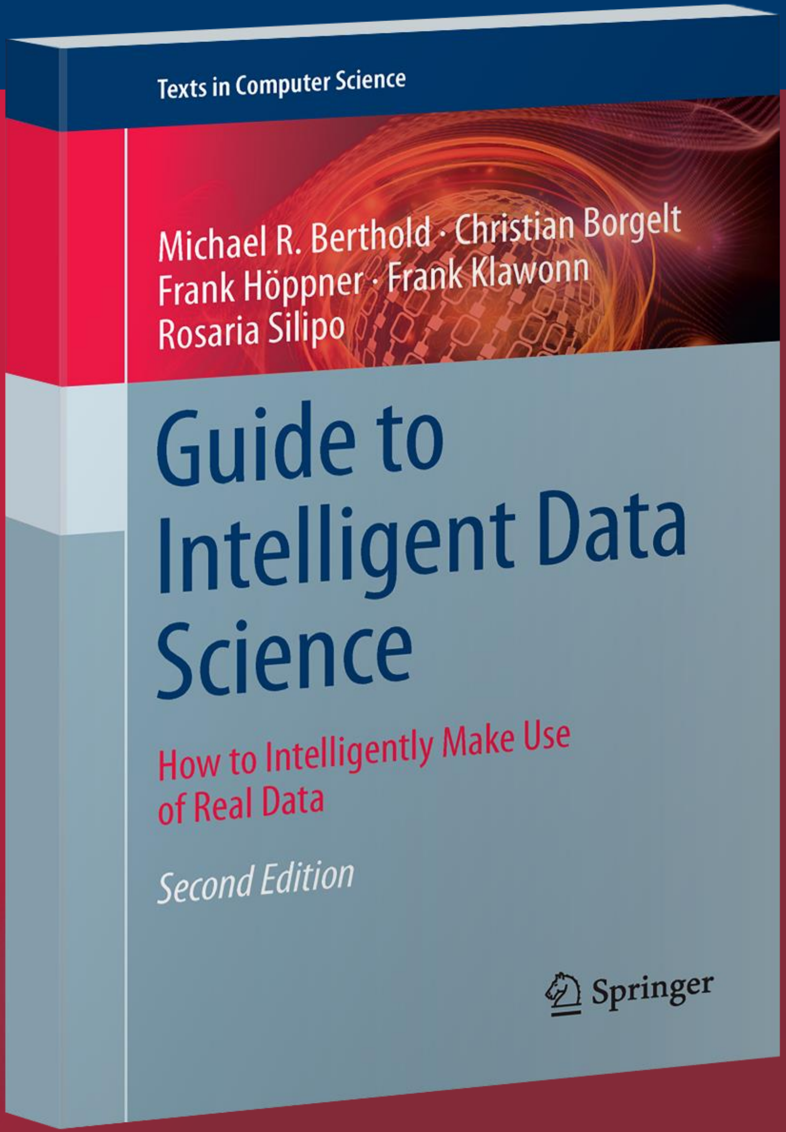
Ensemble Learning



Texts in Computer Science

Michael R. Berthold · Christian Borgelt
Frank Höppner · Frank Klawonn
Rosaria Silipo

Guide to Intelligent Data Science

How to Intelligently Make Use of Real Data

Second Edition

Springer

Summary of this lesson

“If I do not believe the news in todays paper, I buy 100 copies of the paper. Then I believe.”
-Ludwig Wittgenstein

How can we learn from multiple models together?

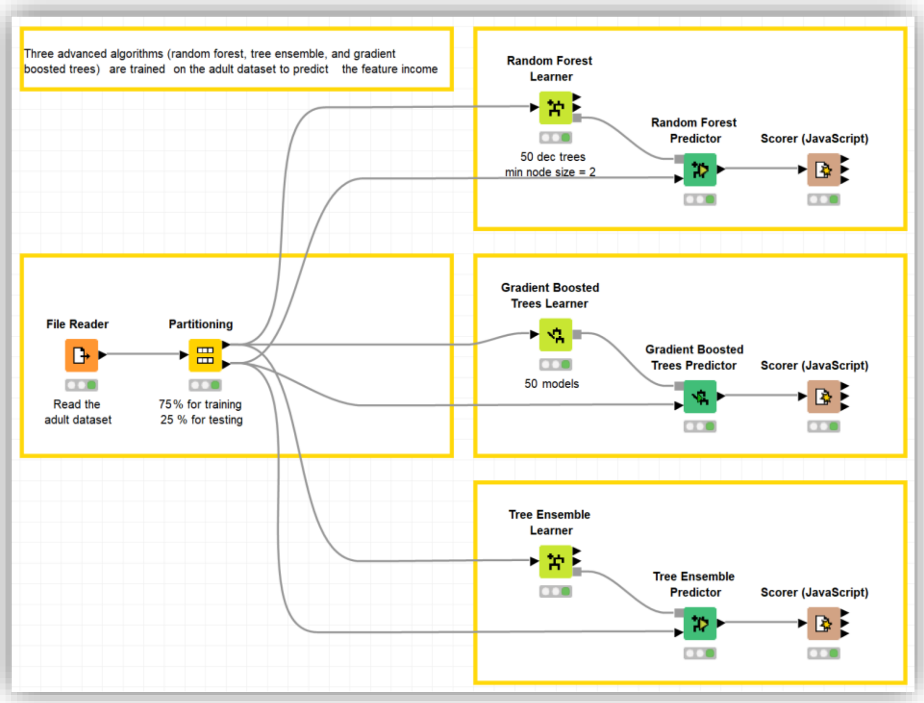
**This lesson refers to chapter 9 of the GIDS book*

Content of this lesson

- Wisdom of the Crowd
- Bagging & Boosting
- Stacking and Cascade Generalization
- Cascading and Delegating
- Tree Ensembles and Random Forest
- AdaBoost
- Gradient Boosted Trees
- Practical Examples

Datasets

- Datasets used : adult dataset
  - „Random Forest, Gradient Boosted Trees, and Tree Ensemble“ https://kni.me/w/Ueq3QR9hty8Osh2E
    - Random forest
    - Gradient boosting
    - Tree ensemble



Three advanced algorithms (random forest, tree ensemble, and gradient boosted trees) are trained on the adult dataset to predict the feature income

Ensemble Models

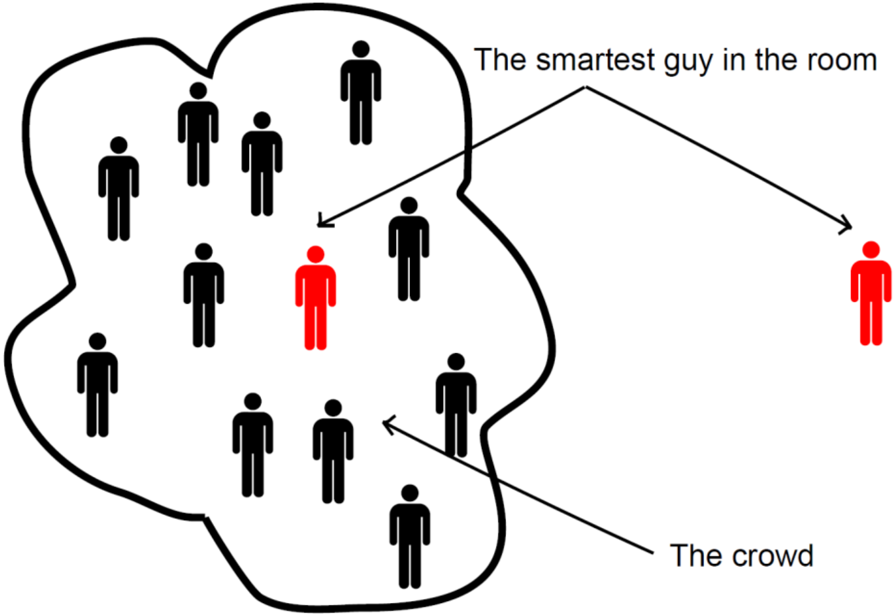
- General idea: take advantage of the “wisdom of the crowd”

- Training of many weak classifiers (or regression models)
- Combining them to construct a classifier (regression model) more accurate than any of the individual ones

- Leads to a more accurate and robust model
- Interpretation of an ensemble learning model is difficult
  - Since it consists of many models!

Wisdom of the Crowd

Wisdom of the Crowd



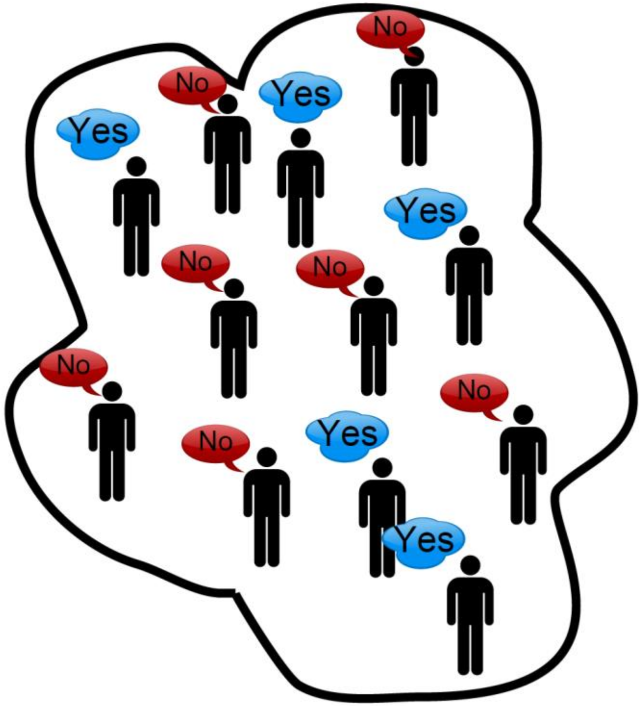
Crowd wiser than **any individual**

- When?
- For which questions?

- The **collective knowledge** of a *diverse* and *independent* body of people typically **exceeds** the knowledge of **any single individual** and can be harnessed by voting.
- http://www.csc.kth.se/utbildning/kth/kurser/DD2431/ml11/schedule/07-ensamble.pdf

Wisdom of the Crowd: Scenario

- Ask each person in the crowd:
- Will Mr. X win the general election in country Y?



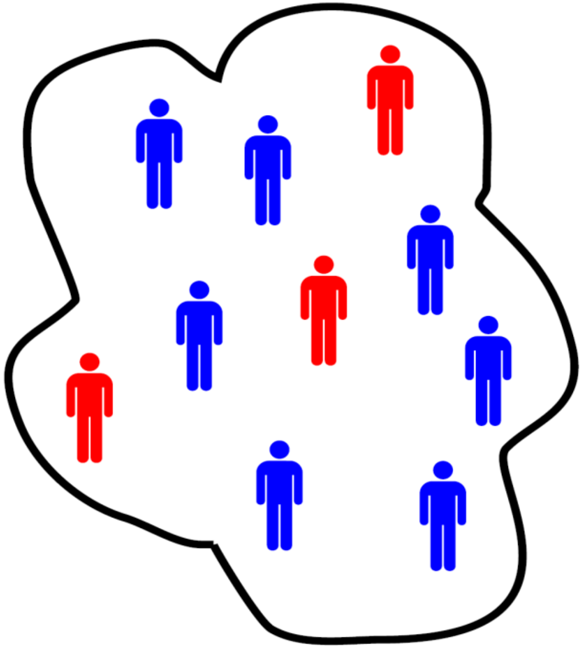
- **The Crowd’s prediction:**
- MAJORITY answer.
- This crowd predicts **No**. (Mr. X will not win the election.)

Wisdom of the Crowd: Scenario

- Has the crowd made a good prediction?

- Composition of crowd:
  - 30% EXPERTS.
  - 70% NON-EXPERTS.

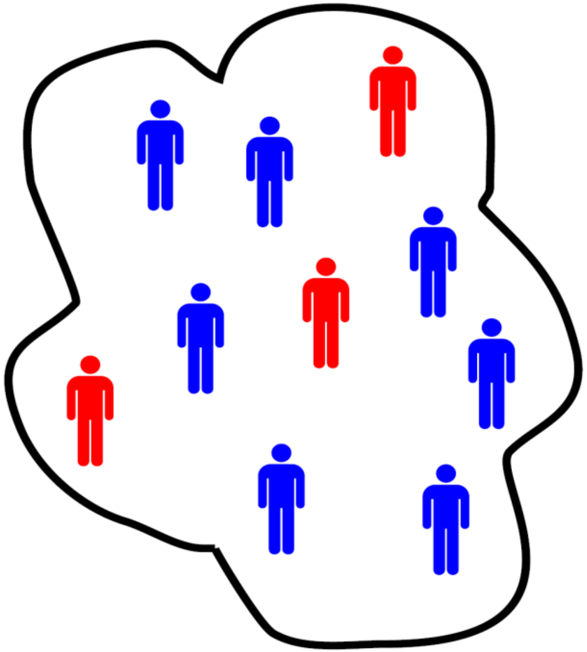
- and their level of expertise:
  - P(correct predict|expert) = p_e
  - P(correct predict|non-expert) = p_{ne}



Wisdom of the Crowd: Scenario

- Let $p_e = 0.8$ and $p_{ne} = 0.5$

- For a random person from the crowd
- P(correct predict|individual) = $0.3\ p_e + 0.7 p_{ne} = 0.59$

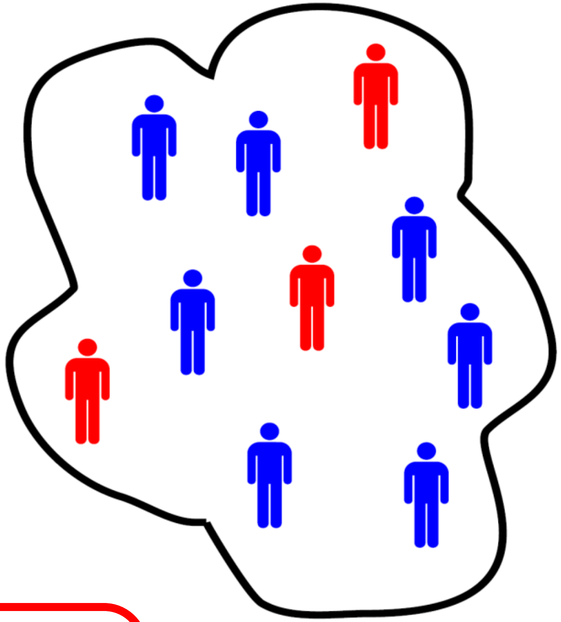


Wisdom of the Crowd: Scenario

- Let $p_e = 0.8$ and $p_{ne} = 0.5$
- For a random person from the crowd
- P(correct predict|individual) = $p_i = 0.59$

- **If crowd contains 50 *independent* people:**
- P(correct predict|crowd)

$$= \sum_{k=26}^{50} \binom{50}{k} {p_i}^k \cdot (1 - p_i)^{50-k} = 0.8745$$



- This crowd has made a prediction with probability .875 of being correct which is $> p_i$.
- **It is wiser than each of the experts!**

Wisdom of the Crowd

- Ensemble of predictors often outperform individual predictors
- Consider a majority voting of 5 independent classifiers in a binary classification problem.
- Each predictor, the error probability is 0.3
- Probability of three or more predictors yielding a wrong result (i.e., the majority misclassifies) is very low:

$$\sum_{i=3}^{5} \binom{5}{i} 0.3^{i} \cdot 0.7^{5-i} = 0.08748$$

- Substantial reduction in error rate!
- In reality, classifiers are rarely independent of each other

But ...

- **Why didn’t I just asked a bunch of experts??**

- Large enough crowd
  - ➔ high probability that a sufficient number of experts will be in crowd (for any question).

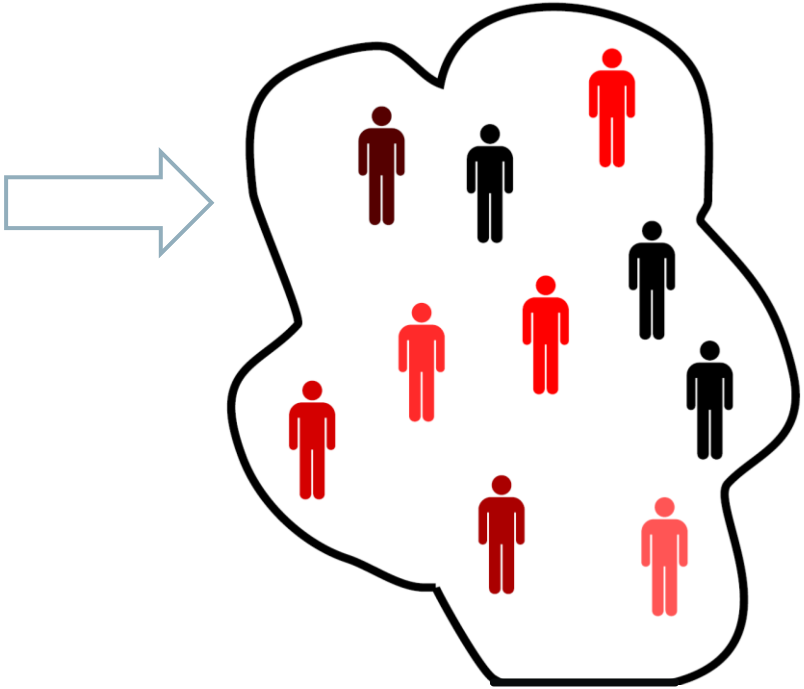
- Random selection
  - ➔ don’t make a biased choice in experts.

- For some questions it may be hard to identify a diverse set of experts

For a random crowd

- Given a **random question** expect each **person** to have a **different level** of **expertise**.

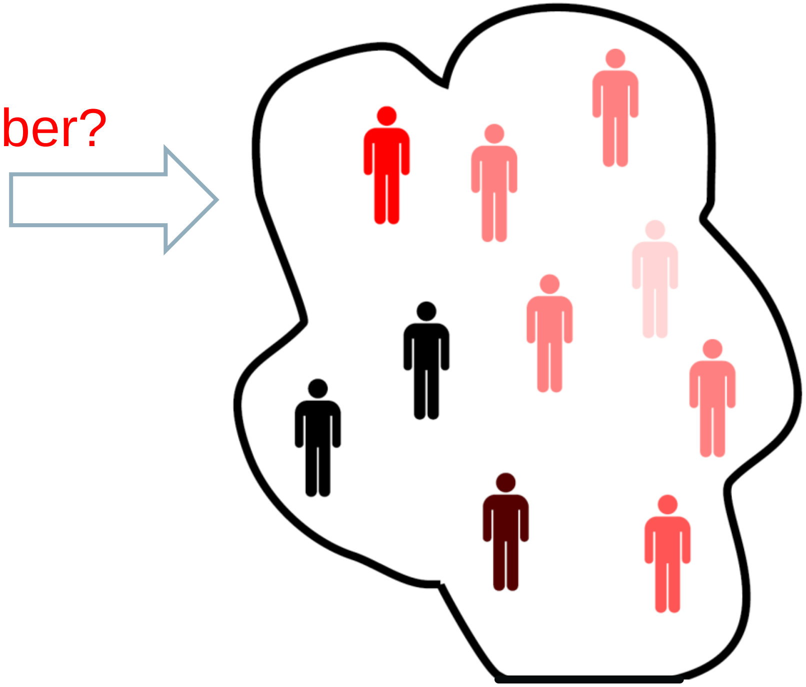
- Will it rain tomorrow?
  - redness proportional to expertise



For a random crowd

- Given a **random question** expect each **person** to have a **different level** of **expertise**.

- Will the world go down in December?
  - redness proportional to expertise



What makes a crowd wise?

According to *James Surowiecki* there are four elements required to form a wise crowd:

- **Diversity of opinion.** People in crowd should have a range of experiences, education and opinions. (Encourages independent predictions)
- **Independence.** Prediction by person in crowd is not influenced by other people in the crowd.
- **Decentralization.** People have specializations and local knowledge.
- **Aggregation.** There is a mechanism for aggregating all predictions into one single prediction.

The crowd must be careful

In the analysis of the crowd it is implicitly assumed:

- each person is not concerned with the opinions of others, no-one is copying anyone else in the crowd.

In the analysis of the crowd we implicitly assumed:

- The non-experts will predict a **completely random wrong answer** - these will cancel each other out (to some degree).

However, there may be a systematic and consistent bias in the non-experts' predictions.

… Back to Ensemble Models

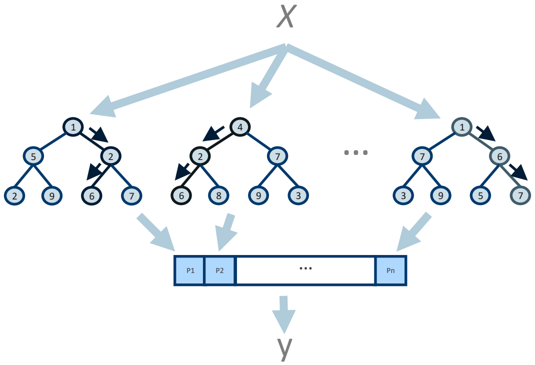
Back to Machine Learning

We will exploit **Wisdom of crowd** ideas for specific tasks by:

- combining (classifier) predictions
- aim to combine independent and diverse predictors (classifiers).

We can also use labeled training data

- to identify the expert classifiers in the pool;
- to identify complementary classifiers;
- to indicate how to best combine them.



Why Do Ensemble Methods Work?

Statistical reason:

- Able to average many good models
- Reduces the influence of bad models

Computational reason:

- Able to explore the model space efficiently

Representational reason:

- Reduce the bias of a learning algorithm by extending its model space

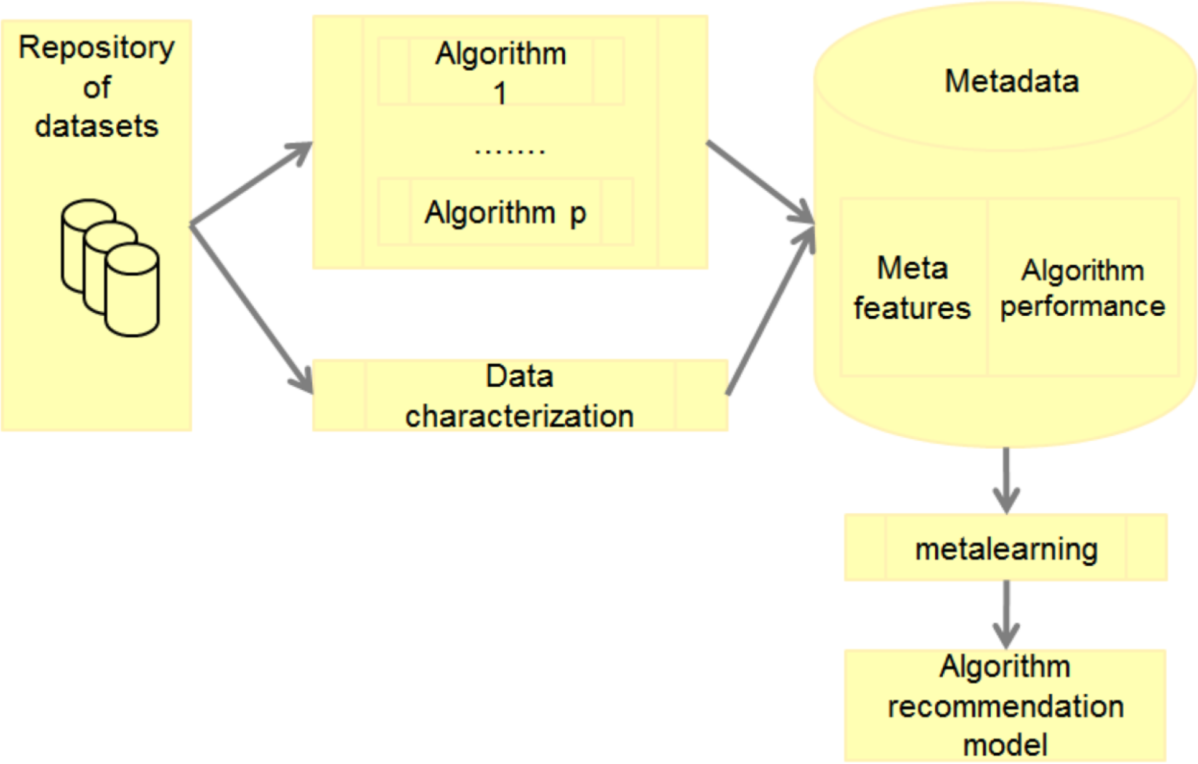
Basic Terms

Remember?

- Bias = model error + algorithmic error
  - **Model error**: the error we get by selecting a model
  - **Algorithmic error**: by selecting the algorithm itself and the parameters of the algorithm

- Base-Learning: Fixed Bias / User parameterized
- Meta-Learning: Dynamic bias selection using meta-knowledge
- Meta-Knowledge: Knowledge achieved during the learning process

Ensemble Learning for Algorithm Recommendation



Combining base-learners: Categories

Philosophy	Technique
Bagging, Boosting	Variation in data
Stacking	Variation among learners (multi-expert)
Cascading, Delegating	Variation among learners (multi-stage)
Arbitrating	Variation among learners (refereed)
Meta decision trees	Variation in data and among learners

Bagging and Boosting

Bagging and Boosting

- Best-known techniques
- Based on selection of multiple data sub sets
- Meta model is created by combining the base models

- Advantages:
  - Reduces overfitting
  - Most effective when the base learner is highly sensitive to data
  - Typically increases accuracy

- Disadvantages:
- Interpretability of interpretable base learners is lost

Bagging

Bagging :

- Select N independent samples of the Training Data
- Learn one model on each of the samples ➔ $h_1, \ldots, h_N$

- Classification: Use the class most predicted by all classifiers
- Regression: Use the mean of all predictions

Boosting

Boosting:

- Tries to learn a weighting for the models
- Later base learners focus more on the examples that previous base learners misclassified
- There is no single “best” boosting method

One boosting method after Schapire

Training :

- Create *c*1: base learner on a sample *t*1 of the data
- Create *t*2: sample which is 50% misclassified by *c*1
- Create *c*2: base learner on the sample *t*2
- Create *t*3: subset of the data where *c*1 predicts differently than *c*2
- Create *c*3: base learner on the sample *t*3

Classification :

- Classify with *c*1 and *c*2
- If unequal, use *c*3 as final classification

Stacking and Cascade Generalization

Stacking

- In Bagging and Boosting: we used always the same base learner
- Stacking exploits differences among base learners

- Two levels of learning
  1. Base learners are trained, each on the whole data set
  2. Meta learners are created on meta data (e.g. predicted class) obtained in level 1

- Two levels of classifying
  1. Base learner are used on data point
  2. Meta learners are applied on base learner predictions

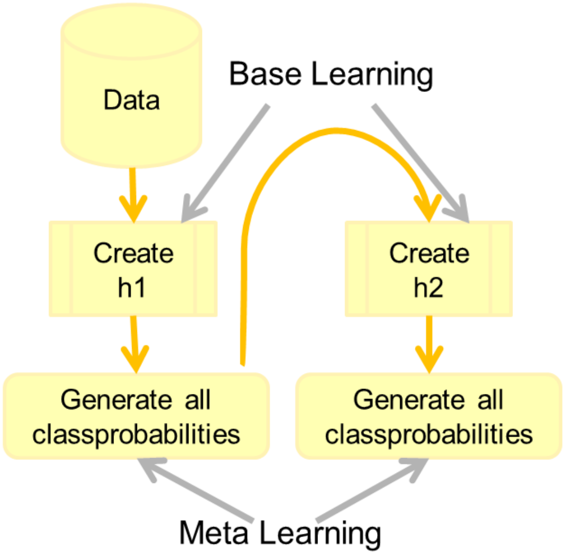
Stacking: Base learners are used in parallel

Cascade Generalization

- Base learners are used in a sequence with ”partial” meta-learners
- Knowledge from previous classifiers can be used in later ones
- After each base learner has been trained, the data set is adjusted using the new information

For classification :

- Only the last model is used, which incorporates the knowledge from previous models (all base methods are used)


Data
Base Learning
Create h1
Create h2
Generate all classprobabilities
Generate all classprobabilities
Meta Learning

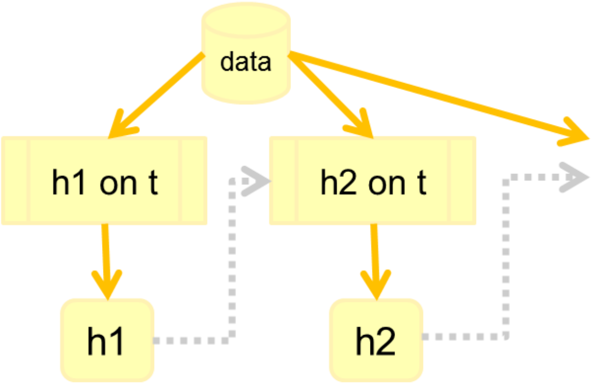
Cascading and Delegating

Cascading and Delegating

- Until now: all base classifiers are used for classification
- Here: Multistage classifiers, not all are required for classification
- Main advantage: faster classification

Cascading

- Multilearner version of boosting
- Uses learned confidence of previous models
- Train base learner h_i using knowledge from previous base learner...
- ...on data, which was most probably misclassified by previous learners



- **Classification:** go through all base models, stop and use as classification if the model has confidence greater than epsilon

Delegating

- Cascading: all instances are used in each step

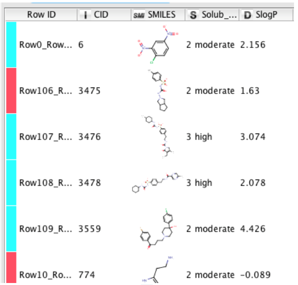
- **Delegating**: only instances below confidence threshold are processed in the next step

- Idea:
  - Use everything and test for which data points you are good enough
  - Pass the remaining work to someone else.
  - If there is no someone else, ... guess
- Advantages:
  - Still understandable (no model combination)
  - Improved efficiency, due to the decreasing number of examples.

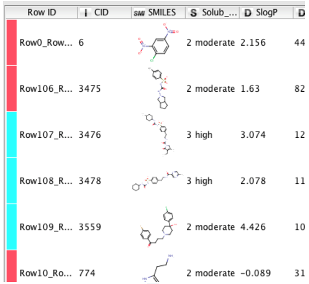
Tree Ensembles and Random Forest

Bagging - Idea

- Bagging ≡ **B**ootstrap **AGG**regatI**NG**
- For each tree / model a training set is generated by sampling uniformly with replacement from the standard training set

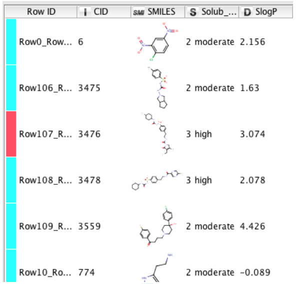


	Row ID	CID	SMILES	Solub_...	SlogP
🟦	Row0_Row...	6		2 moderate	2.156
🟥	Row106_R...	3475		2 moderate	1.63
🟦	Row107_R...	3476		3 high	3.074
🟦	Row108_R...	3478		3 high	2.078
🟦	Row109_R...	3559		2 moderate	4.426
🟥	Row10_Ro...	774		2 moderate	-0.089

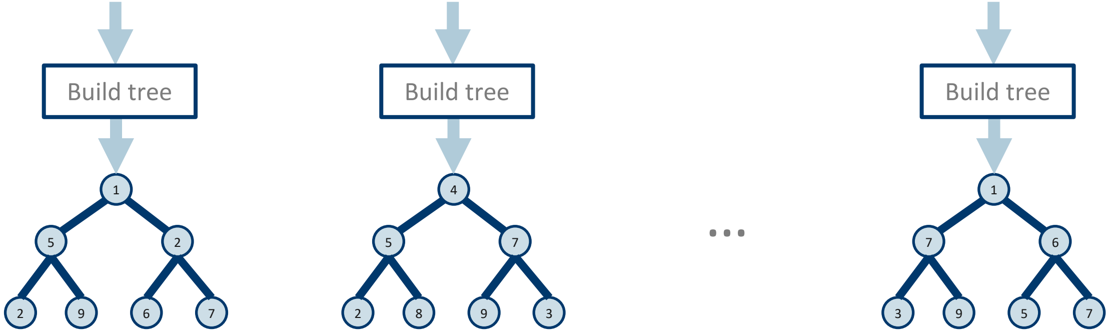


	Row ID	CID	SMILES	Solub_...	SlogP	D
🟥	Row0_Row...	6		2 moderate	2.156	44
🟥	Row106_R...	3475		2 moderate	1.63	82
🟦	Row107_R...	3476		3 high	3.074	12
🟦	Row108_R...	3478		3 high	2.078	11
🟦	Row109_R...	3559		2 moderate	4.426	10
🟥	Row10_Ro...	774		2 moderate	-0.089	31

...



	Row ID	CID	SMILES	Solub_...	SlogP
🟦	Row0_Row...	6		2 moderate	2.156
🟦	Row106_R...	3475		2 moderate	1.63
🟥	Row107_R...	3476		3 high	3.074
🟦	Row108_R...	3478		3 high	2.078
🟦	Row109_R...	3559		2 moderate	4.426
🟦	Row10_Ro...	774		2 moderate	-0.089



Example for Bagging

Full training set

RowID	x_1	x_2	y
Row_1	2	6	Class 1
Row_2	4	1	Class 2
Row_3	9	3	Class 2
Row_4	2	7	Class 1
Row_5	8	1	Class 2
Row_6	2	6	Class 1
Row_7	5	2	Class 2

Sampled training set

RowID	x_1	x_2	y
Row_3	9	3	Class 2
Row_6	2	6	Class 1
Row_1	2	6	Class 1
Row_3	9	3	Class 2
Row_5	8	1	Class 2
Row_6	2	6	Class 1
Row_1	2	6	Class 1

Bagging - Idea

Why sampling with replacement?

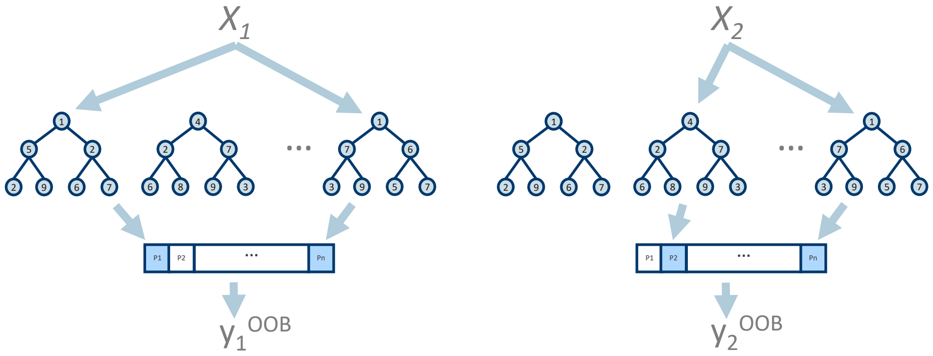
➔ So that a sample approximates the distribution of the population

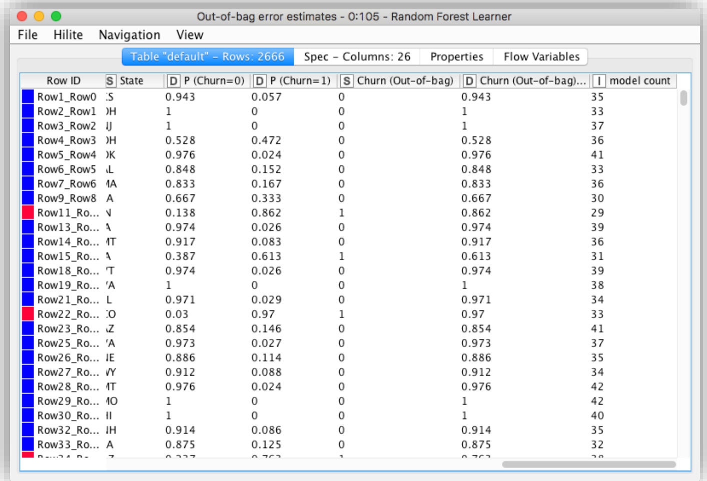
- Frequent values are represented more
- Less frequent values are represented less

Ultimately leads to the model with smaller variance and smaller bias

An Extra Benefit of Bagging: Out of Bag Estimation

- Able to evaluate the model using the training data
- Apply trees to samples that haven’t been used for training

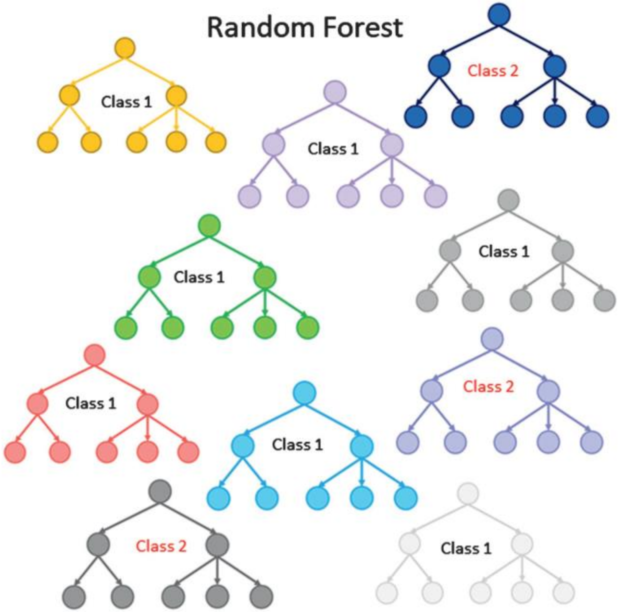



Out-of-bag error estimates - 0:105 - Random Forest Learner
File
Hilite
Navigation
View
Table "default" – Rows: 2666
Spec – Columns: 26
Properties
Flow Variables

	Row ID	S State	D P (Churn=0)	D P (Churn=1)	S Churn (Out-of-bag)	D Churn (Out-of-bag)...	I model count
🟦	Row1_Row0	:S	0.943	0.057	0	0.943	35
🟦	Row2_Row1	)H	1	0	0	1	33
🟦	Row3_Row2	IJ	1	0	0	1	37
🟦	Row4_Row3	)H	0.528	0.472	0	0.528	36
🟦	Row5_Row4	)K	0.976	0.024	0	0.976	41
🟦	Row6_Row5	\L	0.848	0.152	0	0.848	33
🟦	Row7_Row6	ИA	0.833	0.167	0	0.833	36
🟦	Row9_Row8	A	0.667	0.333	0	0.667	30
🟥	Row11_Ro...	N	0.138	0.862	1	0.862	29
🟦	Row13_Ro...	\	0.974	0.026	0	0.974	39
🟦	Row14_Ro...	ИT	0.917	0.083	0	0.917	36
🟦	Row15_Ro...	\	0.387	0.613	1	0.613	31
🟦	Row18_Ro...	'T	0.974	0.026	0	0.974	39
🟦	Row19_Ro...	'A	1	0	0	1	38
🟦	Row21_Ro...	L	0.971	0.029	0	0.971	34
🟥	Row22_Ro...	:O	0.03	0.97	1	0.97	33
🟦	Row23_Ro...	\Z	0.854	0.146	0	0.854	41
🟦	Row25_Ro...	'A	0.973	0.027	0	0.973	37
🟦	Row26_Ro...	IE	0.886	0.114	0	0.886	35
🟦	Row27_Ro...	VY	0.912	0.088	0	0.912	34
🟦	Row28_Ro...	ИT	0.976	0.024	0	0.976	42
🟦	Row29_Ro...	ИO	1	0	0	1	42
🟦	Row30_Ro...	II	1	0	0	1	40
🟦	Row32_Ro...	IH	0.914	0.086	0	0.914	35
🟦	Row33_Ro...	A	0.875	0.125	0	0.875	32

Random Forest

- Bag of decision trees, with an extra element of randomization
- Each node in the decision tree only “sees” a subset of the input features ➔ ***Random Subspace Selection***
- typically $\sqrt{n}$ to pick from
- Random forests tend to be very robust w.r.t. overfitting



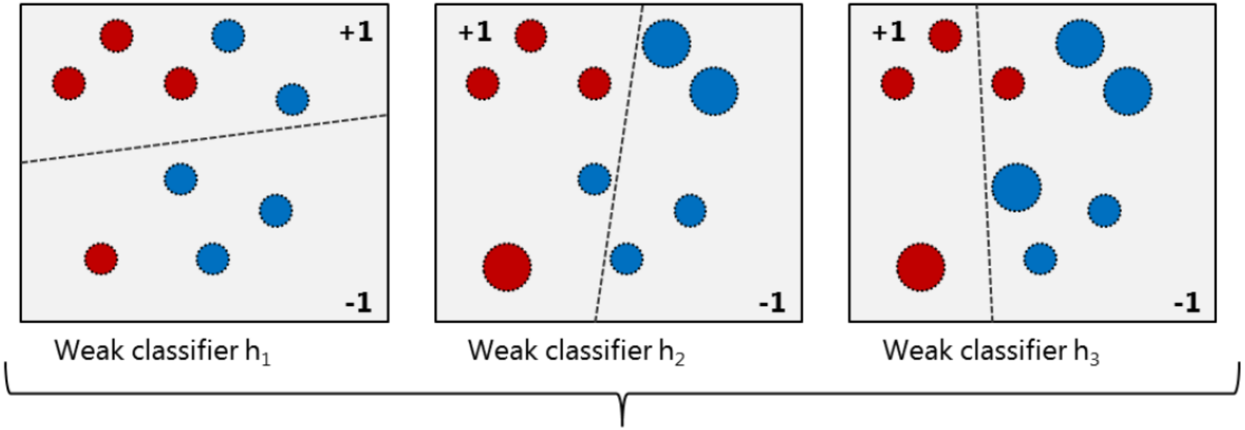
AdaBoost

AdaBoost

- Freund & Schapire (1995)

- AB is a linear classification algorithm
- AB has good generalization properties
  ➔(Avoids overfitting as long as the training data is not too noisy)
- AB is a feature selector

AdaBoost



Strong classifier = $(\alpha_1\ h_1) + (\alpha_t\ h_t) + \dots + (\alpha_T\ h_T)$

- AdaBoost classifier: $\sum_{t=1}^{T}(\alpha_t h_t(\boldsymbol{x}))$
- Where a weak classifier $h_t(\boldsymbol{x})$ is weighted by α_t for steps up to T
- Misclassified data points are weighted more in subsequent steps
- Classification result: $H(\boldsymbol{x}) = sign(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}))$

AdaBoost: Terminology

- Strong Classifier: $\sum_{t=1}^{T}(\alpha_t h_t(\boldsymbol{x}))$

- Where $h_t(\boldsymbol{x})$ is a base classifier weighted by α_t

- Classification result:

$$\mathrm{H}(\boldsymbol{x}) = sgn\left(\sum_{t=1}^{T}(\alpha_t h_t(\boldsymbol{x}))\right)$$

AdaBoost

- Initially, all data points are given the same weight $w_{i,1} = 1/n$.
- At step t, the classifier weight α_t is calculated as

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1 - e_t}{1 + e_t}\right)$$

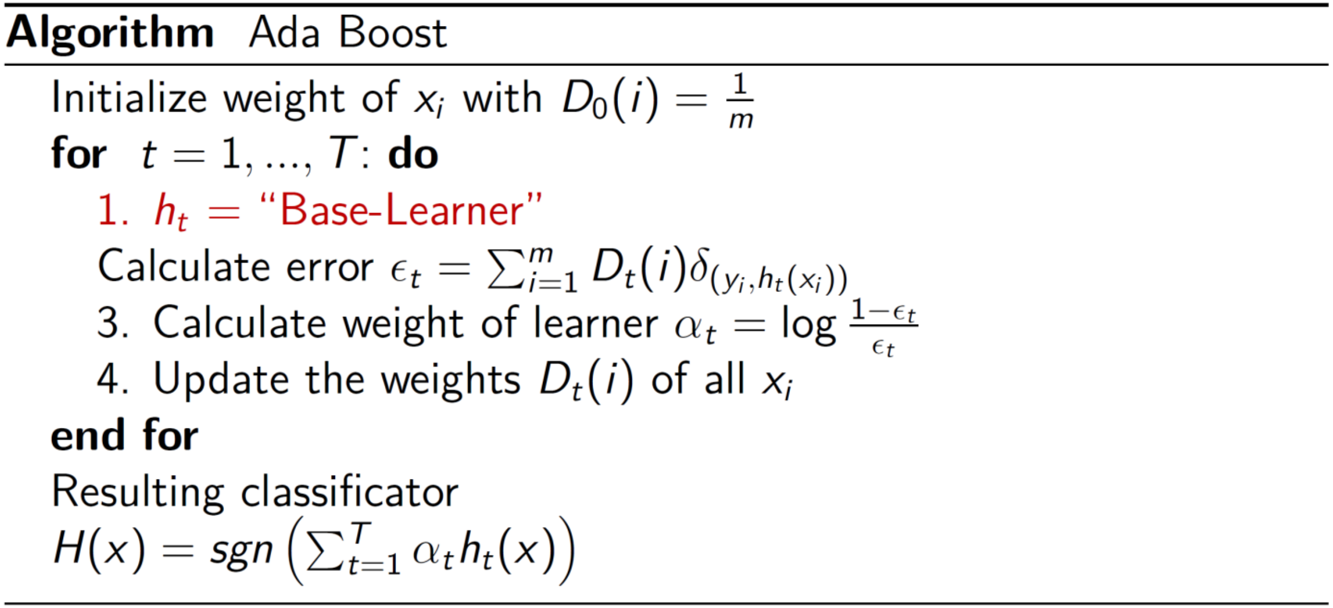
- Where e_t is given by

$$e_t = \frac{\sum_{i=1}^{n} w_{i,t} y_i h_t(\boldsymbol{x}_i)}{\sum_{i=1}^{n} w_{i,t}}$$

- Weights w’s are updated as

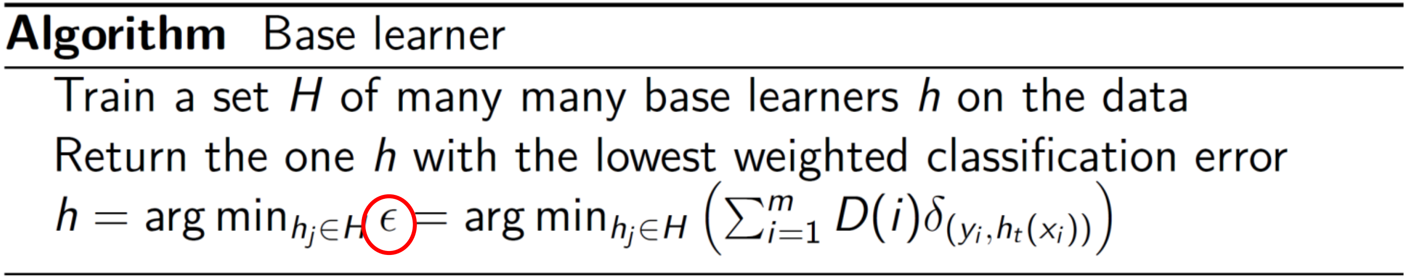
$$w_{i,t+1} = \frac{w_{i,t}\exp(-\alpha_t y_i h_t(\boldsymbol{x}_i))}{\sum_{j=1}^{n} w_{j,t} \exp(-\alpha_t y_j h_t(\boldsymbol{x}_j))}$$

Adaboost pseudo-code Overview



```
Algorithm Ada Boost
  Initialize weight of $x_i$ with $D_0(i) = \frac{1}{m}$
  for $t = 1, ..., T$: do
      1. $h_t$ = "Base-Learner"
      Calculate error $\epsilon_t = \sum_{i=1}^{m} D_t(i)\delta_{(y_i, h_t(x_i))}$
      3. Calculate weight of learner $\alpha_t = \log \frac{1-\epsilon_t}{\epsilon_t}$
      4. Update the weights $D_t(i)$ of all $x_i$
  end for
  Resulting classificator
  $H(x) = sgn\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$
```

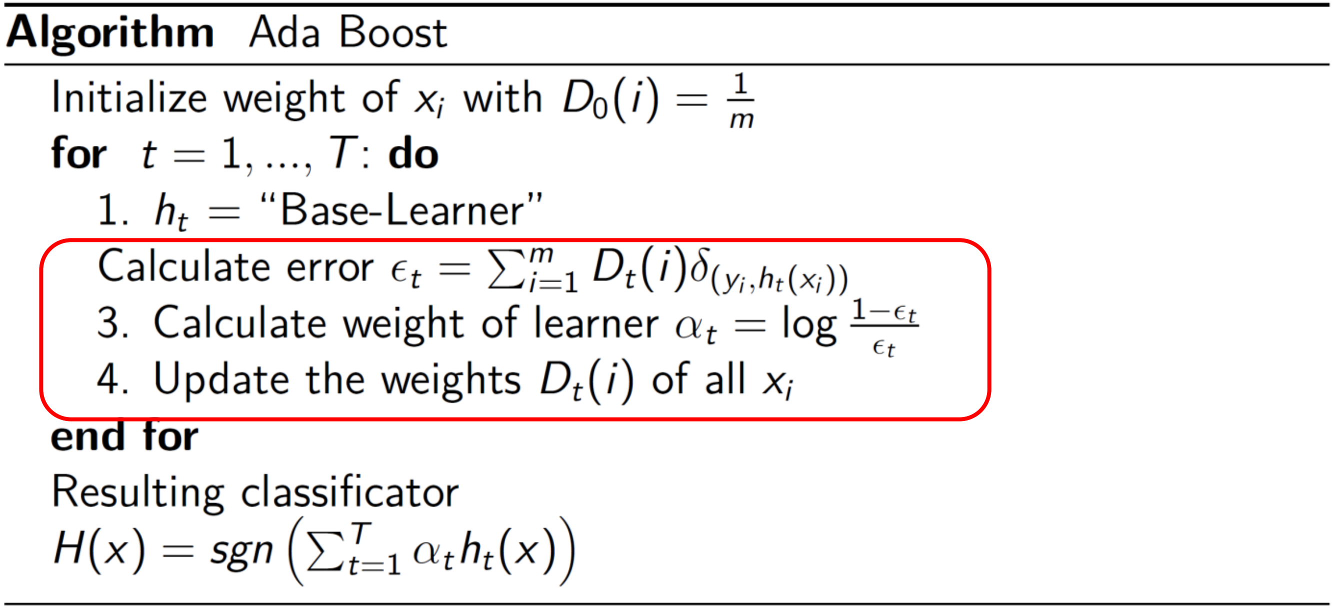
Adaboost pseudo-code Overview



```
Algorithm  Base learner
    Train a set $H$ of many many base learners $h$ on the data
    Return the one $h$ with the lowest weighted classification error
    $h = \arg\min_{h_j \in H} \epsilon = \arg\min_{h_j \in H} \left( \sum_{i=1}^{m} D(i)\delta_{(y_i, h_t(x_i))} \right)$
```

Make sure $\epsilon < 0.5$

Adaboost pseudo-code Overview



```
Algorithm Ada Boost
    Initialize weight of $x_i$ with $D_0(i) = \frac{1}{m}$
    for $t = 1, ..., T$: do
        1. $h_t$ = "Base-Learner"
        Calculate error $\epsilon_t = \sum_{i=1}^{m} D_t(i)\delta_{(y_i, h_t(x_i))}$
        3. Calculate weight of learner $\alpha_t = \log \frac{1-\epsilon_t}{\epsilon_t}$
        4. Update the weights $D_t(i)$ of all $x_i$
    end for
    Resulting classificator
    $H(x) = sgn\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$
```

Update of weights of Training Samples

- Update weights and normalize them:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \exp(-\alpha_t \, y_i \, h_t(x_i))$$

$$Z_t(i) = \sum_{i=0}^{m} D_t(i) \exp(-\alpha_t \, y_i \, h_t(x_i))$$

- Weight of correct classified examples is decreased
- Weight of incorrect classified examples is increased

AdaBoost: Why does it work?

- **Weighting the base learners** “Upper-Bound Theorem”
- Primary goal is to minimize

$$\epsilon_{tr}(H) = \frac{1}{m}\ |\{i: H(x_i \neq y_i)\}|$$

- Global error is bounded by

$$\epsilon_{tr}(H) \leq Z_t \qquad t = 1, \dots, T$$

$$Z_t = \sum_{i=0}^{m} D_t(i) \exp(-\alpha_t\ y_i\, h_t(x_i))$$

AdaBoost: Why does it work?

- **Weighting the base learners**

- That's why
- Minimizing $Z_t = \sum_{i=0}^{m} D_t(i) \exp(-\alpha_t\ y_i\ h_t(x_i))$ results in a minimization of the global error

- Upper-Bound can be minimized by ...
  1. Choosing the optimal hypothesis h_t ...
  2. ... with an optimal weight α_t

- Minimizing Z_t results in $\alpha_t = \log \frac{1 - \epsilon_t}{\epsilon_t}$

AdaBoost

Advantages

- Very simple to implement
- Feature selection on very large features spaces
- Fairly good generalization

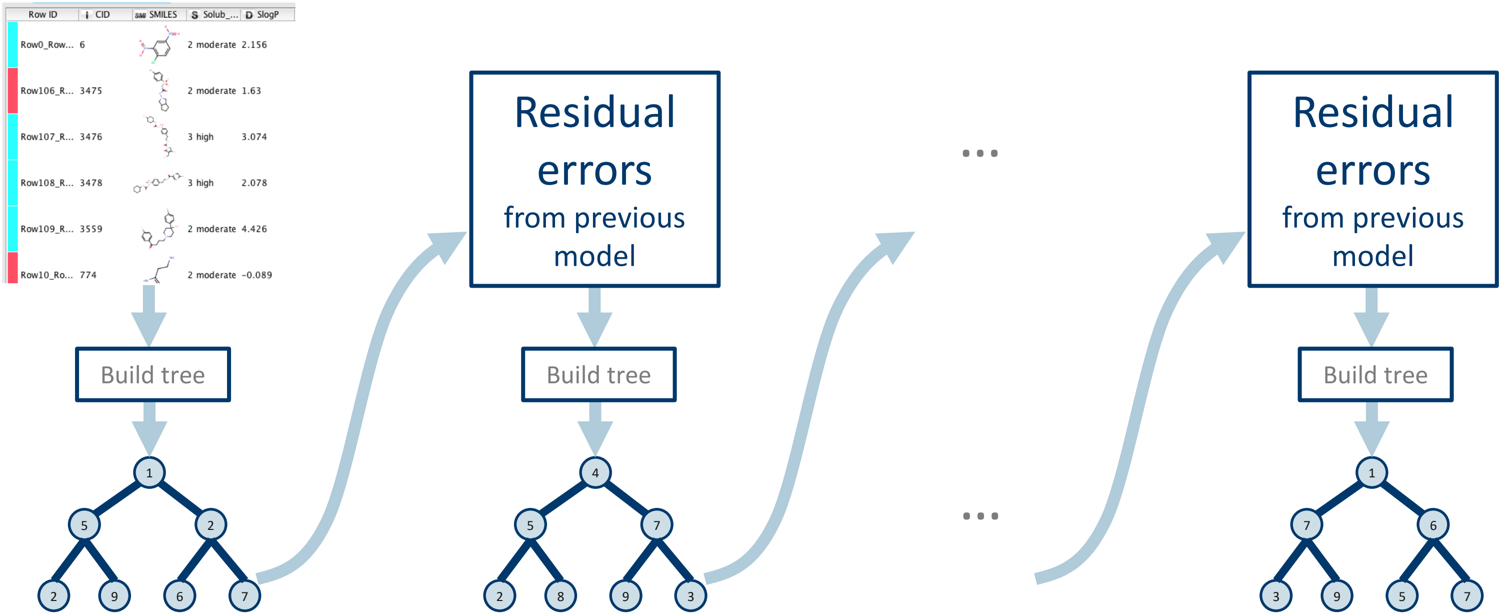
Disadvantages

- Can overfit in presence of noise
- Unclear which weak-learning algorithm fits best for a given problem

Gradient Boosted Trees

Boosting - Idea

- Starts with a single tree built from the data
- Fits a tree to residual errors from the previous model to refine the model sequentially



	Row ID	CID	SMILES	Solub_...	SlogP
🟦	Row0_Row...	6		2 moderate	2.156
🟥	Row106_R...	3475		2 moderate	1.63
🟦	Row107_R...	3476		3 high	3.074
🟦	Row108_R...	3478		3 high	2.078
🟦	Row109_R...	3559		2 moderate	4.426
🟥	Row10_Ro...	774		2 moderate	-0.089

Build tree
1
5
2
2
9
6
7
Residual errors from previous model
Build tree
4
5
7
2
8
9
3
...
...
Residual errors from previous model
Build tree
1
7
6
3
9
5
7

Gradient Boosted Trees

A shallow tree (depth 4 or less) is built at each step

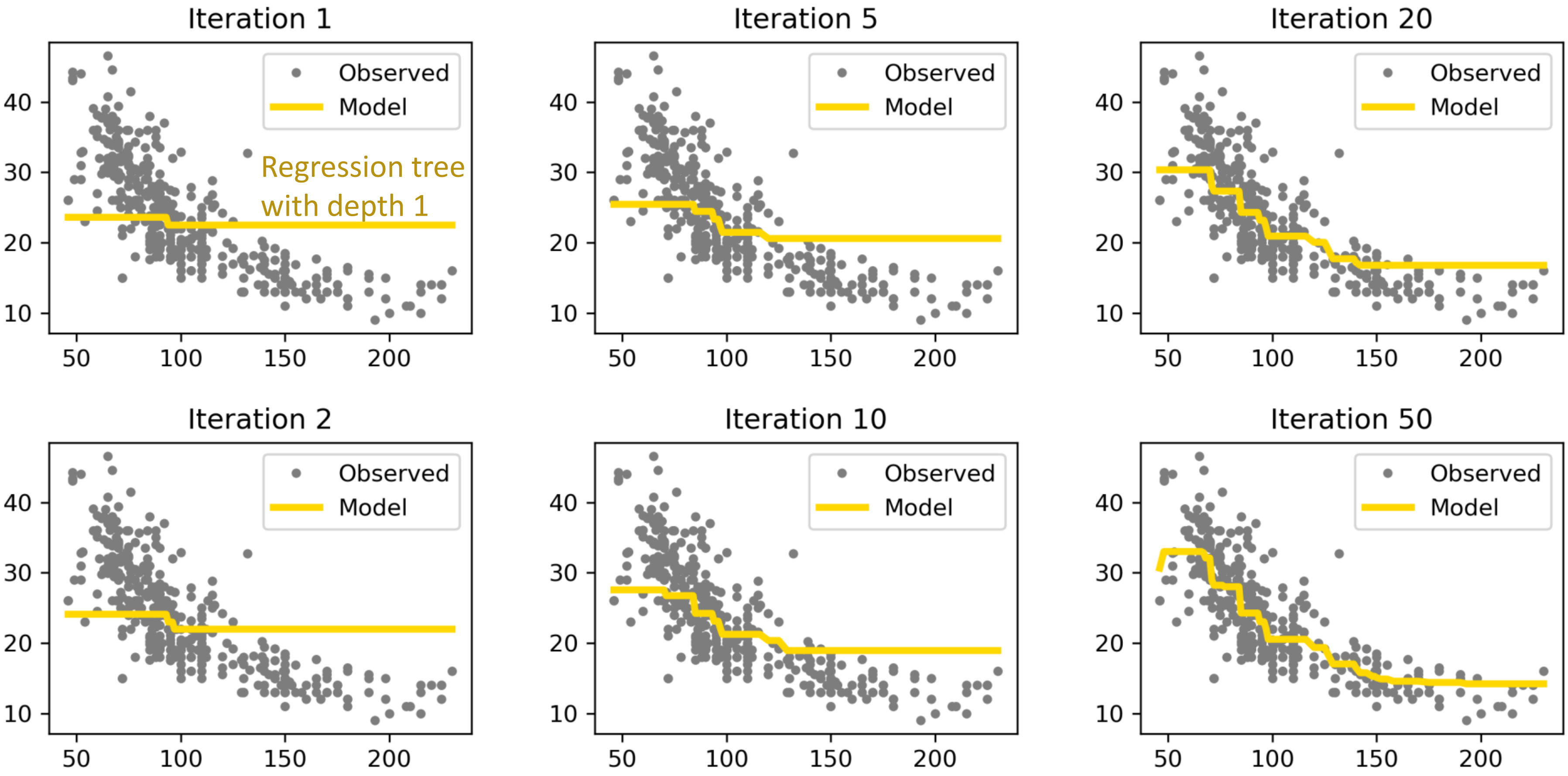
- To fit residual errors from the previous step
- Resulting in a tree $h_m(x)$

The resulting tree is added to the latest model to update

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

- Where $F_{m-1}(x)$ is the model from the previous step
- The weight γ_m is chosen to minimize the loss function
  - Loss function: quantifies the difference between model predictions and data

Gradient Boosted Trees Example – Regression



Iteration 1

Regression tree with depth 1

Iteration 2

Iteration 5

Iteration 10

Iteration 20

Iteration 50

Gradient Boosted Trees

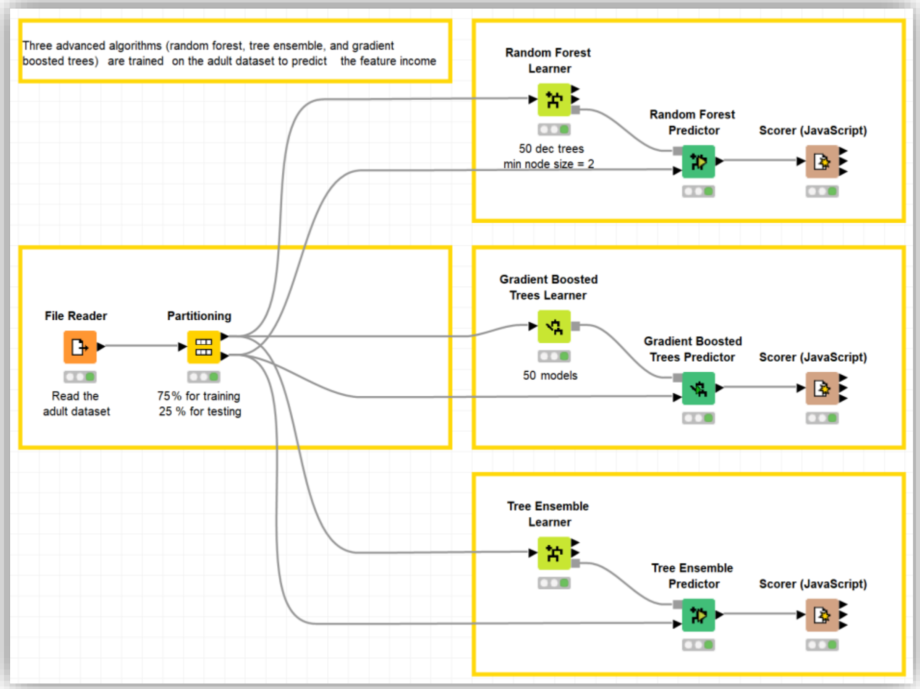
- Can be used for classification and regression

- Large number of iterations – prone to overfitting
  - ~100 iterations are sufficient

- Can introduce randomness in choice of data subsets (“stochastic gradient boosting”) and choice of input features

Practical Examples with KNIME Analytics Platform

Tree Ensemble, Random Forest, and Gradient Boosted Tree



Three advanced algorithms (random forest, tree ensemble, and gradient boosted trees) are trained on the adult dataset to predict the feature income

- Tree Ensemble, Random Forest, and Gradient Boosted Tree applied to the adult dataset

Thank you

For any questions please contact: education@knime.com