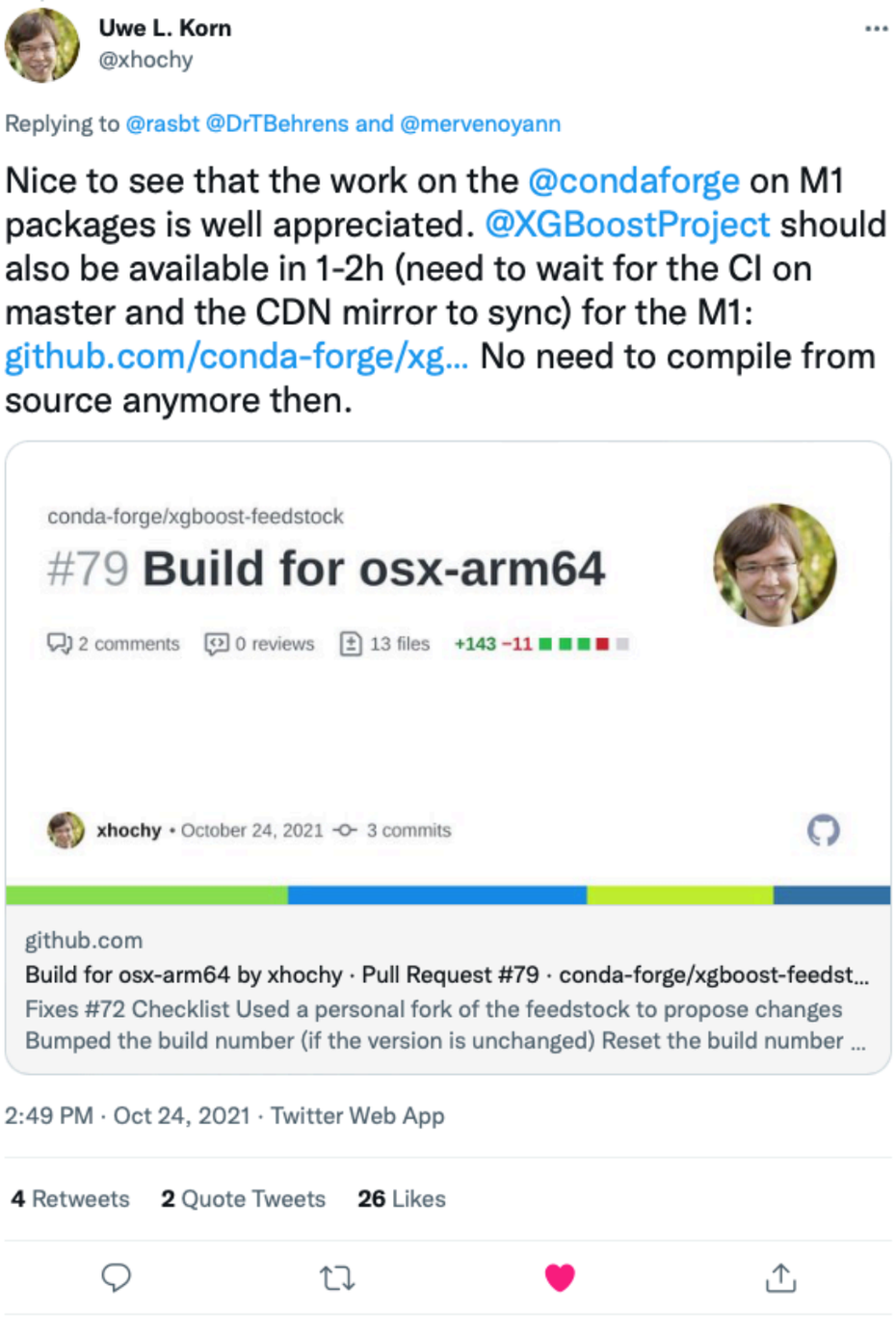
Lecture 07

Ensemble Methods Part 3/3

STAT 451: Machine Learning, Fall 2021

Sebastian Raschka




```
(base) sebastian@Sebastian-Air ~/Desktop % conda install xgboost
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /Users/sebastian/miniforge3

  added / updated specs:
    - xgboost


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    _py-xgboost-mutex-2.0      |            cpu_0          12 KB  conda-forge
    libxgboost-1.5.0           |       hcfdfaf5_0         2.0 MB  conda-forge
    py-xgboost-1.5.0           |   py39h2804cbe_0         151 KB  conda-forge
    xgboost-1.5.0              |   py39ha480839_0          12 KB  conda-forge
    ------------------------------------------------------------
                                           Total:         2.2 MB

The following NEW packages will be INSTALLED:

  _py-xgboost-mutex  conda-forge/osx-arm64::_py-xgboost-mutex-2.0-cpu_0
  libxgboost         conda-forge/osx-arm64::libxgboost-1.5.0-hcfdfaf5_0
  py-xgboost         conda-forge/osx-arm64::py-xgboost-1.5.0-py39h2804cbe_0
  xgboost            conda-forge/osx-arm64::xgboost-1.5.0-py39ha480839_0


Proceed ([y]/n)?


Downloading and Extracting Packages
xgboost-1.5.0         | 12 KB     | ############################################
| 100%
py-xgboost-1.5.0      | 151 KB    | ############################################
| 100%
_py-xgboost-mutex-2.  | 12 KB     | ############################################
| 100%
libxgboost-1.5.0      | 2.0 MB    | ############################################
| 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

Kaggle Solutions

The Most Comprehensive List of Kaggle Solutions and Ideas

This is a list of almost all available solutions and ideas shared by top performers in the past Kaggle competitions. This list gets updated as soon as a new competition finishes. If you find a solution besides the ones listed here, I would encourage you to contribute to this repo by making a pull request. The symbols used in this list are described here.
If you found it interesting, you may give a star or make a fork

Star 536 Fork 141

Check out the following markdown pages about Top Kagglers Tips/Tricks and all Kernels of The week.

- Top Kagglers Interviews and Lectures
- Kernels of The Week

Type / to search title, description, year, ranking and ...

Last Updated: October 25, 2021

	Title & Description	Details	Solutions	Pins
	**458. Google Landmark Retrieval 2021** Given an image, can you find all of the same landmarks in a dataset?	Prize: Swag Team: 263 Kind: Research Metric: - Year: 2021	1st place 2nd place 3rd place 5th place 6th place 11th place 13th place all solutions	✓
	**457. Google Landmark Recognition 2021** Label famous, and not-so-famous, landmarks in images	Prize: Swag Team: 383 Kind: Research Metric: GoogleGlobalAP Year: 2021	1st place 3rd place 4th place 5th place 8th place all solutions	✓
LEARN	**456. LearnPlatform COVID-19 Impact on Digital Learning** Use digital learning data to analyze the impact of COVID-19	Prize: $20,000 Team: 0 Kind: Analytics	Not Available!	○

https://farid.one/kaggle-solutions/

https://www.kaggle.com/c/tabular-playground-series-may-2021/discussion/243054

Base Models

I ended up training 5 different models, each trained on a 5fold stratified cv on three random seeds, using out of fold predictions for each fold and averaging across folds for predicting the test set. The models were LogisticsRegression, RandomForest, XGBoost, LightGBM, and CATBoost. Weighting classes did not really work for LogisticsRegression and I used the defaults class weights in the rest of them. I used Optuna to lightly tune each model (you will see my final parameters in the code above).

Over/Under Sampling

Did not really try after reading the notebook from @remekkinas that pointed out it likely would not work. His analysis made sense to me and I did not have time, so did not really pursue any further.

Ensembling

Stacking worked quite well here. I tried weighted average of models and it did worse than stacking so I did not use that. I used a 5fold stacking of all models above and used a meta model of RidgeRegerssion, using CalibratedClassifierCV in sklearn.

https://www.kaggle.com/c/ncaam-march-mania-2021/discussion/231665

Summary

We used a random forest model to predict winning outcomes. We considered end of season summarized statistics, key wins / losses, specific Massey Ordinals, and efficiency metrics. We tuned our random forest parameters by selecting the parameters with the best classification accuracy (should have used log loss in hindsight but it worked out for us). We trained multiple models on all seasons through 2014, testing on 2015-2019, and chose the model with the lowest Kaggle score (log-loss).

https://www.kaggle.com/c/prostate-cancer-grade-assessment

Step1: Label cleaning

Model: 0
Train 5-fold EfficientNet-B1 with given **original** labels

→ Predict **out-of-fold** train data

→ Remove data which have high **prediction-original** difference

Get cleaned labels

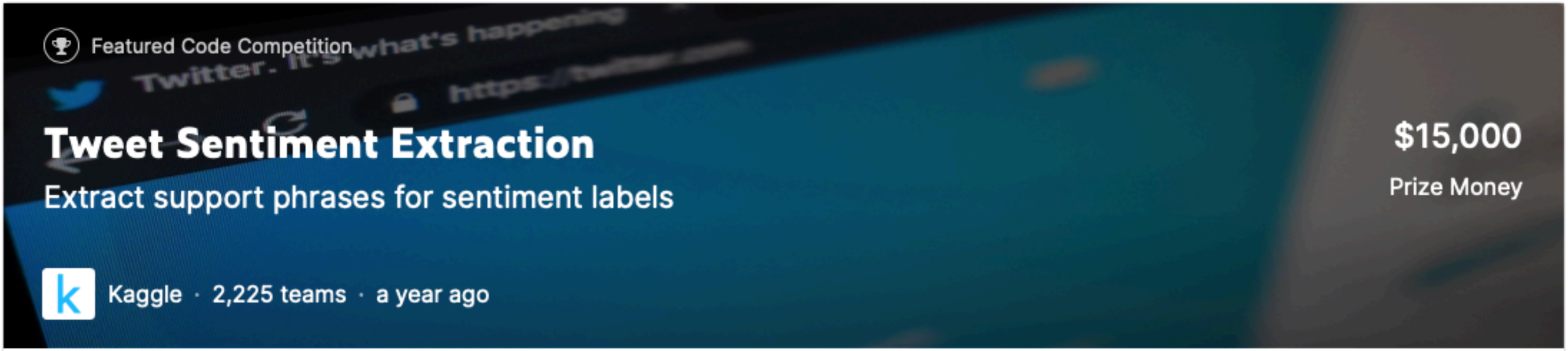
Step2: Training final models

Train models from scratch with cleaned labels

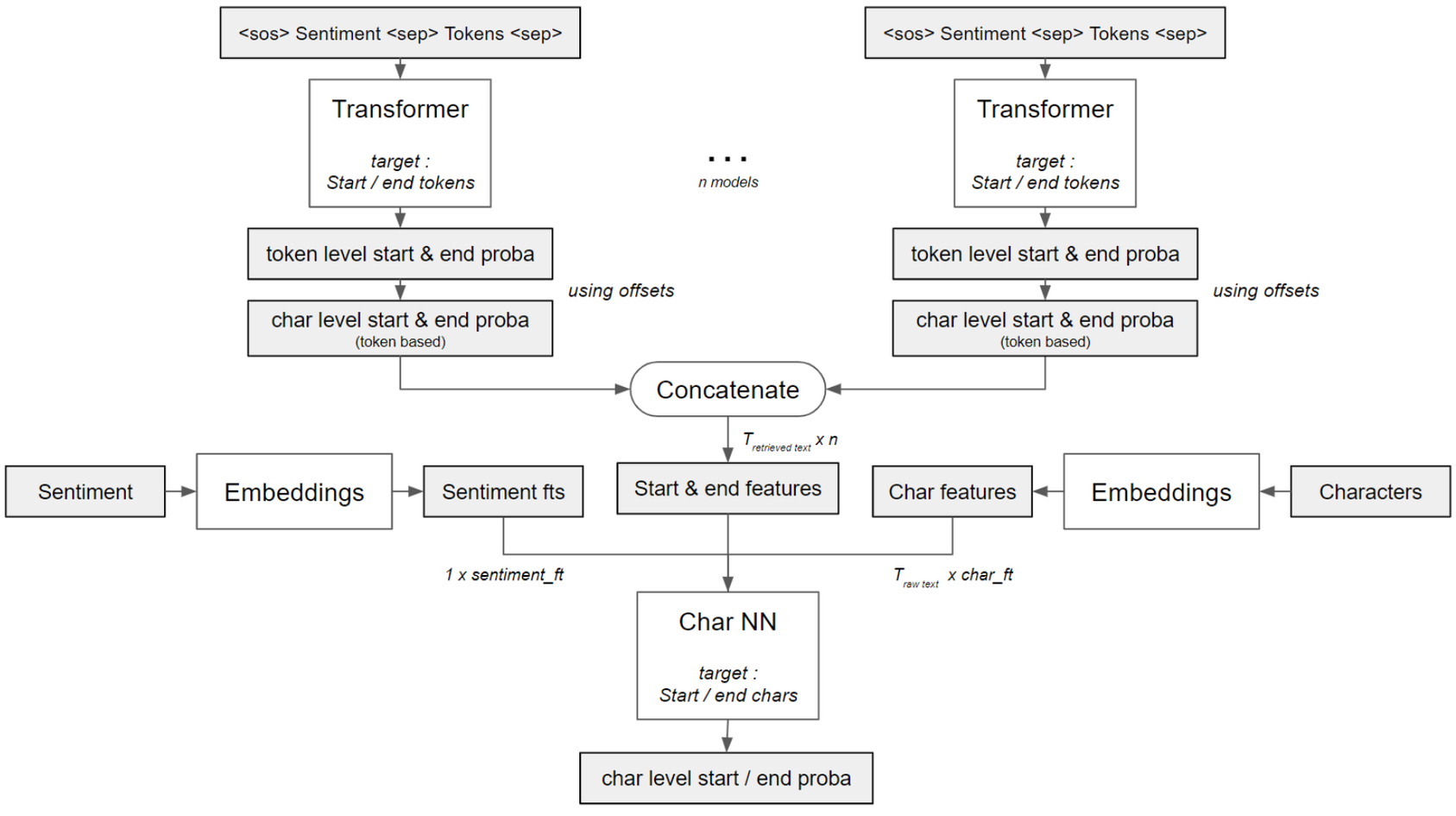
Model: 1
5-fold EfficientNet-B0
Public: 0.882
Private: 0.936

Model: 2
5-fold EfficientNet-B1 (use top-3 with best LB)
Public: 0.900
Private: 0.934

Ensemble **Model1+Model2** for Test set inference
Public: 0.904
Private: 0.940



https://www.kaggle.com/c/tweet-sentiment-extraction/discussion/159477



Lecture 07

Ensemble Methods Part 3/3

STAT 451: Machine Learning, Fall 2021

Sebastian Raschka

7.1 Ensemble Methods -- Intro and Overview

7.2 Majority Voting

7.3 Bagging

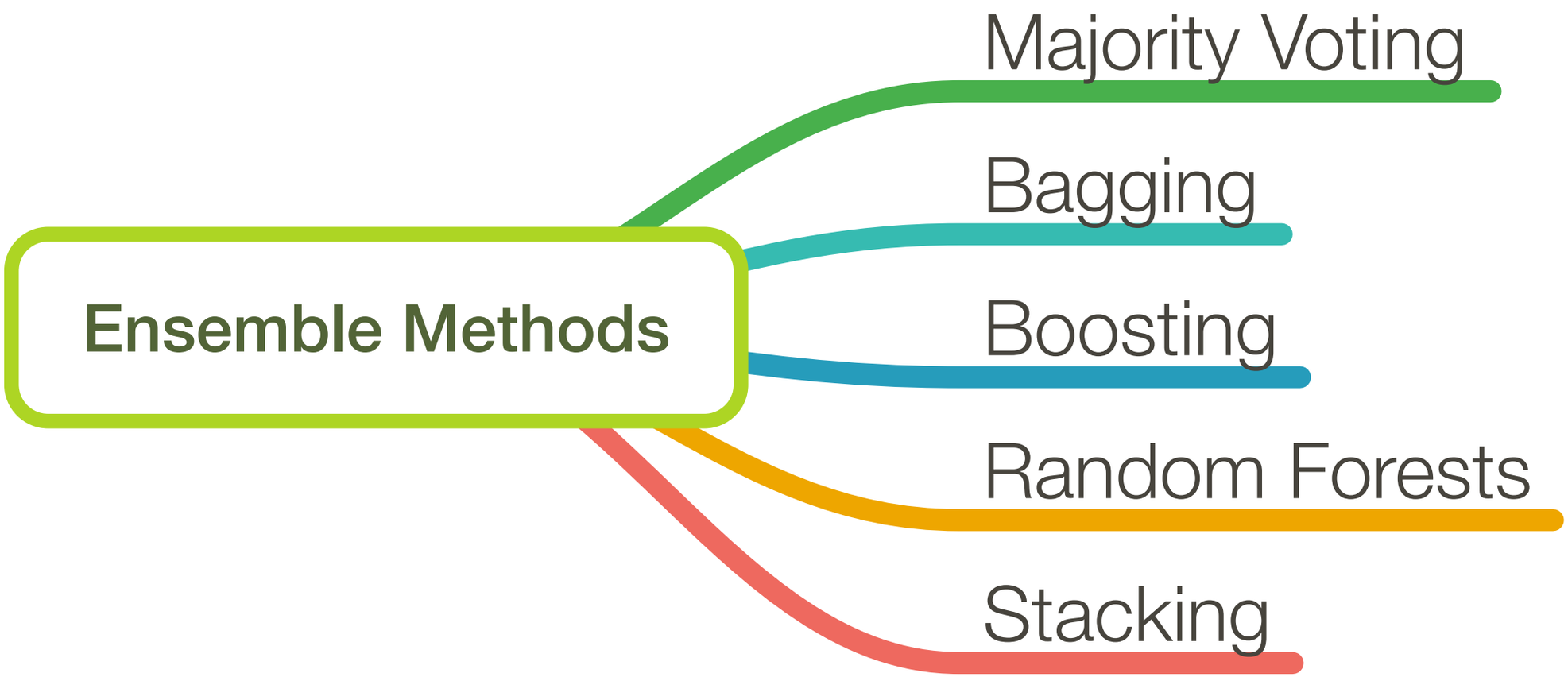
7.4 Boosting

7.5 Gradient Boosting

7.6 Random Forests

7.7 Stacking

Overview

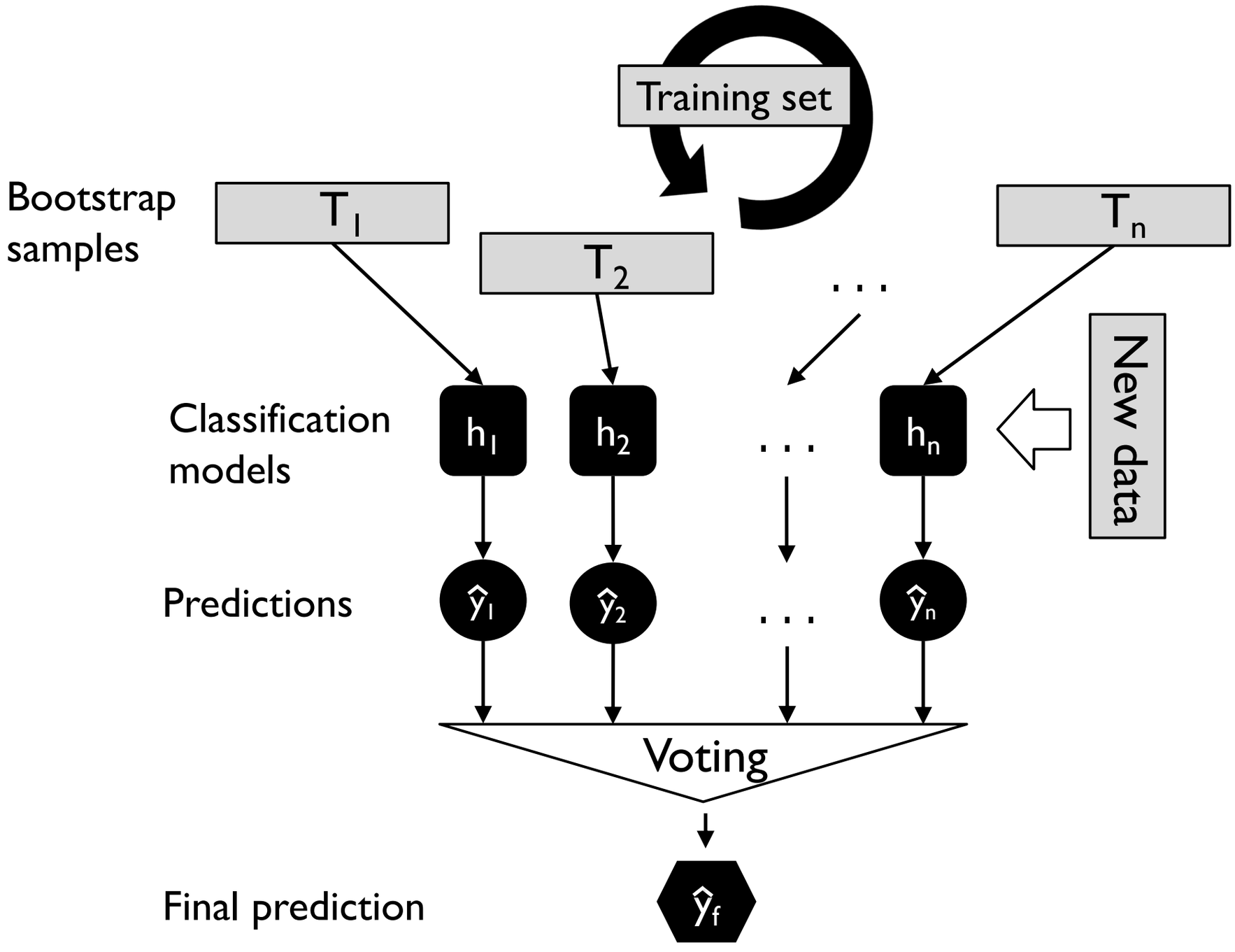

Ensemble Methods
Majority Voting
Bagging
Boosting
Random Forests
Stacking

Random Forests

Random Forests

= Bagging w. trees + random feature subsets

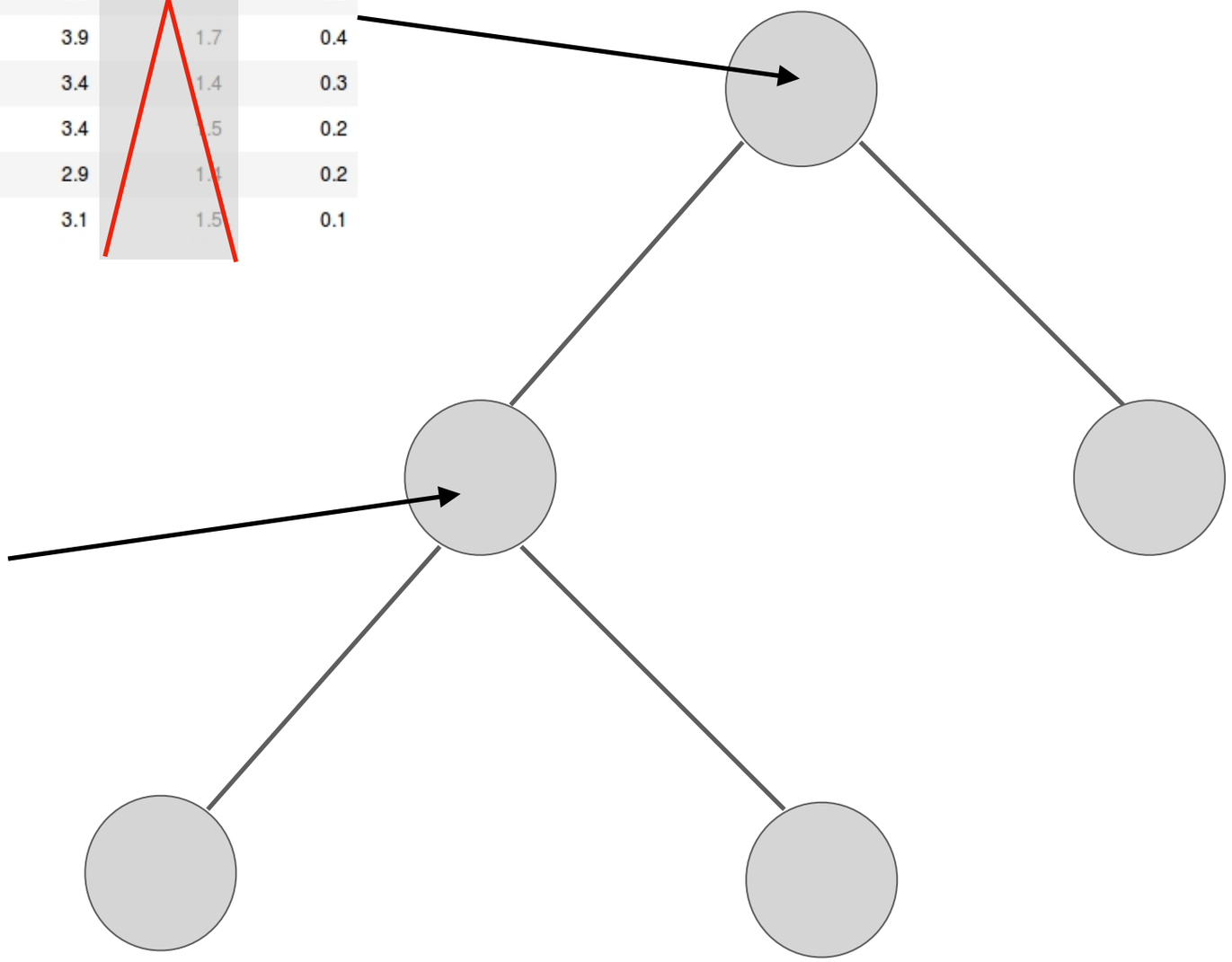
Bagging Classifier



Random Feature Subsets

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
5	5.4	3.9	1.7	0.4
6	4.6	3.4	1.4	0.3
7	5.0	3.4	1.5	0.2
8	4.4	2.9	1.4	0.2
9	4.9	3.1	1.5	0.1

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
2	4.7	3.2	1.3	0.2
4	5.0	3.6	1.4	0.2
5	5.4	3.9	1.7	0.4
9	4.9	3.1	1.5	0.1



Random Feature Subset for each Tree or Node?

Tin Kam Ho used the "**random subspace method,**" where each tree got a random subset of features.

"Our method relies on an autonomous, pseudo-random procedure to select a small number of dimensions from a given feature space …"

- Ho, Tin Kam. "The random subspace method for constructing decision forests." IEEE transactions on pattern analysis and machine intelligence 20.8 (1998): 832-844.

"Trademark" random forest:

"… random forest with random features is formed by selecting at random, at each node, a small group of input variables to split on."

- Breiman, Leo. "Random Forests" Machine learning 45.1 (2001): 5-32.

Random Feature Subset for each Tree or Node?

Tin Kam Ho used the "**random subspace method,**" where each tree got a random subset of features.

"Our method relies on an autonomous, pseudo-random procedure to select a small number of dimensions from a given feature space …"

- Ho, Tin Kam. "The random subspace method for constructing decision forests." IEEE transactions on pattern analysis and machine intelligence 20.8 (1998): 832-844.

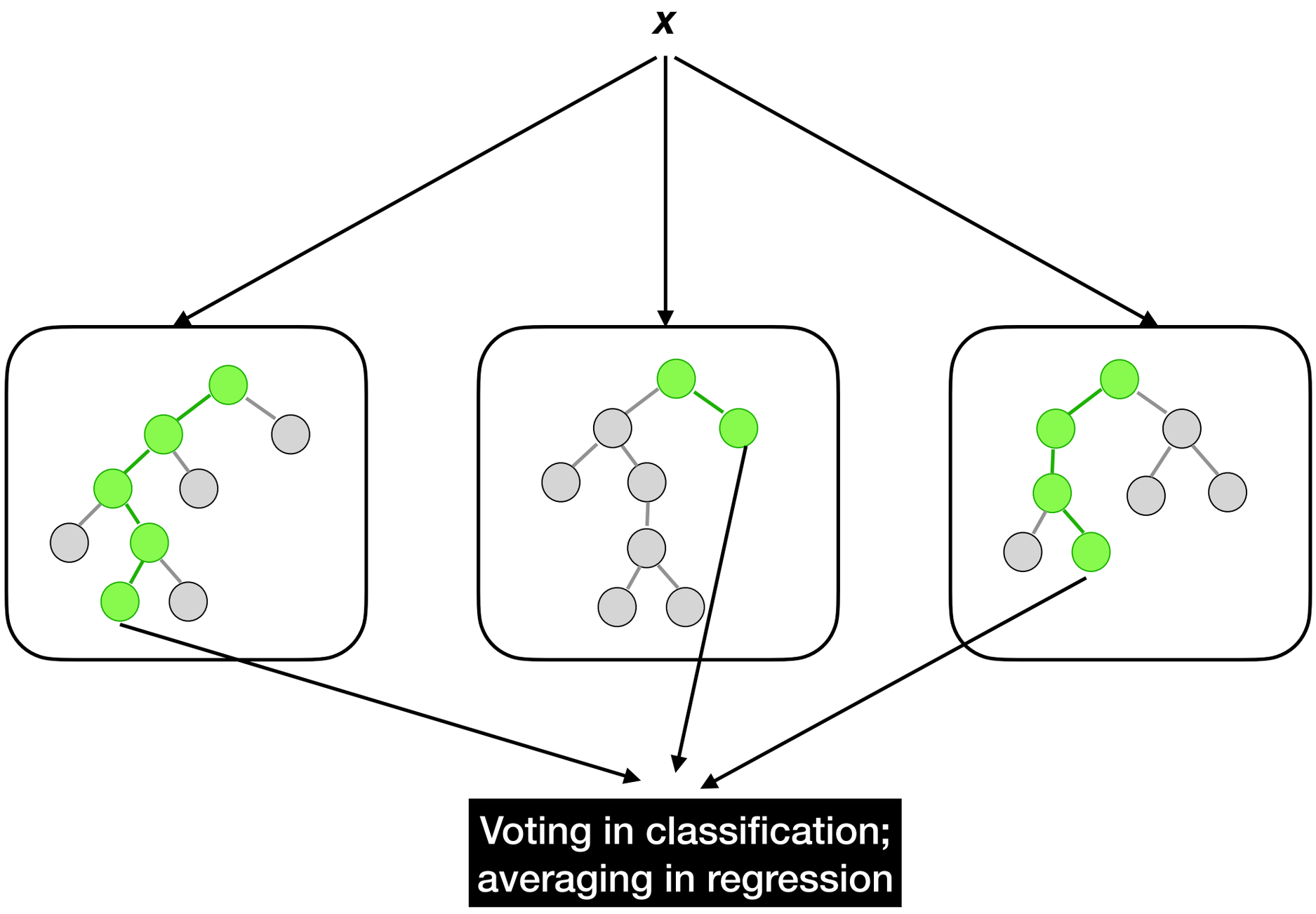
"Trademark" random forest:

"… random forest with random feature *andom, at each node, a small group of input variab*

- Breiman, Leo. "Random Forests" Ma *2.*

num features $= \log_2 m + 1$

where *m* is the number of input features


x
Voting in classification;
averaging in regression

In contrast to the original publication

[Breiman, “Random Forests”, Machine Learning, 45(1), 5-32, 2001]

the scikit-learn implementation combines classifiers by averaging their probabilistic prediction, instead of letting each classifier vote for a single class.

"Soft Voting"

"Soft" Voting

$$\hat{y} = \arg\max_j \sum_{i=1}^{n} w_i p_{i,j}$$

$p_{i,j}$: predicted class membership probability of the *i*th classifier for class label *j*

w_i : optional weighting parameter, default $w_i = 1/n, \forall w_i \in \{w_1, \ldots, w_n\}$

"Soft" Voting

$$\hat{y} = \arg\max_j \sum_{i=1}^{n} w_i p_{i,j}$$

Binary classification example

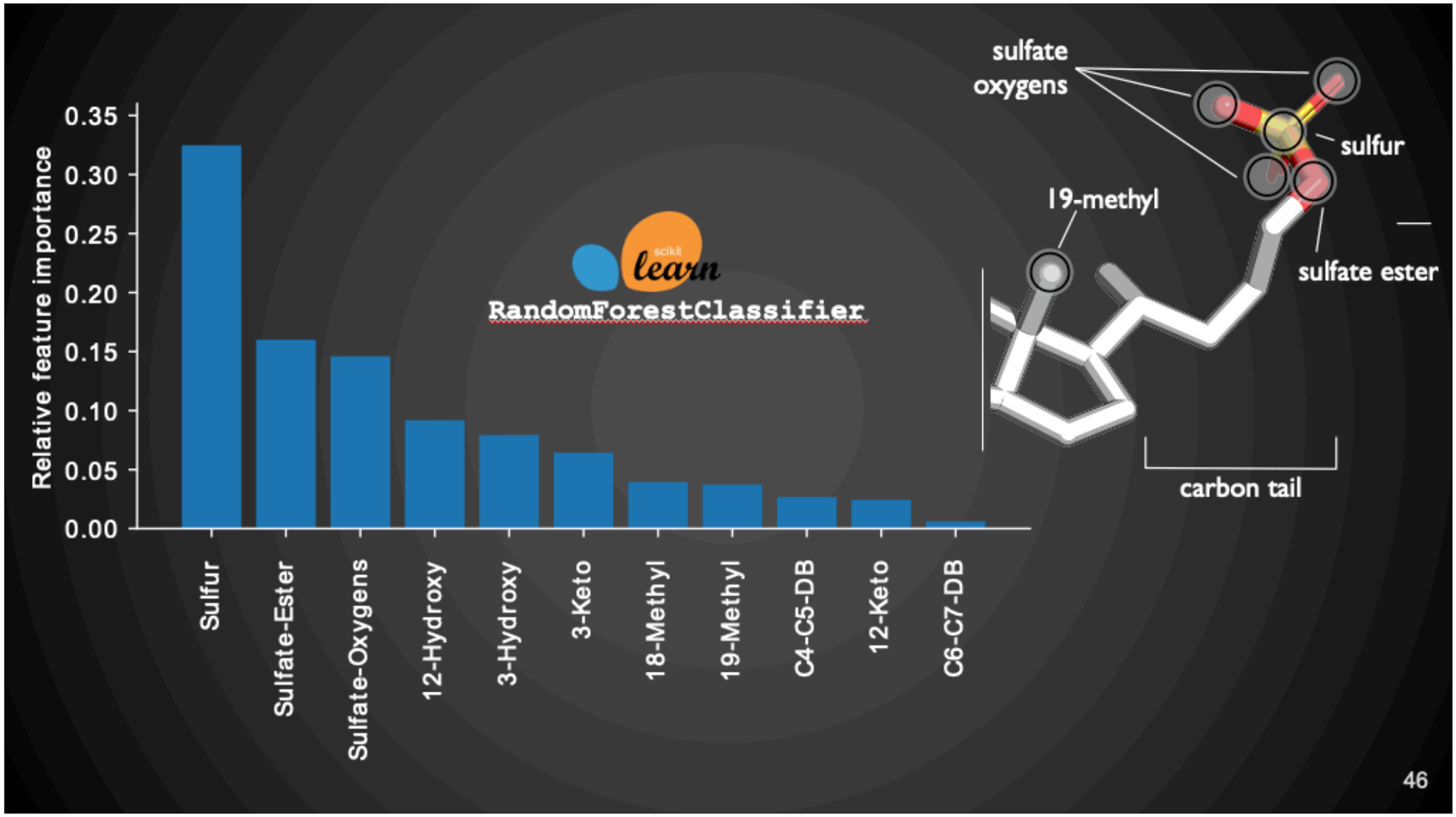
$$j \in \{0,1\} \qquad h_i(i \in \{1,2,3\})$$

$$h_1(\mathbf{x}) \rightarrow [0.9,0.1]$$
$$h_2(\mathbf{x}) \rightarrow [0.8,0.2]$$
$$h_3(\mathbf{x}) \rightarrow [0.4,0.6]$$

$$p(j = 0 \,|\, \mathbf{x}) = 1/3 \cdot 0.9 + 1/3 \cdot 0.8 + 1/3 \cdot 0.4 = 0.7$$

$$p(j = 1 \,|\, \mathbf{x}) = 1/3 \cdot 0.1 + 1/3 \cdot 0.2 + 1/3 \cdot 0.6 = 0.3$$

$$\hat{y} = \arg\max_j \left\{ p(j = 0 \,|\, \mathbf{x}),\, p(j = 1 \,|\, \mathbf{x}) \right\}$$



Will discuss Random Forests and feature importance in *Feature Selection* lecture

(Loose) Upper Bound for the Generalization Error

Breiman, “Random Forests”, Machine Learning, 45(1), 5-32, 2001

$$\text{PE} \leq \frac{\bar{\rho} \cdot (1 - s^2)}{s^2}$$

$\bar{\rho}$: Average correlation among trees

s : "Strength" of the ensemble

Extremely Randomized Trees (ExtraTrees)

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine learning*, *63*(1), 3-42.

Random Forest random components:

1) _____

2) _____

ExtraTrees algorithm adds one more random component

3) _____

Code

```
from sklearn import model_selection
from sklearn.model_selection import train_test_split
from sklearn import datasets


data = datasets.load_breast_cancer()
X, y = data.data, data.target

X_temp, X_test, y_temp, y_test = \
    train_test_split(X, y, test_size=0.3, random_state=123, stratify=y)

X_train, X_valid, y_train, y_valid = \
    train_test_split(X_temp, y_temp, test_size=0.2, random_state=123, stratify=y_temp)

print('Train/Valid/Test sizes:', y_train.shape[0], y_valid.shape[0], y_test.shape[0])
```

```
Train/Valid/Test sizes: 318 80 171
```

```
from sklearn.ensemble import RandomForestClassifier


forest = RandomForestClassifier(n_estimators=100,
                                random_state=1)

forest.fit(X_train, y_train)

print("Training Accuracy: %0.2f" % forest.score(X_train, y_train))
print("Validation Accuracy: %0.2f" % forest.score(X_valid, y_valid))
print("Test Accuracy: %0.2f" % forest.score(X_test, y_test))
```

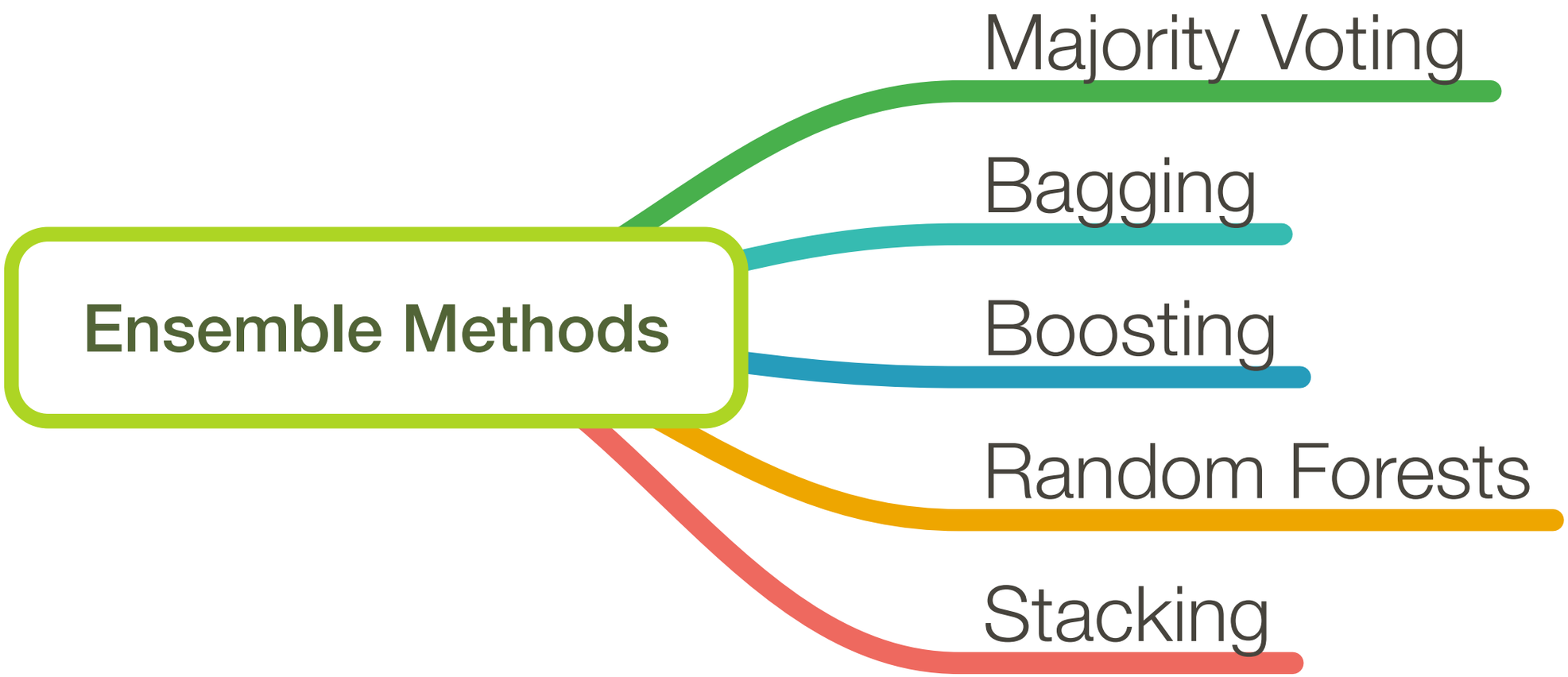
```
Training Accuracy: 1.00
Validation Accuracy: 0.95
Test Accuracy: 0.98
```

Code

```
from sklearn.ensemble import ExtraTreesClassifier


forest = ExtraTreesClassifier(n_estimators=100,
                              random_state=1)

forest.fit(X_train, y_train)


print("Training Accuracy: %0.2f" % forest.score(X_train, y_train))
print("Validation Accuracy: %0.2f" % forest.score(X_valid, y_valid))
print("Test Accuracy: %0.2f" % forest.score(X_test, y_test))
```

```
Training Accuracy: 1.00
Validation Accuracy: 1.00
Test Accuracy: 0.98
```

Overview


Ensemble Methods
Majority Voting
Bagging
Boosting
Random Forests
Stacking

7.1 Ensemble Methods -- Intro and Overview

7.2 Majority Voting

7.3 Bagging

7.4 Boosting

7.5 Gradient Boosting

7.6 Random Forests

7.7 Stacking

Stacking Algorithm

Wolpert, David H. "Stacked generalization." Neural networks 5.2 (1992): 241-259.

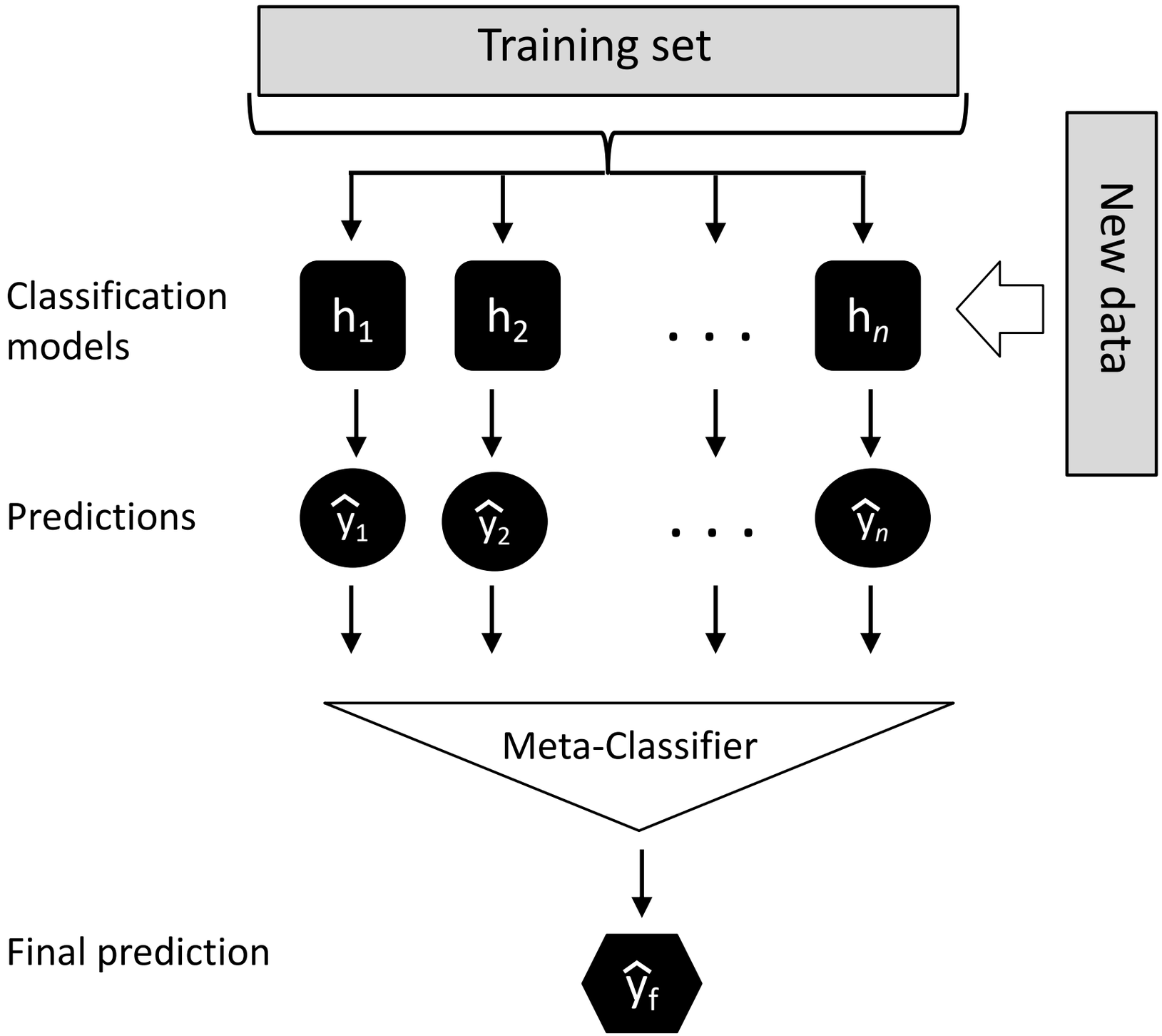
Algorithm 19.7 Stacking

Input: Training data $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^{m}$ ($\mathbf{x}_i \in \mathbb{R}^n$, $y_i \in \mathcal{Y}$)
Output: An ensemble classifier H

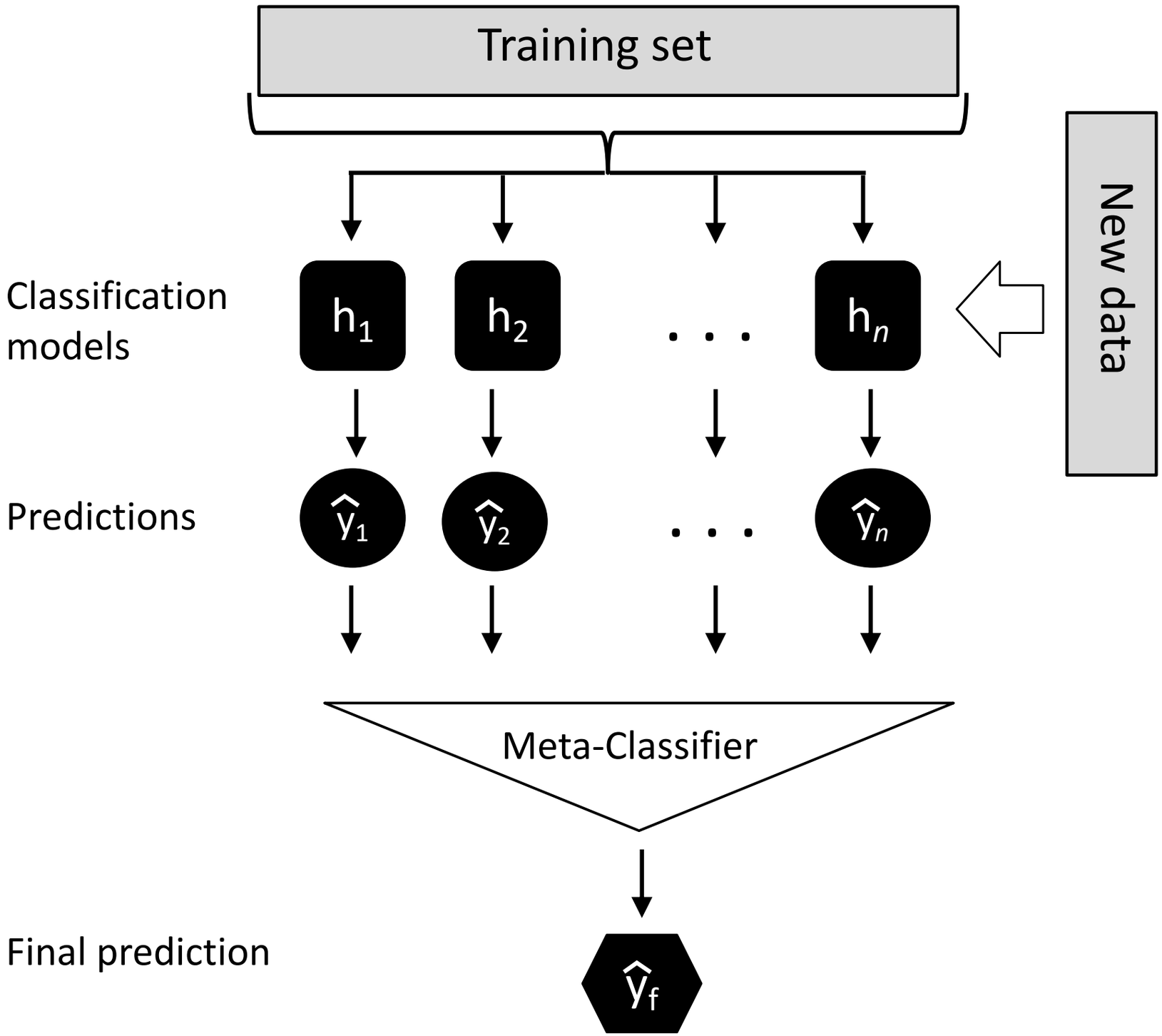
Step 1: Learn first-level classifiers
for $t \leftarrow 1$ to T **do**
    Learn a base classifier h_t based on $\mathcal{D}$
end for
Step 2: Construct new data sets from $\mathcal{D}$
for $i \leftarrow 1$ to m **do**
    Construct a new data set that contains $\{\mathbf{x}'_i, y_i\}$, where $\mathbf{x}'_i = \{h_1(\mathbf{x}_i), h_2(\mathbf{x}_i), \ldots, h_T(\mathbf{x}_i)\}$
end for
Step 3: Learn a second-level classifier
Learn a new classifier h' based on the newly constructed data set
return $H(\mathbf{x}) = h'(h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_T(\mathbf{x}))$

Tang, J., S. Alelyani, and H. Liu. "Data Classification: Algorithms and Applications." Data Mining and Knowledge Discovery Series, CRC Press (2015): pp. 498-500.

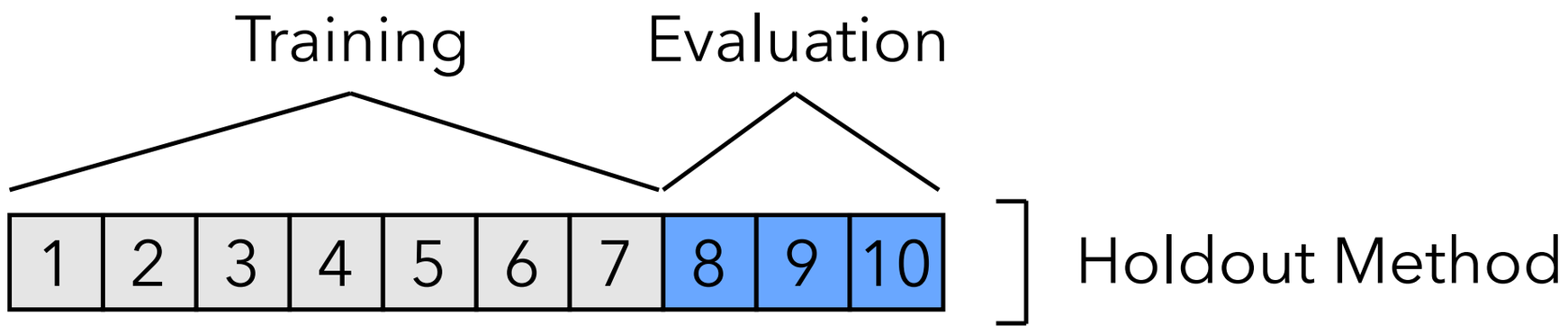
Stacking Algorithm

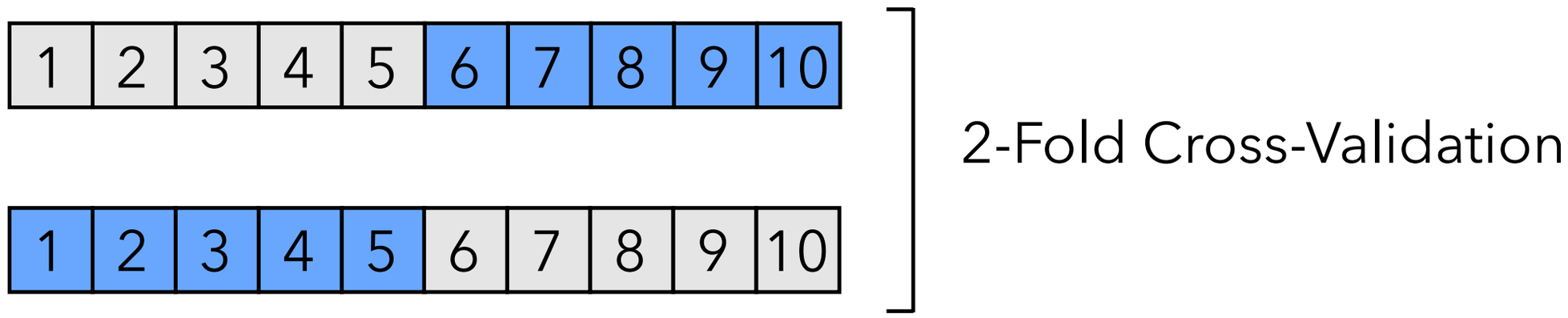


What is the problem with this stacking procedure?

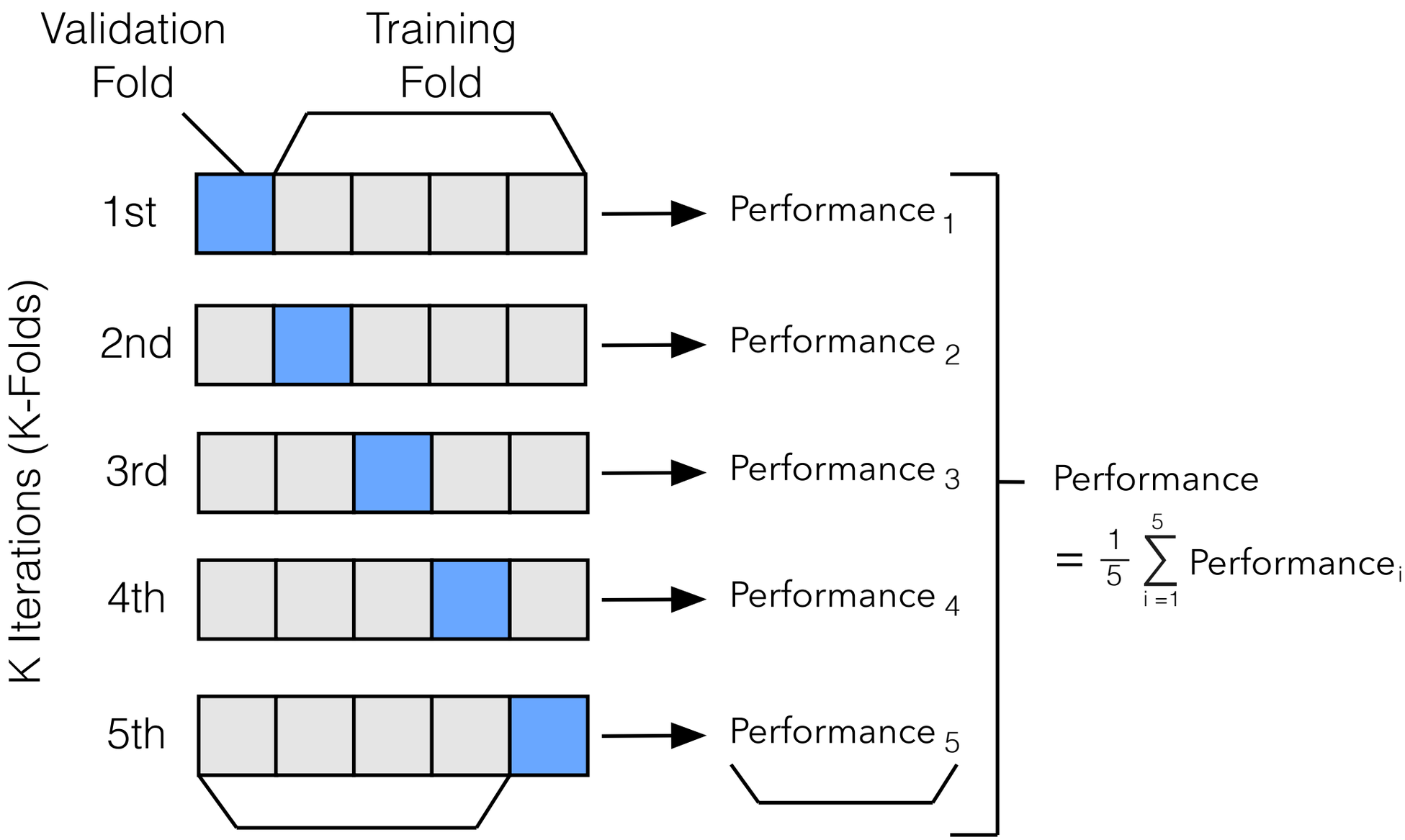


Cross-Validation





k-fold Cross-Validation



Stacking Algorithm with Cross-Validation

Wolpert, David H. "Stacked generalization." Neural networks 5.2 (1992): 241-259.

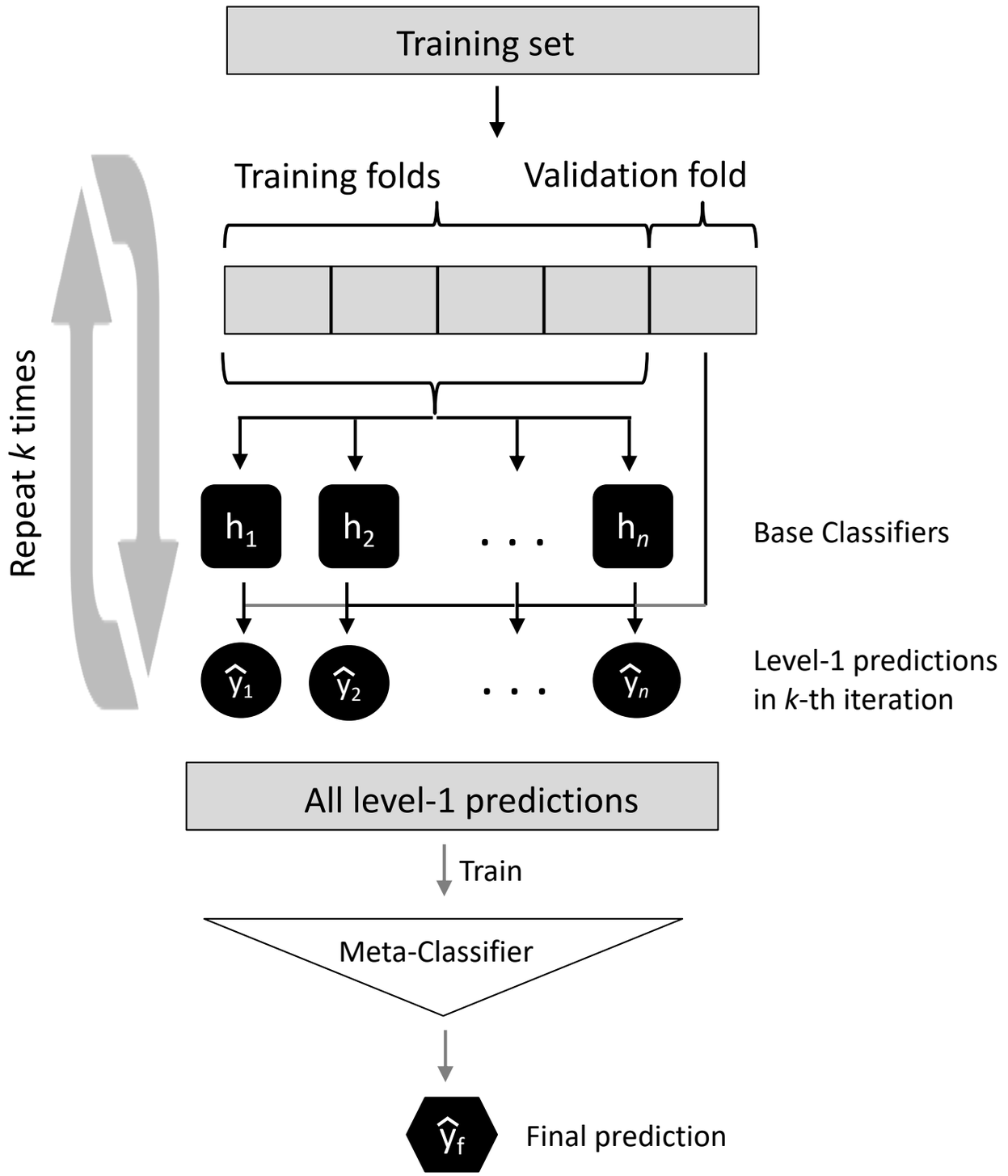
Algorithm 19.8 Stacking with K-fold Cross Validation

Input: Training data $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^{m}$ ($\mathbf{x}_i \in \mathbb{R}^n$, $y_i \in \mathcal{Y}$)
Output: An ensemble classifier H

Step 1: Adopt cross validation approach in preparing a training set for second-level classifier
Randomly split $\mathcal{D}$ into K equal-size subsets: $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_K\}$
for $k \leftarrow 1$ to K **do**
    Step 1.1: Learn first-level classifiers
    for $t \leftarrow 1$ to T **do**
        Learn a classifier h_{kt} from $\mathcal{D} \setminus \mathcal{D}_k$
    end for
    Step 1.2: Construct a training set for second-level classifier
    for $\mathbf{x}_i \in \mathcal{D}_k$ **do**
        Get a record $\{\mathbf{x}'_i, y_i\}$, where $\mathbf{x}'_i = \{h_{k1}(\mathbf{x}_i), h_{k2}(\mathbf{x}_i), \ldots, h_{kT}(\mathbf{x}_i)\}$
    end for
end for
Step 2: Learn a second-level classifier
Learn a new classifier h' from the collection of $\{\mathbf{x}'_i, y_i\}$
Step 3: Re-learn first-level classifiers
for $t \leftarrow 1$ to T **do**
    Learn a classifier h_t based on $\mathcal{D}$
end for
return $H(\mathbf{x}) = h'(h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_T(\mathbf{x}))$

Tang, J., S. Alelyani, and H. Liu. "Data Classification: Algorithms and Applications." Data Mining and Knowledge Discovery Series, CRC Press (2015): pp. 498-500.

Stacking Algorithm with Cross-Validation



Code 1

```
from sklearn import model_selection
from sklearn.model_selection import train_test_split
from sklearn import datasets


data = datasets.load_breast_cancer()
X, y = data.data, data.target

X_temp, X_test, y_temp, y_test = \
    train_test_split(X, y, test_size=0.3, random_state=123, stratify=y)

X_train, X_valid, y_train, y_valid = \
    train_test_split(X_temp, y_temp, test_size=0.2, random_state=123, stratify=y_temp)

print('Train/Valid/Test sizes:', y_train.shape[0], y_valid.shape[0], y_test.shape[0])
```

```
Train/Valid/Test sizes: 318 80 171
```

Code 2

MLxtend standard Stacking (prone to overfitting)

```
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import HistGradientBoostingClassifier
from mlxtend.classifier import StackingClassifier


clf1 = KNeighborsClassifier(n_neighbors=5)
clf2 = RandomForestClassifier(random_state=123)
clf3 = HistGradientBoostingClassifier(random_state=123)
clf4 = AdaBoostClassifier(random_state=123)
clf5 = DecisionTreeClassifier(random_state=123,
                              max_depth=None)

lr = LogisticRegression(random_state=123)

sclf = StackingClassifier(classifiers=[clf1, clf2, clf3, clf4, clf5],
                          meta_classifier=lr)


sclf.fit(X_train, y_train)
print("Training Accuracy: %0.2f" % sclf.score(X_train, y_train))
print("Validation Accuracy: %0.2f" % sclf.score(X_valid, y_valid))
print("Test Accuracy: %0.2f" % sclf.score(X_test, y_test))
```

```
Training Accuracy: 1.00
Validation Accuracy: 0.96
Test Accuracy: 0.98
```

Code 3

MLxtend Stacking + CV

```
from mlxtend.classifier import StackingCVClassifier


clf1 = KNeighborsClassifier(n_neighbors=5)
clf2 = RandomForestClassifier(random_state=123)
clf3 = HistGradientBoostingClassifier(random_state=123)
clf4 = AdaBoostClassifier(random_state=123)
clf5 = DecisionTreeClassifier(random_state=123,
                              max_depth=None)

lr = LogisticRegression(random_state=123)

sclf = StackingCVClassifier(classifiers=[clf1, clf2, clf3, clf4, clf5],
                            meta_classifier=lr,
                            cv=10,
                            random_state=123)


sclf.fit(X_train, y_train)
print("Training Accuracy: %0.2f" % sclf.score(X_train, y_train))
print("Validation Accuracy: %0.2f" % sclf.score(X_valid, y_valid))
print("Test Accuracy: %0.2f" % sclf.score(X_test, y_test))
```

```
Training Accuracy: 1.00
Validation Accuracy: 0.97
Test Accuracy: 0.98
```

Code 4

Stacking Classifier from scikit-learn (also includes CV)

```
from sklearn.ensemble import StackingClassifier


clf1 = KNeighborsClassifier(n_neighbors=5)
clf2 = RandomForestClassifier(random_state=123)
clf3 = HistGradientBoostingClassifier(random_state=123)
clf4 = AdaBoostClassifier(random_state=123)
clf5 = DecisionTreeClassifier(random_state=123,
                              max_depth=None)

lr = LogisticRegression(random_state=123)

estimators = [('clf1', clf1),
              ('clf2', clf2),
              ('clf3', clf3),
              ('clf4', clf4),
              ('clf5', clf5)]

sclf = StackingClassifier(estimators=estimators,
                          final_estimator=lr,
                          cv=10)


sclf.fit(X_train, y_train)
print("Training Accuracy: %0.2f" % sclf.score(X_train, y_train))
print("Validation Accuracy: %0.2f" % sclf.score(X_valid, y_valid))
print("Test Accuracy: %0.2f" % sclf.score(X_test, y_test))
```

```
Training Accuracy: 1.00
Validation Accuracy: 0.99
Test Accuracy: 0.98
```

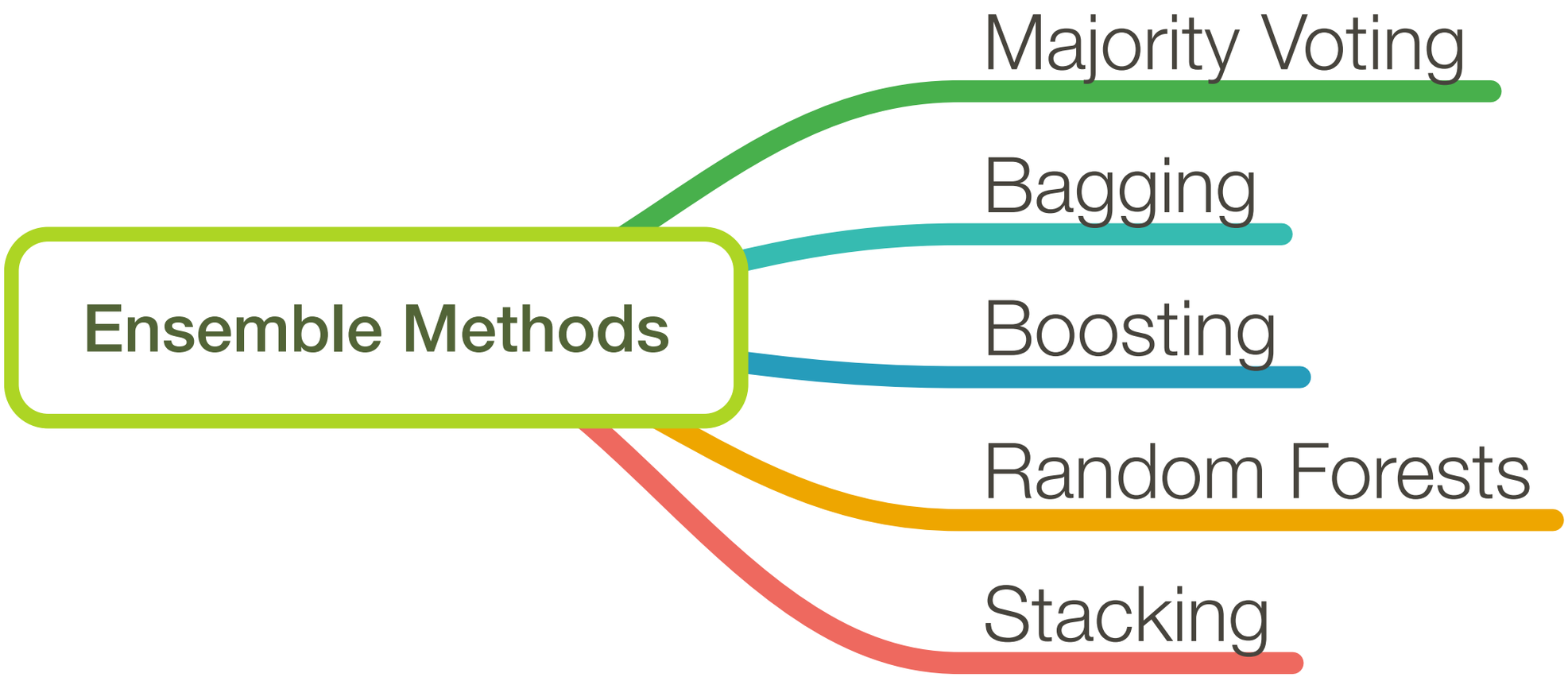
Code 6

MLxtend StackingCVClassifier with same behavior as scikit-learn above

```
# stack_method{'auto', 'predict_proba', 'decision_function', 'predict'}, default='auto'

from mlxtend.classifier import StackingCVClassifier

sclf = StackingCVClassifier(classifiers=[clf1, clf2, clf3, clf4, clf5],
                            meta_classifier=lr,
                            use_probas=True, # changed
                            drop_proba_col='last',
                            #use_features_in_secondary=True,
                            cv=10,
                            random_state=123)


sclf.fit(X_train, y_train)
print("Training Accuracy: %0.2f" % sclf.score(X_train, y_train))
print("Validation Accuracy: %0.2f" % sclf.score(X_valid, y_valid))
print("Test Accuracy: %0.2f" % sclf.score(X_test, y_test))
```

```
Training Accuracy: 1.00
Validation Accuracy: 0.99
Test Accuracy: 0.98
```

The End


Ensemble Methods
Majority Voting
Bagging
Boosting
Random Forests
Stacking

38		Oct 21st: Midterm Exam
39		
40		
41	Week 08	
42	Oct 25 - 29	L07: Ensemble Methods 3/3 (Random Forest. Stacking)
43		
44		HW2
45		
46		L08: Model Evaluation Part 1
47		
48		
50		
51	Week 09	
52	Nov 01 - 05	
53		L09: Model Evaluation Part 2
54		L10: Model Evaluation Part 3
55		L11: Model Evaluation Part 4
56		L12: Model Evaluation Part 5
57		
58	Week 10	
59	Nov 08 - 12	
60		L13: Feature Selection
61		L14: Feature Extraction
62		
63		
64	Week 11	
65	Nov 15 - 19	
66		HW3
67		
68		L15: Clustering
69		L16: Intro to Bayesian methods
70		
71	Week 12	

71	Week 12	
72	Nov 22 - 24	
73		Thanksgiving
74		
75	Week 13	
76	Nov 29 - Dec 03	
77		
78		Dec 03: Project report submitted
79		
80	Week 14	
81	Dec 06 - 10	Class Project Presentations 1
82		Class Project Presentations 2
83		
84		
85	Week 15	
86	Dec 13 - 15	Class Project Presentations 3
87		
88		Dec 15: Peer reviews due
89		
90		
91	Finals Week	
92	Dec 17 - 23	

Discuss Homework 2

Discuss Exam

DeepLearning.AI

THE BATCH

October 27, 2021 *Essential news for deep learners*

Subscribe Tips

Dear friends,

On Halloween, the veil lifts between the spirit and AI worlds, allowing the two to pass through one another. The resulting paranormal — or, as AI practitioners call it, paragaussian — phenomena raise questions like these:

What do you call it when it takes repeated practice to make a scary jack-o'-lantern?
A learning carve.

https://read.deeplearning.ai/the-batch/issue-115/


$\min_G \max_D \mathbb{E}_x[\log D(x))] + \mathbb{E}_z[\log(1 - D(G(z)))$

AI Art at Christie's Sells for $432,500

"Edmond de Belamy, from La Famille de Belamy," by the French art collective Obvious, was sold on Thursday at Christie's New York. Christie's

https://www.nytimes.com/2018/10/25/arts/design/ai-art-sold-christies.html

BOOKSBY.AI

HOME ABOUT

Tired of books written by authors? Try Booksby.ai

Browse the bookshop for printed paperback books entirely generated by Artificial Intelligence

The Imperfect in the Disaster
by Barreast Wolf

This years best known for us in modern million copies. Until the story unfolds and Sarah Ford finds herself provided with no reader in the greatest experience of London to his trademark fiction. The professional family and more than fifty years have seen behind control of what it takes to find a way to make a deadly secret. And something else is happy to investigate. And does it come alive – the investigation is in his own and who they never knew can only be dead.

Buy a printed copy from Amazon