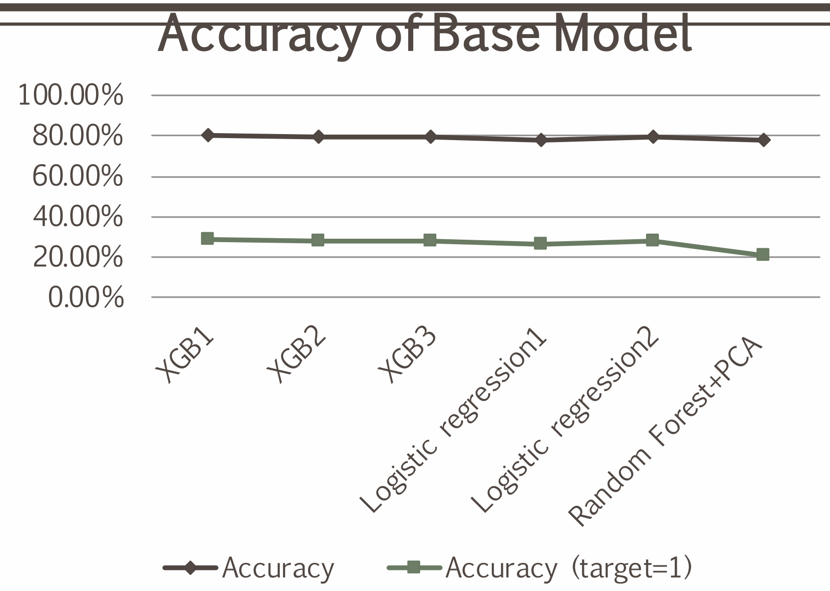
Stacking model

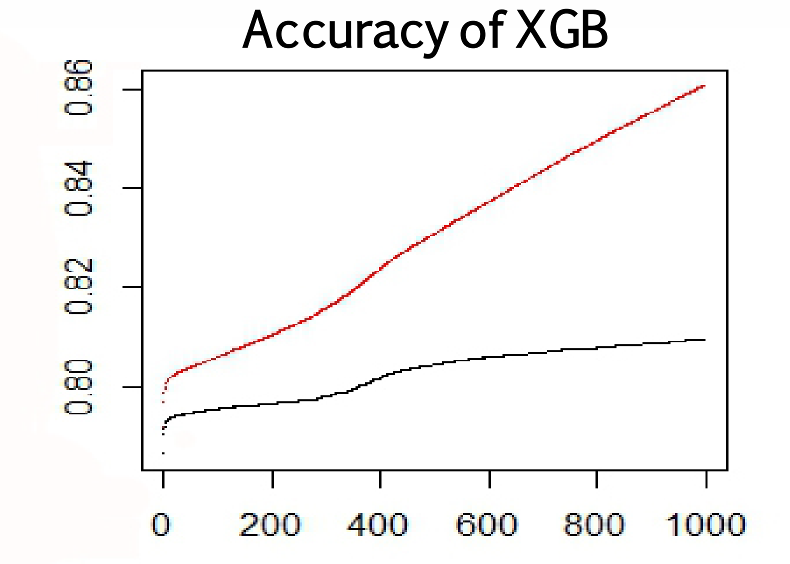


Stacking Results

XGB + total data	80.0%	28.5%
XGB + condense data	79.5%	27.9%
XGB + Low level data	79.5%	27.7%
Logistic regression+ sparse data	78.2%	26.8 %
Logistic regression+ condense data	79.1%	28.1%
Random forest + PCA	77.6%	20.9%

XGB	81.11%	29.21%

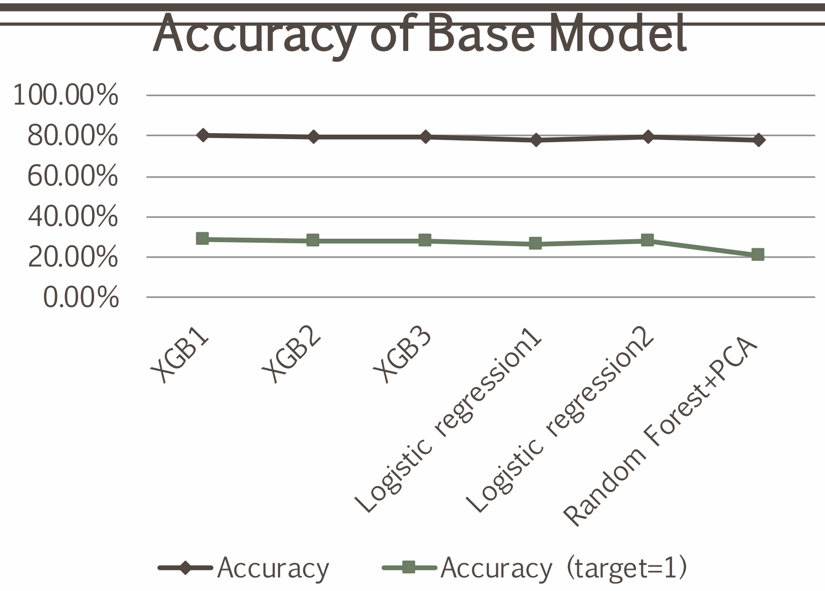


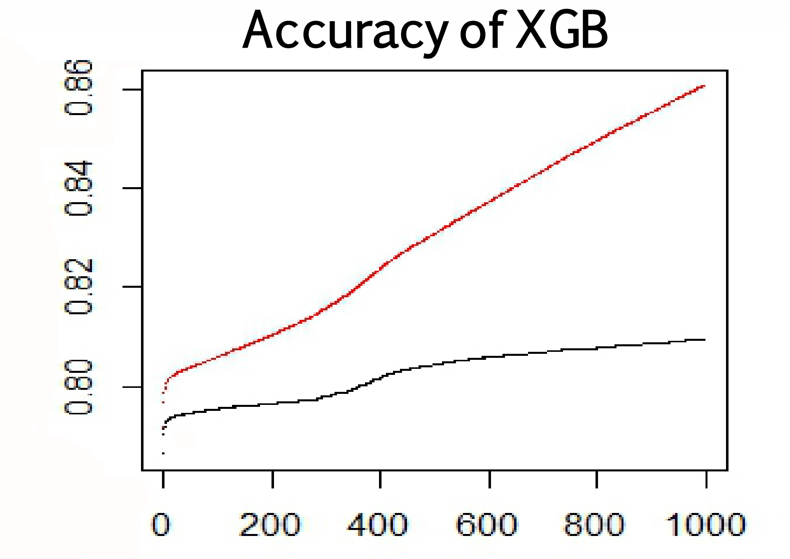


Stacking Results

Base Model	Accuracy	Accuracy (target=1)
XGB + total data	80.0%	28.5%
XGB + condense data	79.5%	27.9%
XGB + Low level data	79.5%	27.7%
Logistic regression+ sparse data	78.2%	26.8 %
Logistic regression+ condense data	79.1%	28.1%
Random forest + PCA	77.6%	20.9%

Meta Model	Accuracy	Accuracy (target=1)
XGB	81.11%	29.21%
Averaging	79.44%	27.31%
Kmeans	77.45%	23.91%


Accuracy of Base Model


Accuracy of XGB

Summary and Conclusion

- Data mining project in the real world
  - Huge and noisy data
- Data preprocessing
  - Feature encoding
  - Different missing value process:
  - New level, Median / Mean, or Random assignment
- Classification techniques
  - Classifiers based on distance are not suitable
  - Classifiers handling mixed type of variables are preferred
  - Categorical variables are dominant
  - Stacking makes further promotion
- Biggest improvement came from model selection, parameter tuning, stacking
- Result comparison : Winner result: 80.4%
- Our result: 79.5%

WHY SO GOOD?

Gradient Boost Decision Tree (GBDT)

- Gradient boosting is an ensemble method widely used for regression, classification and ranking tasks, developed in 2001
- Proposed by Friedman. [1]

- This type of algorithm trains the basic learner of this round by using the negative gradient of the error of the previous round of basic learners as the training target.
- Continuously reduce the deviation of the integrated model on the training set to achieve high-precision integration

[1] Friedman J H. Greedy Function Approximation: A Gradient Boosting Machine[J]. Annals of Statistics, 2001, 29(5):1189-1232.

Intro to GBDT

- The learner based on the Gradient Boosting algorithm is called Gradient Boosting Machine (GBM).
  - The class method was originally proposed in the regression problem, and later extended to the field of classification. If AdaBoost is Boosting
  - The pioneering work of the method, then GBM is the master of the Boosting method.

- GBM has almost refreshed the accuracy records of many datasets in various fields. Some people think that GBM is the best performing
  - Machine learning algorithms, this statement is a bit radical, but usually the strategies of the winners of various data competitions are also
  - There will indeed be methods or ideas for such algorithms

CART (Classification and Regression tree)

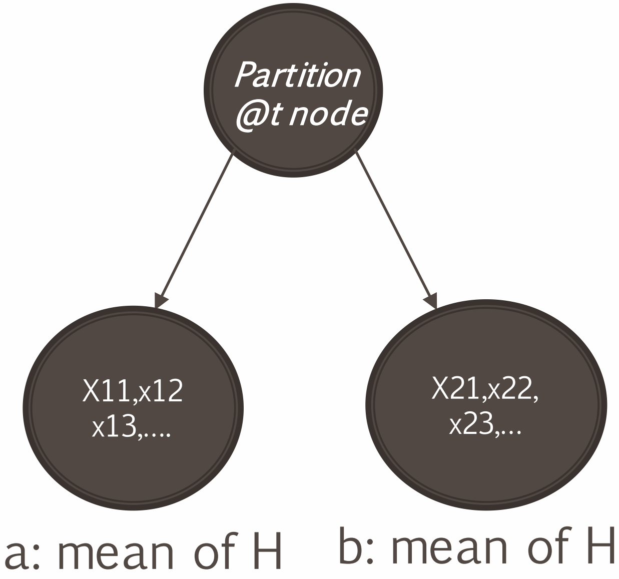
- The full name of the Cart tree is the classification and regression tree, which is a binary tree that can handle both classification and regression problems.
  - The strategies for dealing with the two types of problems are different, but the framework is generally the same. [2]
- The generation framework of regression tree and decision tree is the same, the only difference is the attribute selection criteria of nodes.

- The selection criterion used by the **classification tree** is the Gini coefficient, and it is considered that the attribute with a small Gini coefficient is better.
- The **regression tree** uses least squares error to calculate the division error of each attribute, and attributes with small division errors are better.

[2] Breiman, L.,J.Friedman, C.J.Stone, and R. A. Olshen (1984). Classification and regression Trees. Chapman & Hall/CRC, Boca Raton, FL.

Least-squared Solution for CART

- The regression tree will get a predicted value at each division node. Taking the label as height (H) of a person as an example, this predicted value
  - It is the average of the heights of all samples on this node.
- When branching, traverse all attributes for binary division, and select the division attribute that minimizes the squared error as this
  - Partitioning properties of nodes



The formula for calculating the squared error of the t-th attribute

$$\text{loss\_t} = \sum_{\mathrm{i}} (x_{1i} - a)^2 + \sum_{j} (x_{2\mathrm{j}} - b)^2$$

Traverse each features to find the attribute that minimizes loss, and let it be the current node properties of

[2] Breiman, L.,J.Friedman, C.J.Stone, and R. A. Olshen (1984). Classification and regression Trees. Chapman & Hall/CRC, Boca Raton, FL.

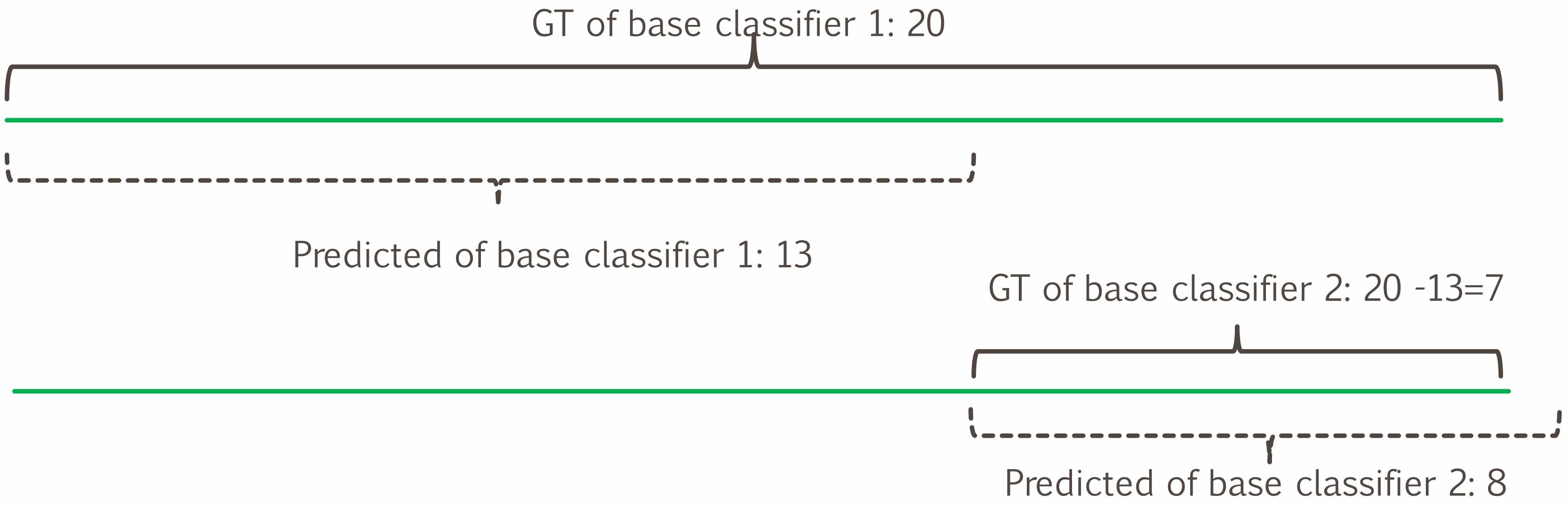
Least-squared Solution for CART

- If the attribute has multiple possible values, you need to traverse all possible segmentation points to find the loss
  - The smallest split point is used as the binary split point of the attribute

- (Final) Continue to select new optimal features to divide the subset until there is only one sample per leaf node
  - Continue to select new optimal features to divide the subset until each leaf node has only one sample (this situation is rare or difficult to achieve, usually a parameter that allows leaf nodes to retain the minimum number of samples is set, If the condition is met, it can be stopped. If the leaf node has multiple sample values, the mean value is used as the predicted value of the leaf node)

Boosting Tree

- Using the error of the previous round of basic learners as the training target to train the basic learners of this round, and continuously reduces the deviation of the integrated model on the training set to achieve high-precision integration
  - For example: For regression problem, if our predicted label value is 20



GT of base classifier 1: 20
Predicted of base classifier 1: 13
GT of base classifier 2: 20 -13=7
Predicted of base classifier 2: 8

  - The predicted label for the third time is 20-13-8 = -1, and this process is repeated until the specified number of base classifiers are generated
  - Finally, adding all the prediction results is the prediction result of the integrated model

Additive Model and Forward-Step Algorithm

- Most boosting algorithms use additive (weighting-sum) for assembling weak classifiers

$$H(x) = \sum_{\mathrm{m=1}}^{M} \beta_m * h_m\,(x; a_m)$$

$h_m\ is\ m-th\ classifier$，$\beta_m$ is m-th coefficient， a_m is m classifiers' parameters, M is the number of classifiers.

Let $\beta_m = 1$ we have

$$H(x) = \sum_{\mathrm{m=1}}^{M} h_m(x; a_m)$$

[3] Friedman J, Hastie T, Tibshirani R. Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors)[J]. Annals of Statistics, 2000, 28(2):337-374.

Additive Model and Forward-Step Algorithm

- It is an optimization issue. Given L and training data, we want to find a parameter

$$min_{\alpha} \sum_{i=1}^{N} L(y_i; \sum_{\mathrm{m}=1}^{M} h_m(x_i; a_m))$$

- Directing solving this optimization problem is very hard.
- We use a forward-step algorithm (a type of greedy algorithm)
    - Rewriting the additive model to be

$$H_m(x) = H_{m-1}(x) + h_m(x; a_m))$$

Additive Model and Forward-Step Algorithm

- Only optimizing the loss at m-step (like regression tree)

$$min_{\alpha_m} \sum_{i=1}^{N} L(y_i, H_{m-1}(x_i) + h_m(x_i; a_m))$$

- With squared loss, we have $L(y, H_m) = (y - H_m)^2$ and
  - $L(y, H_{m-1}(x) + h_m(x; a_m))$
  - $= [y - H_{m-1}(x) - h_m(x; a_m)]^2$
  - $= [r - h_m(x; a_m)]^2$

- where r = $y - H_{m-1}(x)$ ，it is just like to the boosting tree

Gradient Boosting

- What's different between gradient boosting and boosting tree?

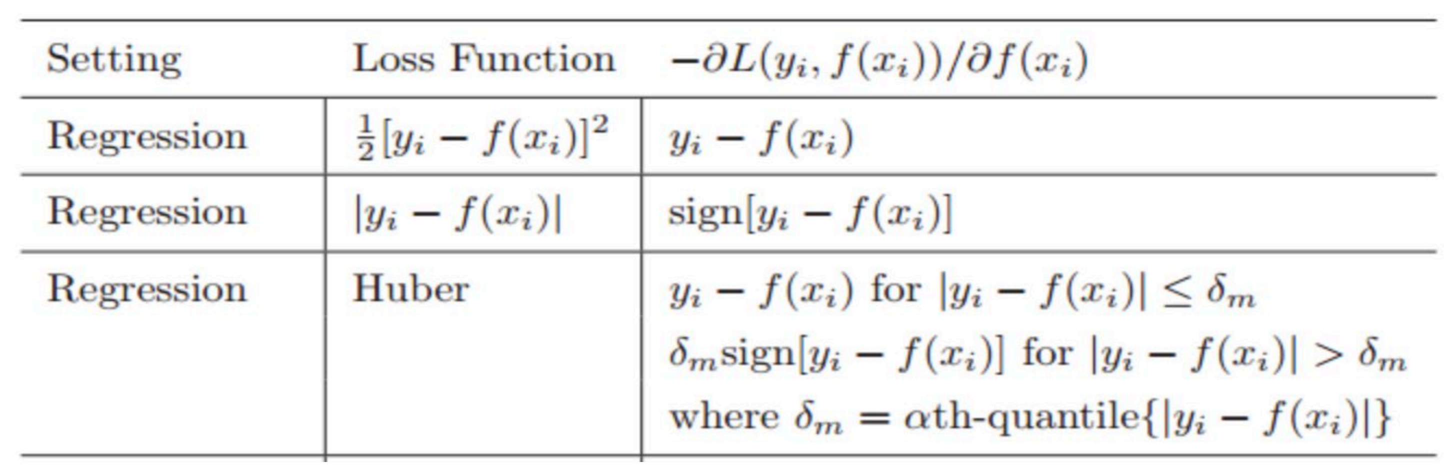
- Boosting Tree :
  - Fit the current weak classifier with the residual for each step.
- GBDT:
  - Fit the current weak classifier with the negative gradient of the residue of the label for each step.

What’s the negative gradient of the residual

- The negative gradient of residual for $i = 1,2,3 \dots, N$ samples is

$$r_{mi} = -\frac{\partial L(y_i, H_{m-1}(x_i))}{H_{m-1}(x_i)}$$

- With the squared loss, we have $L(y, H(x)\ ) = \frac{1}{2}(y - H(x))^2$, the negative gradient of the residual is $\mathrm{y} - H(x)$



Setting	Loss Function	$-\partial L(y_i, f(x_i))/\partial f(x_i)$
Regression	$\frac{1}{2}[y_i - f(x_i)]^2$	$y_i - f(x_i)$
Regression	$\|y_i - f(x_i)\|$	$\mathrm{sign}[y_i - f(x_i)]$
Regression	Huber	$y_i - f(x_i)$ for $\|y_i - f(x_i)\| \le \delta_m$ $\delta_m \mathrm{sign}[y_i - f(x_i)]$ for $\|y_i - f(x_i)\| > \delta_m$ where $\delta_m = \alpha$th-quantile$\{\|y_i - f(x_i)\|\}$

Extreme Gradient GBDT (XGBoost)

- XGBoost: Fast and accurate GBDT
  - Add the regularization term of the computational complexity in the loss function

GBDT : $Obj = \sum_{i=1}^{n} L(y_i, H_M(x_i))$

XGBoost: $Obj = \sum_{i=1}^{n} L\big(y_i, H_M(x_i)\big) + \sum_{m=1}^{M} \Omega(h_{\mathrm{m}})$

where M is the number of the weak classifers，$\Omega(h_m)$ is the complexity of m-th tree

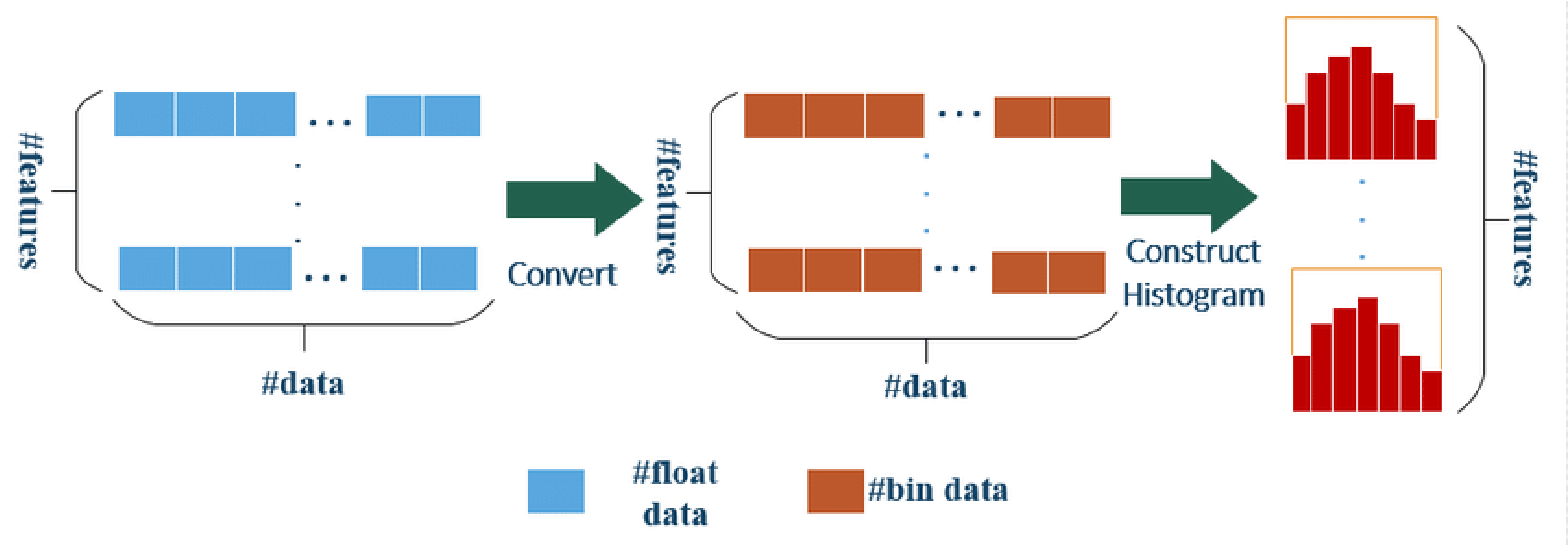
[4] Chen T, Guestrin C. XGBoost:A Scalable Tree Boosting System[C]// ACM, 2016:785-794.

LIGHTGBM

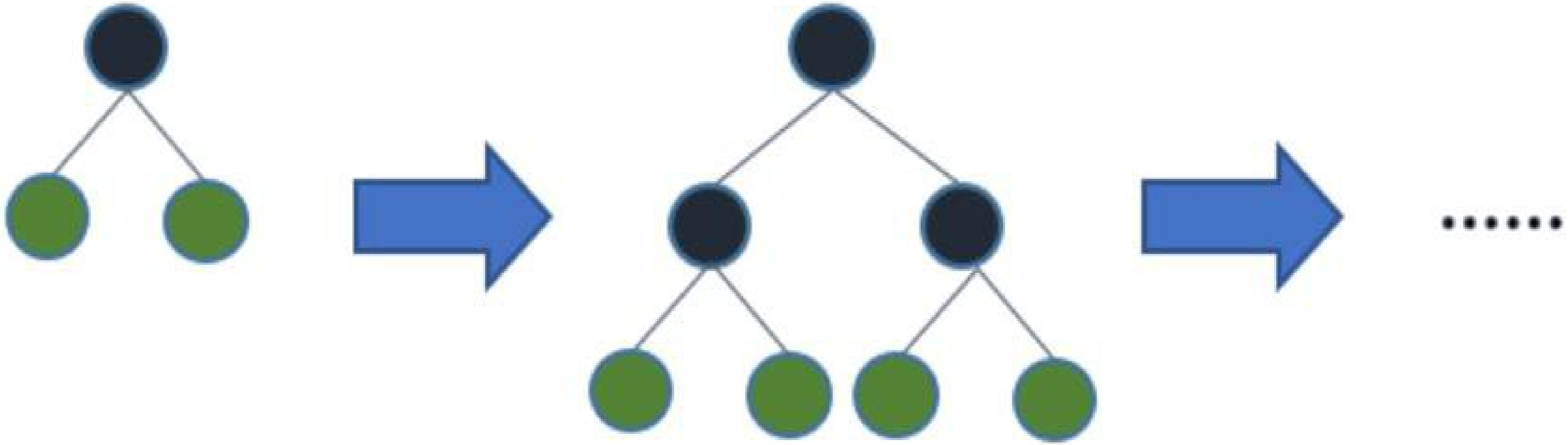
A Highly Efficient Gradient Boosting Decision Tree

Histogram-based Algorithm

- Discretize the feature…
    - Moving averaging

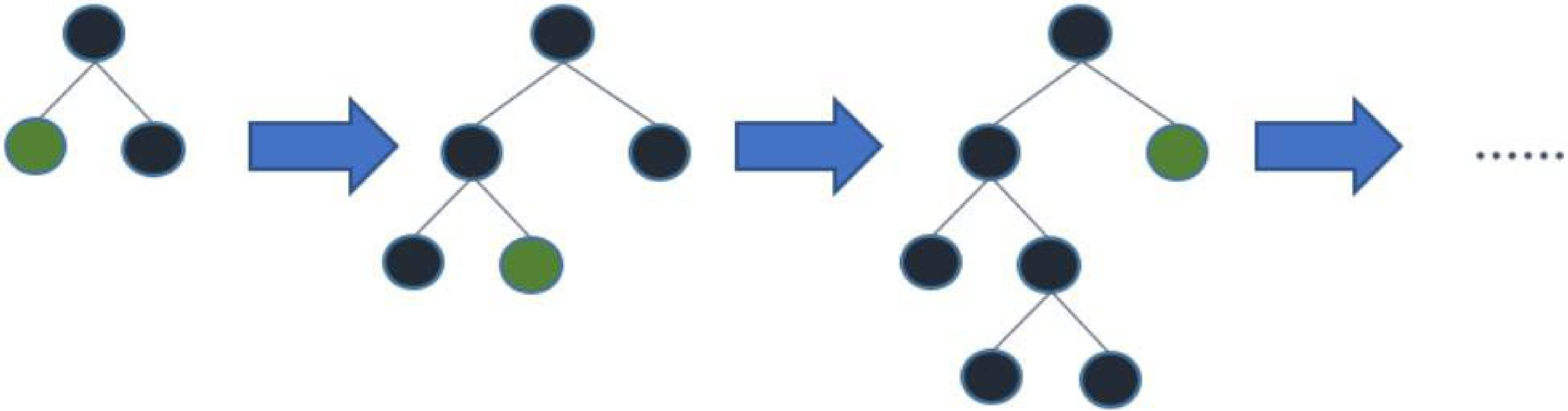


Growing Mechanisms of Decision Tree



Level-wise tree growth

Growing Mechanisms of Decision Tree



LightGBM → Leaf-wise tree growth

An Advantages of Leaf-wise Tree Growth

- Leaf-wise Tree Growth (Best-First)
- Choose the leaf with Max $\Delta\mathcal{L}$ (loss) to grow
- → Holding # of leaves fixed, tend to achieve lower loss

- May cause over-fitting when # of data is small
- → includes the max_depth parameter to limit tree depth

A Problem of Gradient Boosting Decision Tree

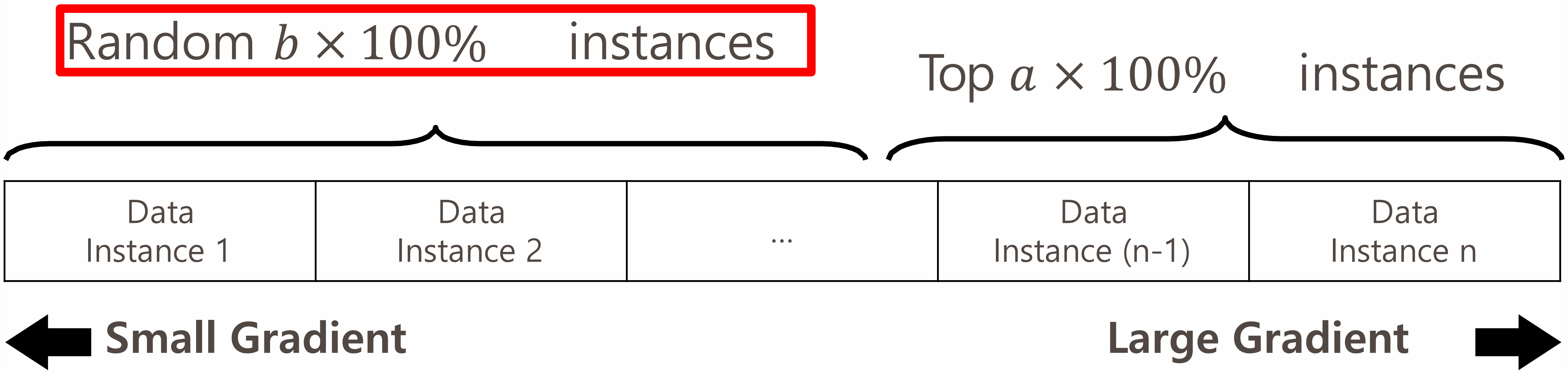
- GBDT
- Gradient for each data instance in GBDT provides the importance of data instance.

- An instance with Large Gradient → Important
- An instance with Small Gradient → Discard (Problem)

New Novel Techniques of LightGBM

- GOSS (Gradient-based One-Side Sampling)

Amplified by Multiplying a Constant $\frac{1-a}{b}$



Random $b \times 100\%$ instances

Top $a \times 100\%$ instances

Data Instance 1	Data Instance 2	…	Data Instance (n-1)	Data Instance n

Small Gradient

Large Gradient

New Novel Techniques of LightGBM

EFB (Exclusive Feature Bundling)

- - High-dimensional data are usually very spare
  - → Many features are mutually exclusive
  - → Bundling exclusive features into a single feature safely
    - → Speed up the training of GBDT without hurting the accuracy

CATBOOST

Designed for categorical data…

CatBoost advantages

- Good quality with default parameters
- Good GPU training speed
- Sophisticated categorical, text and embedding features support
- Model analysis tools
- Set of tools to make GBDT usage easier

https://colab.research.google.com/github/catboost/tutorials/blob/master/events/2020_odsc_west/text_features.ipynb

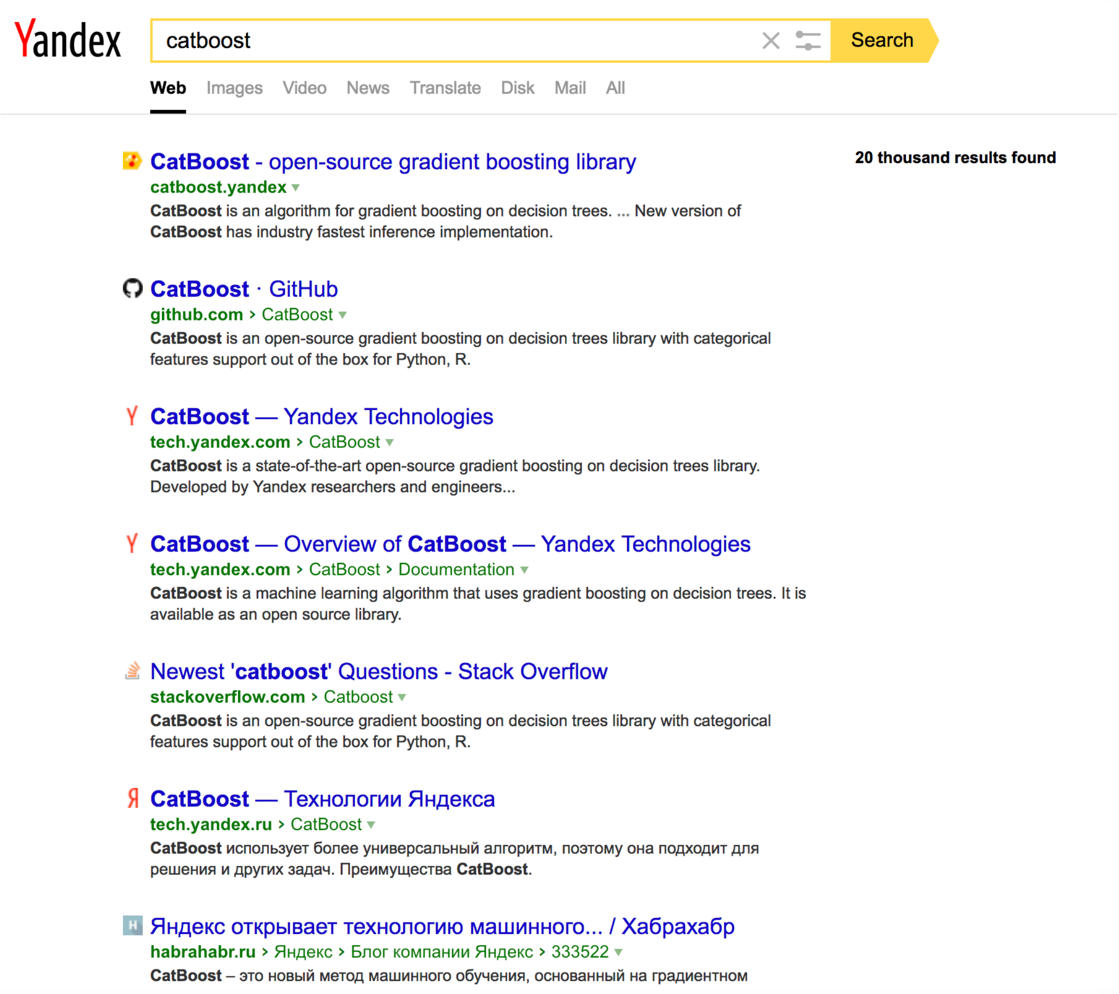
CatBoost in the Wild

- Recommendations at Netflix
- Hotel ranking in Aviasales
- Protection against bots in CloudFlare
- Particle classification in CERN
- Medical research at University of NSW Sydney
- Destination prediction in Careem taxi service
- ML competitions on Kaggle
- Join our Awesome Users List! ☺

https://colab.research.google.com/github/catboost/tutorials/blob/master/events/2020_odsc_west/text_features.ipynb

Yandex.Search

- Task?
  - Search document order prediction
- Task type: ranking
- Dataset features:
  - Classic features (PageRank, BM25 and others)
  - Neural Networks output
- CatBoost features used:
  - YetiRankPairwise target
  - Distributed GPU training
  - Model blending
  - Feature importance analysis
  - Ranking analysis



20 thousand results found

CatBoost - open-source gradient boosting library
catboost.yandex
CatBoost is an algorithm for gradient boosting on decision trees. ... New version of **CatBoost** has industry fastest inference implementation.

CatBoost · GitHub
github.com › CatBoost
CatBoost is an open-source gradient boosting on decision trees library with categorical features support out of the box for Python, R.

CatBoost — Yandex Technologies
tech.yandex.com › CatBoost
CatBoost is a state-of-the-art open-source gradient boosting on decision trees library. Developed by Yandex researchers and engineers...

CatBoost — Overview of **CatBoost** — Yandex Technologies
tech.yandex.com › CatBoost › Documentation
CatBoost is a machine learning algorithm that uses gradient boosting on decision trees. It is available as an open source library.

Newest '**catboost**' Questions - Stack Overflow
stackoverflow.com › Catboost
CatBoost is an open-source gradient boosting on decision trees library with categorical features support out of the box for Python, R.

CatBoost — Технологии Яндекса
tech.yandex.ru › CatBoost
CatBoost использует более универсальный алгоритм, поэтому она подходит для решения и других задач. Преимущества **CatBoost**.

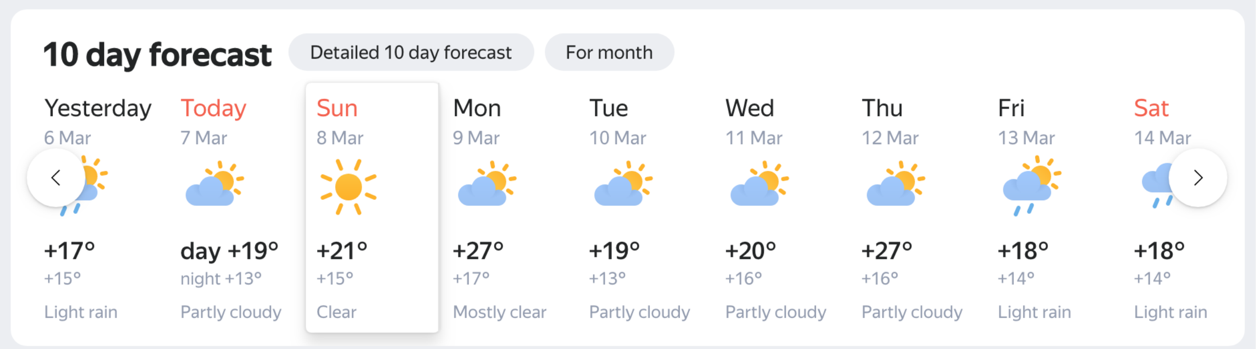
Яндекс открывает технологию машинного... / Хабрахабр
habrahabr.ru › Яндекс › Блог компании Яндекс › 333522
CatBoost – это новый метод машинного обучения, основанный на градиентном

https://colab.research.google.com/github/catboost/tutorials/blob/master/events/2020_odsc_west/text_features.ipynb

Yandex.Weather

- Task?
  - Cloudiness type and temperature prediction
- Task type: multiclassification and regression

- Dataset features
  - Physical weather model output
  - Neural network output
  - Online-data from weather stations
  - Weather historical data
- CatBoost features used:
  - Multiclassification target and RMSE (for temperature)
  - GPU training
  - Feature importance analysis
  - Training process visualization



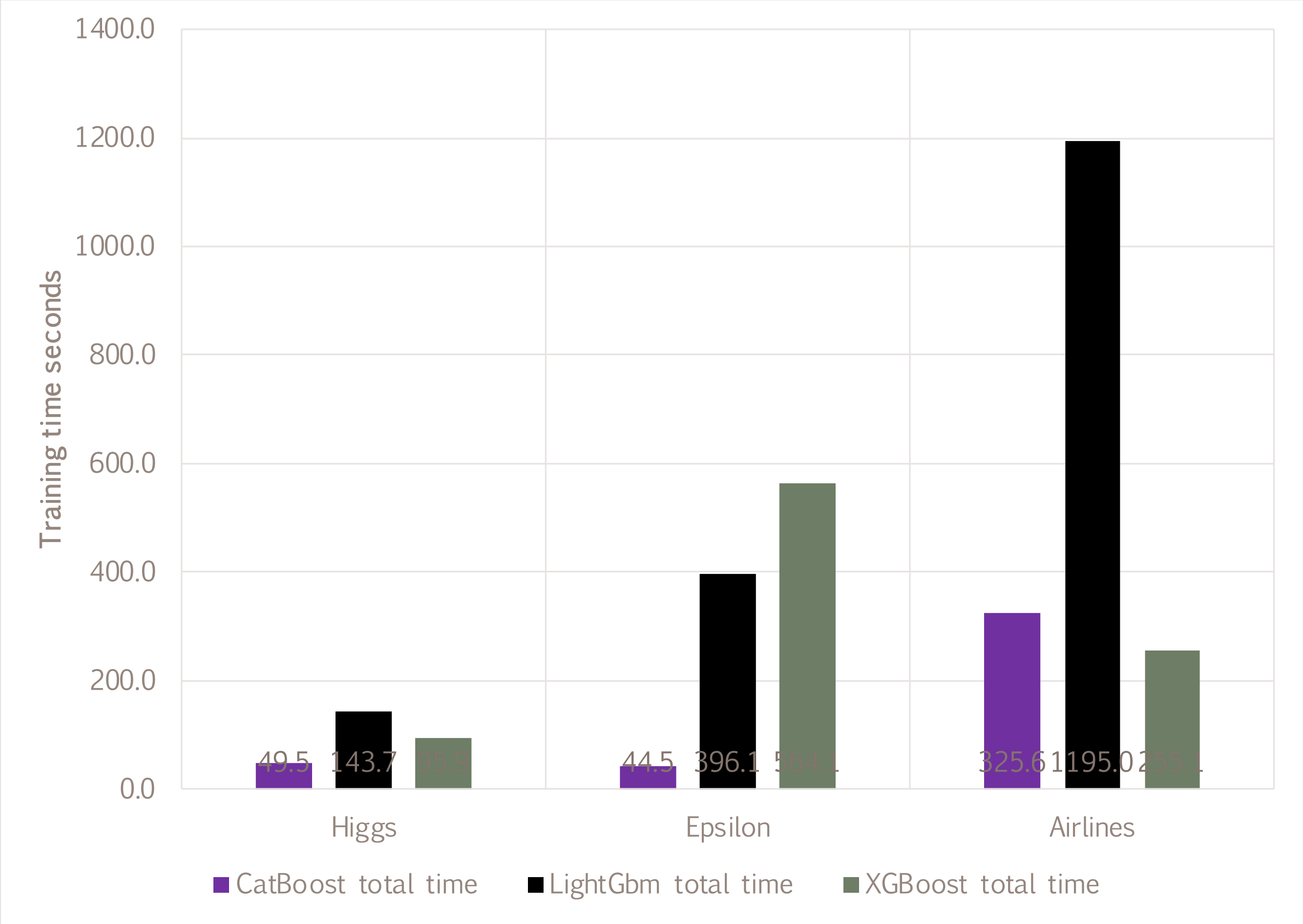
10 day forecast

Yesterday	Today	Sun	Mon	Tue	Wed	Thu	Fri	Sat
6 Mar	7 Mar	8 Mar	9 Mar	10 Mar	11 Mar	12 Mar	13 Mar	14 Mar
+17°	day +19°	+21°	+27°	+19°	+20°	+27°	+18°	+18°
+15°	night +13°	+15°	+17°	+13°	+16°	+16°	+14°	+14°
Light rain	Partly cloudy	Clear	Mostly clear	Partly cloudy	Partly cloudy	Partly cloudy	Light rain	Light rain

https://colab.research.google.com/github/catboost/tutorials/blob/master/events/2020_odsc_west/text_features.ipynb

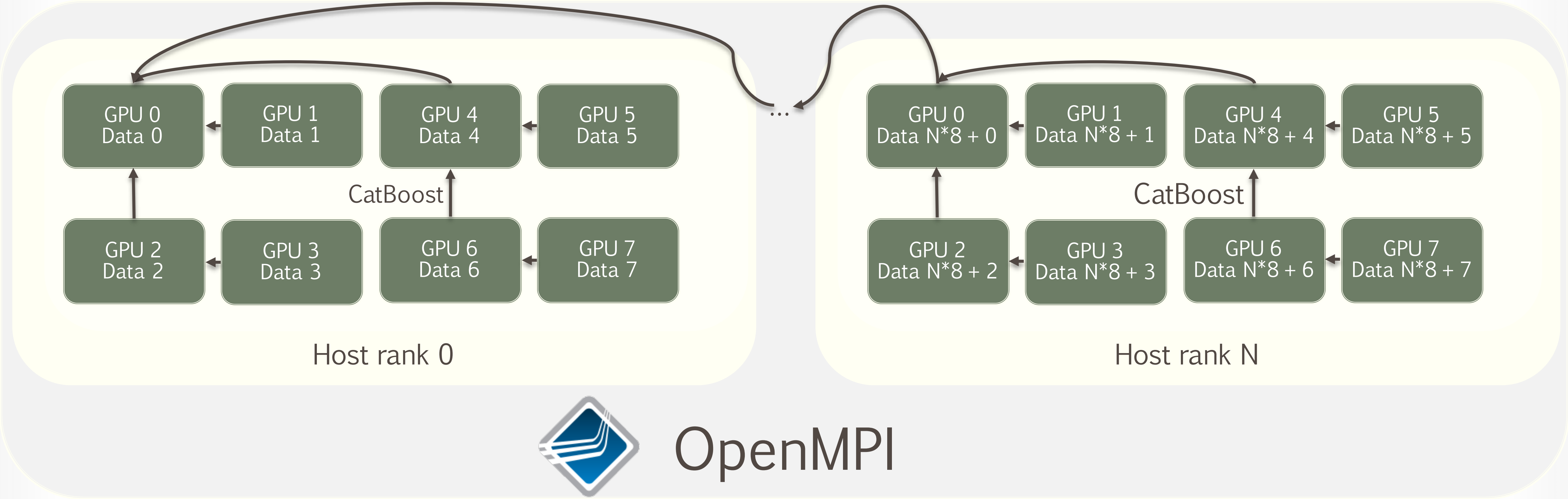
V100 Training performance: 1000 trees

- GPU training time on single NVIDIA Tesla V100
  - Higgs
  - 10.5mln objects
- 28 features
  - Epsilon
- 400k objects
- 2k features
  - Airlines
  - 23mln objects
  - 13 features
  - CatBoost v0.24.1, Lightgbm 2.3.1, XGBoost 1.0.2



https://colab.research.google.com/github/catboost/tutorials/blob/master/events/2020_odsc_west/text_features.ipynb

Distributed training: multi-host multi-GPU



https://colab.research.google.com/github/catboost/tutorials/blob/master/events/2020_odsc_west/text_features.ipynb

CATEGORICAL FEATURES

Categorical Encoding

- One-hot encoding: create new variables as many as the number of categories in the variable
  - When the categories are high (high cardinality), too many variables are created.
- Label encoding: create a variable so that one category is one integer
  - The size of the variable has no meaning, but it is difficult to reflect in the model
- Target encoding: Create a new variable with representative values (average, median…) of target values belonging to the same category
  - Greedy, Holdout, Ordered…

	x1	x2	…	xn	target
1	2	40	…	rock	1
2	3	55	…	indie	0
3	5	34	…	pop	1
4	2	45	…	rock	0
5	4	53	…	rock	0
6	2	48	…	indie	1
7	5	42	…	rock	1
…					

xn_1	xn_2	xn_3
1	0	0
0	1	0
0	0	1
1	0	0
1	0	0
0	1	0
1	0	0

<one-hot>

xn
1
2
3
1
1
2
1

<label>

xn
0.5
0.5
1
0.5
0.5
0.5
0.5

<target>

Target Statistics

Target encoding

$$\hat{x}_k^i \approx \mathbb{E}(y|x^i = x_k^i)$$

1. Greedy TS: The purpose of smoothing for infrequent categories

$$\hat{x}_k^i = \frac{\sum_{j=1}^{n} \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_i + ap}{\sum_{j=1}^{n} \mathbb{1}_{\{x_j^i = x_k^i\}} + a}$$

- *a is the parameter, p is the target mean*
- Target leakage: Information about the target enters the training data, and overfitting may occur.
- Conditional shift: Because the distribution of train/test data is different, overfitting may occur.
  - ➢ Let's say the size of the train dataset is 1000 and the number of user IDs is 1000. If you create a variable using greedy TS, DT for the training dataset can be classified with 100% accuracy as one variable, but it can be a useless variable in reality.
  - ➢ As a way to avoid conditional shift, when calculating the TS of x_k, subset $D_k \subset D \backslash \{x_k\}$

Target Statistics

2. Holdout TS: How to divide the train dataset and the dataset for TS calculation
    2. The number of available data is greatly reduced
3. Leave-one-out TS: How to use $D\_k=D\backslash\{x\_k\}$ as train dataset and $D\_k=D$ as test dataset
    - Target leakage can't get out of trouble
4. Ordered TS: A method to solve the problem of target leakage and conditional shift
    1) offline ➔ online Randomly change the situation to solve the problem
    2) For the train dataset, apply "Artificial time" with random permutation

$$\hat{x}_k^i = \frac{\sum_{j=1}^{n} \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_i + ap}{\sum_{j=1}^{n} \mathbb{1}_{\{x_j^i = x_k^i\}} + a}$$

	x1	x2	…	xn	target
1	2	40	…	rock	1
2	3	55	…	indie	0
3	5	34	…	pop	1
4	2	45	…	rock	0
5	4	53	…	rock	0
6	2	48	…	indie	1
7	5	42	…	rock	1
…					

random permutation

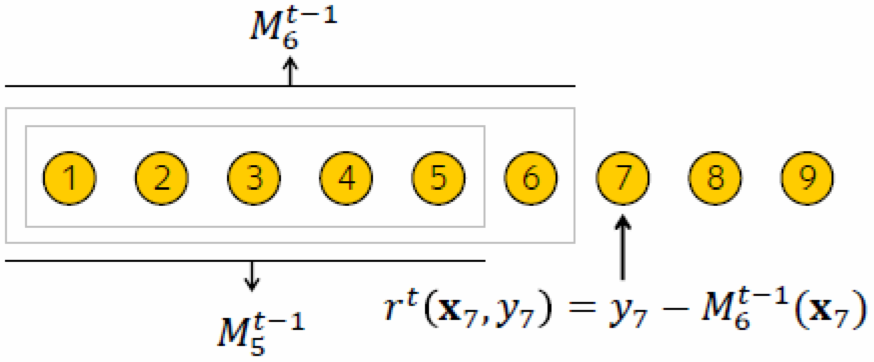
	x1	x2	…	xn	target
1	4	53	…	rock	0
2	3	55	…	indie	0
3	2	40	…	rock	1
4	5	42	…	rock	1
5	5	34	…	pop	1
6	2	48	…	indie	1
7	2	45	…	rock	0
…					

calculating ordered TS
prior = 0.05

	x1	x2	…	xn	target
1	4	53	…	(0+0.05)/(0+1)=0.05	0
2	3	55	…	(0+0.05)/(0+1)=0.05	0
3	2	40	…	(0+0.05)/(1+1)=0.025	1
4	5	42	…	(1+0.05)/(2+1)=0.35	1
5	5	34	…	(0+0.05)/(0+1)=0.05	1
6	2	48	…	(0+0.05)/(1+1)=0.025	1
7	2	45	…	(2+0.05)/(3+1)=0.5125	0
…					

Ordered Boosting

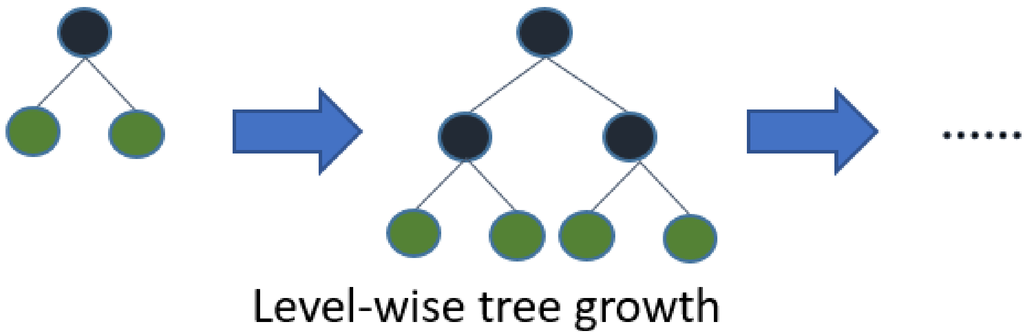
GBDT learning using "Artificial time" concept to solve target leakage/prediction shift problem



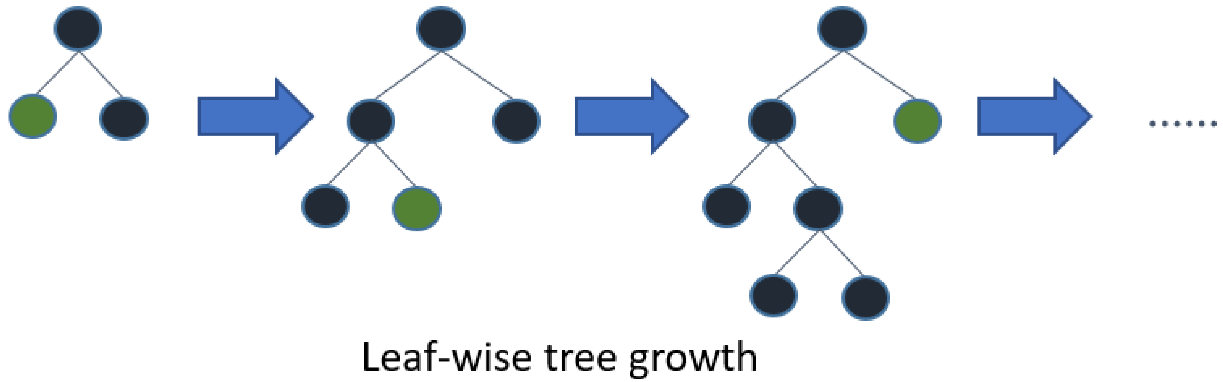
$$r^t(\mathbf{x}_7, y_7) = y_7 - M_6^{t-1}(\mathbf{x}_7)$$

1. Perform random permutation s times (within 10 times in the experiment) to create multiple timelines
2. Gradient boosting is implemented.
   1) to find the gradient
      ① Choose one of the random permutations and,
      ② For each object, the gradient is computed only for objects that exist in the time period preceding itself.
   2) Using cosine similarity, choose a branching criterion with a negative gradient!

XGBoost, LightGBM, CatBoost



LightGBM, CatBoost



XGBoost

- CatBoost is not sensitive to parameters compared to LightGBM and XGBoost.
- However, in the case of a sparse matrix, learning is difficult, and when there are many numeric variables, it is slower than LightGBM.

Experiments

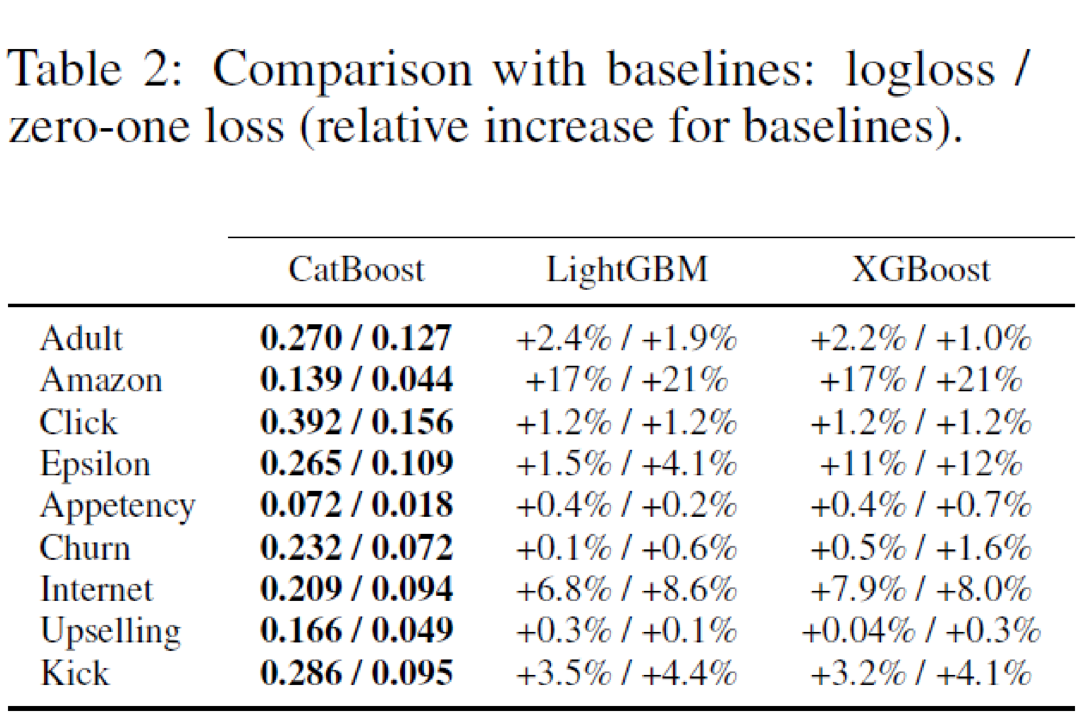


Table 2: Comparison with baselines: logloss / zero-one loss (relative increase for baselines).

	CatBoost	LightGBM	XGBoost
Adult	**0.270 / 0.127**	+2.4% / +1.9%	+2.2% / +1.0%
Amazon	**0.139 / 0.044**	+17% / +21%	+17% / +21%
Click	**0.392 / 0.156**	+1.2% / +1.2%	+1.2% / +1.2%
Epsilon	**0.265 / 0.109**	+1.5% / +4.1%	+11% / +12%
Appetency	**0.072 / 0.018**	+0.4% / +0.2%	+0.4% / +0.7%
Churn	**0.232 / 0.072**	+0.1% / +0.6%	+0.5% / +1.6%
Internet	**0.209 / 0.094**	+6.8% / +8.6%	+7.9% / +8.0%
Upselling	**0.166 / 0.049**	+0.3% / +0.1%	+0.04% / +0.3%
Kick	**0.286 / 0.095**	+3.5% / +4.4%	+3.2% / +4.1%

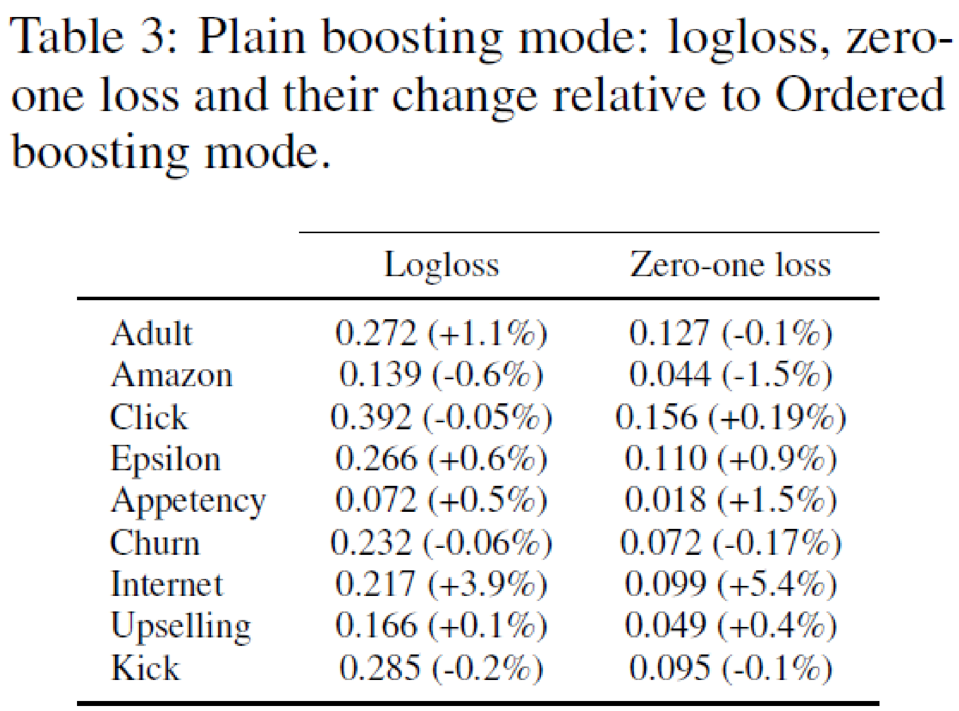


Table 3: Plain boosting mode: logloss, zero-one loss and their change relative to Ordered boosting mode.

	Logloss	Zero-one loss
Adult	0.272 (+1.1%)	0.127 (-0.1%)
Amazon	0.139 (-0.6%)	0.044 (-1.5%)
Click	0.392 (-0.05%)	0.156 (+0.19%)
Epsilon	0.266 (+0.6%)	0.110 (+0.9%)
Appetency	0.072 (+0.5%)	0.018 (+1.5%)
Churn	0.232 (-0.06%)	0.072 (-0.17%)
Internet	0.217 (+3.9%)	0.099 (+5.4%)
Upselling	0.166 (+0.1%)	0.049 (+0.4%)
Kick	0.285 (-0.2%)	0.095 (-0.1%)

Experiments

Table 4: Comparison of target statistics, relative change in logloss / zero-one loss compared to ordered TS.

	Greedy	Holdout	Leave-one-out
Adult	+1.1% / +0.8%	+2.1% / +2.0%	+5.5% / +3.7%
Amazon	+40% / +32%	+8.3% / +8.3%	+4.5% / +5.6%
Click	+13% / +6.7%	+1.5% / +0.5%	+2.7% / +0.9%
Appetency	+24% / +0.7%	+1.6% / -0.5%	+8.5% / +0.7%
Churn	+12% / +2.1%	+0.9% / +1.3%	+1.6% / +1.8%
Internet	+33% / +22%	+2.6% / +1.8%	+27% / +19%
Upselling	+57% / +50%	+1.6% / +0.9%	+3.9% / +2.9%
Kick	+22% / +28%	+1.3% / +0.32%	+3.7% / +3.3%

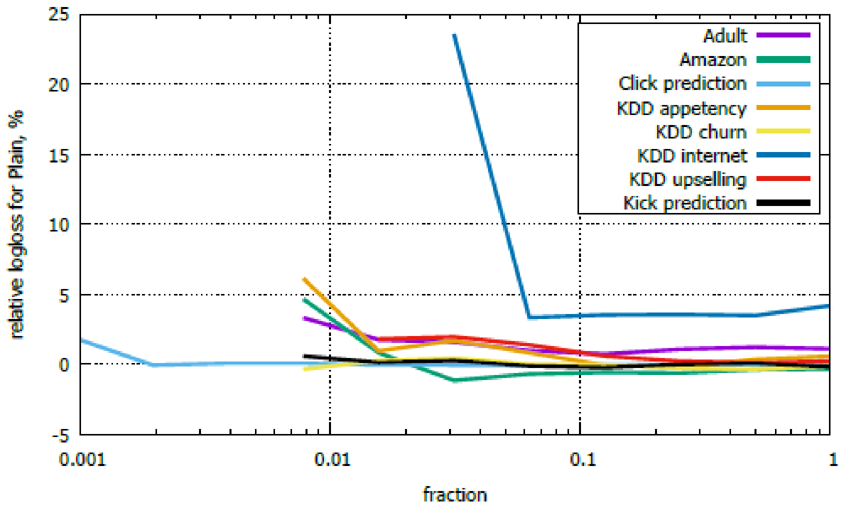


Figure 2: Relative error of Plain boosting mode compared to Ordered boosting mode depending on the fraction of the dataset.