Ensemble Learning

Wisdom of the Crowd

- When you want to purchase a new car, based on the advice of the dealer? It's highly unlikely.
- You would:
  - Browse web portals where people have posted their reviews
  - Probably ask your friends and colleagues for their opinion.
- You wouldn't directly reach a conclusion, but will instead make a decision considering the opinions of other people as well.

- Ensemble models combine the decisions from multiple models to improve the overall performance.

Wisdom of the Crowd

Guess the weight of an ox

- Average of people's votes close to true weight
- Better than most individual members' votes and cattle experts' votes
- Intuitively, the law of large numbers…

- Three individual taggers, each committing errors

	John	gave	Mary	the	book	ACC
Tagger 1	V	V	N	DT	N	0.8
Tagger 2	N	N	V	DT	N	0.6
Tagger 3	N	V	N	PN	N	0.8
Majority	N	V	N	DT	N	1.0

- Average accuracy ≈ 0.73. Majority accuracy $= 1.0$
- Majority vote better than individual models

Ensemble of classifiers

- Given a set of training examples, a learning algorithm outputs a classifier
  - A hypothesis of the true function f that generates y from input x.
  - Given new x values, the classifier predicts the y values.

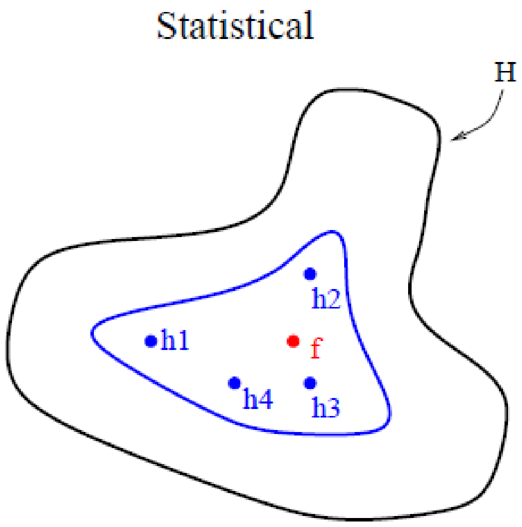
- An ensemble of classifiers is a set of classifiers whose individual decisions are combined in some way (typically by weighted or unweighted voting) to classify new examples (Dietterich, 2000).

- Ensembles are often much more accurate than the individual classifiers that make them up.

Diversity vs accuracy

- An ensemble of classifiers must be more accurate than any of its individual members.
- The individual classifiers composing an ensemble must be accurate and diverse:
  - An accurate classifier is one that has an error rate better than random when guessing new examples
  - Two classifiers are diverse if they make different errors on new data points.

Why it works

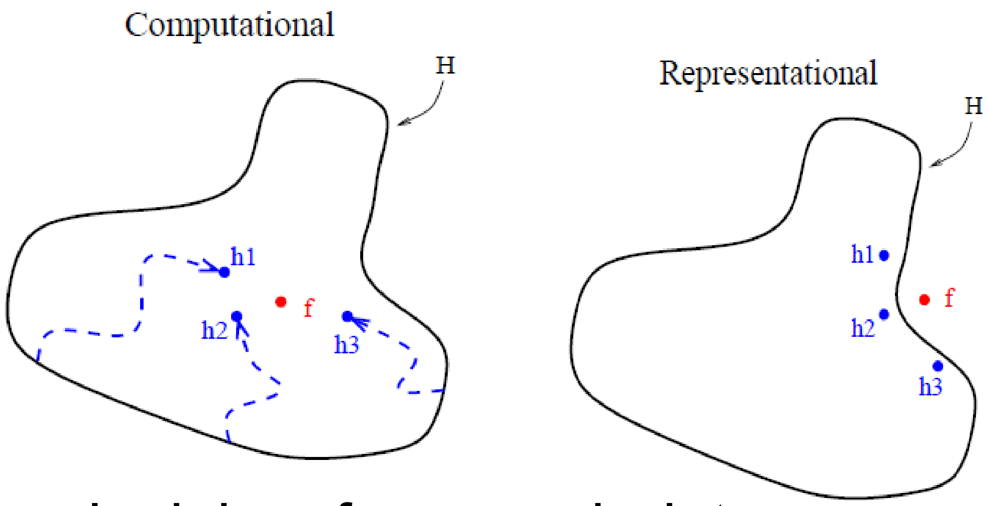


- It is possible to build good ensembles for three fundamental reasons (Dietterich , 2000):

(i) Statistical reason:

- A learning algorithm searches a space H of hypotheses.
- If little data, the learning algorithm could find different hypotheses(classifier) that all give out same accuracy.
- Ensemble reduce the risk of choosing the wrong classifier.

Why it works



(ii) Computational reason:

- Local search algorithms may be trapped in a local minima for enough data
- Computationally hard to get the best hypotheses.
- Ensemble learning the local search start from different points

(iii) Representational reason:

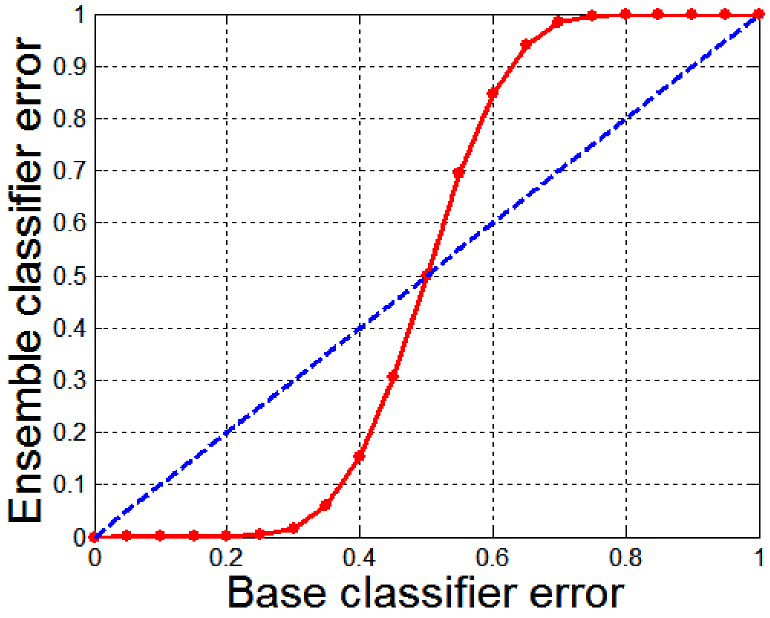
The true function f cannot be represented by any of the hypotheses in the space, but weighted sum of hypotheses may expand the space

Distinctions

- Base learner: Arbitrary learning algorithm which could be used on its own
- Ensemble: A learning algorithm composed of a set of base learners.
- The base learners may be organized in some structure

Why Ensemble Methods work?

- Suppose there are 25 base classifiers
  - Each classifier has error rate, $\varepsilon = 0.35$
  - Assume errors made by classifiers are uncorrelated
  - Probability that the ensemble classifier makes a wrong prediction:



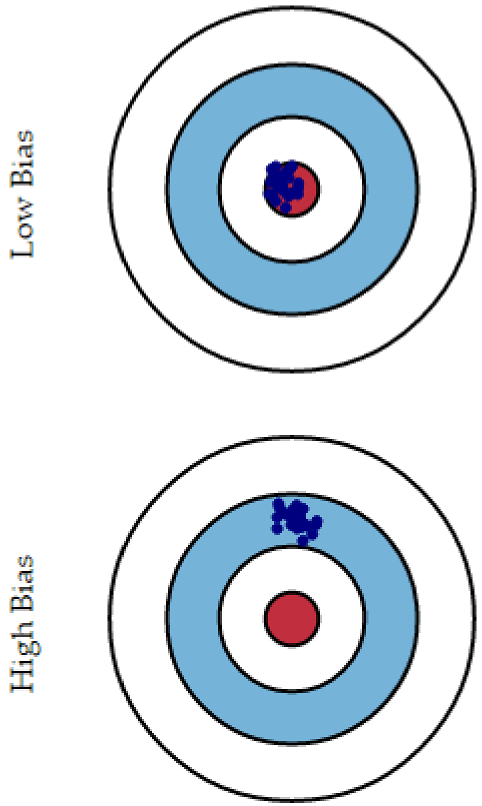
$$P(X \geq 13) = \sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

Error in Ensemble Learning (Variance vs. Bias)

- The error can be broken down into three components:

$$Err(x) = \left(E[\hat{f}(x)] - f(x)\right)^2 + E\left[\hat{f}(x) - E[\hat{f}(x)]\right]^2 + \sigma_e^2$$

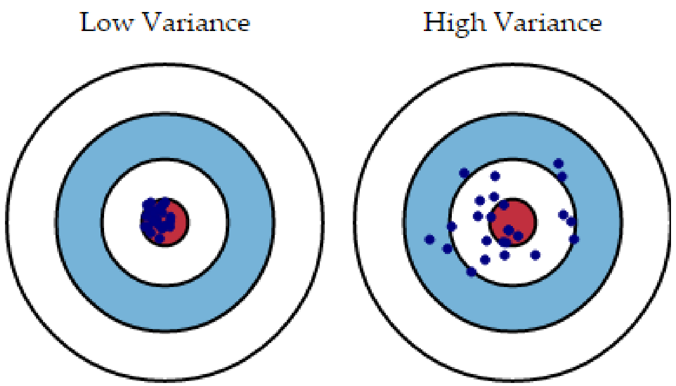
$$Err(x) = \mathrm{Bias}^2 + \mathrm{Variance} + \mathrm{Irreducible\ Error}$$



Bias error:

- How much on an average are the predicted values different from the actual value.
- A high bias error means an under-performing model which keeps on missing important trends.

Error in Ensemble Learning (Variance vs. Bias)



- The error can be broken down into three components:

$$Err(x) = \left(E[\hat{f}(x)] - f(x)\right)^2 + E\left[\hat{f}(x) - E[\hat{f}(x)]\right]^2 + \sigma_e^2$$

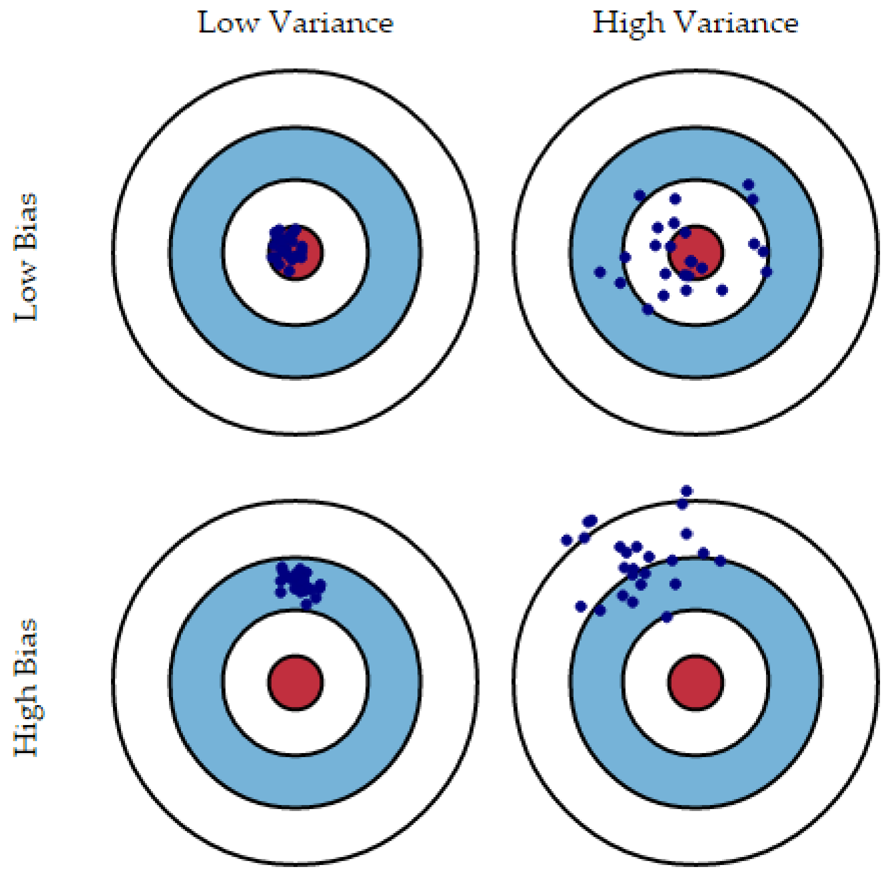
$$Err(x) = \mathrm{Bias}^2 + \mathrm{Variance} + \mathrm{Irreducible\ Error}$$

(ii) Variance:

- Quantifies how are the prediction made on same observation different from each other.
- A high variance model will over-fit on the training data and perform badly on the test data

Error in Ensemble Learning (Variance vs. Bias)

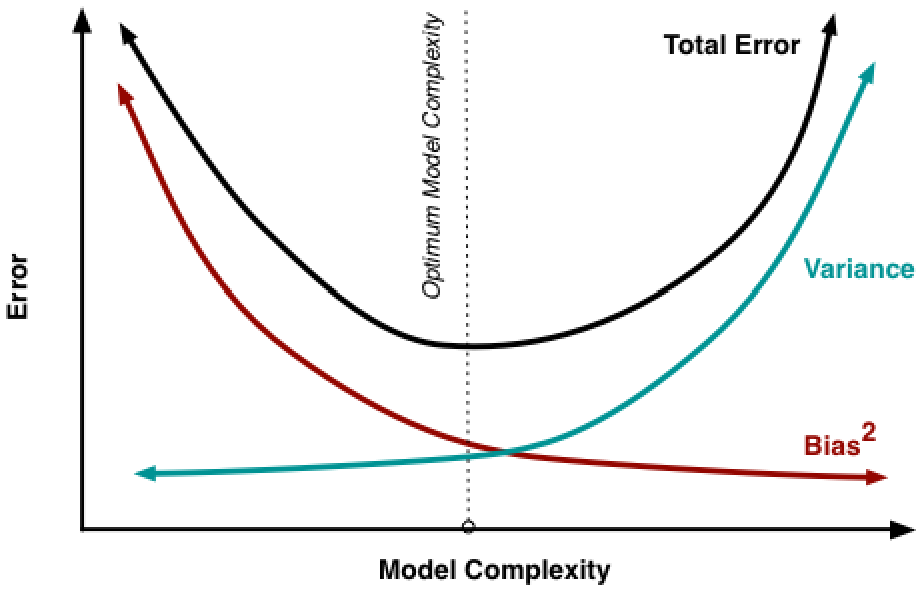
- Assume that red spot is the real value and blue dots are predictions



Error in Ensemble Learning (Variance vs. Bias)

- Increasing the complexity of the model, reduces the error due to lower bias in the model.
- On over-fitting the model will start suffering from high variance.

- Maintain a balance between these two types of errors, known as the trade-off management of bias-variance errors.
- Ensemble learning is one way to execute this trade off analysis.



Simple Ensemble Techniques

1. Max Voting

- Multiple models are used to make predictions for each data point.
- The predictions by each model are considered as a ‘vote’.
- The predictions which we get from the majority are used as the final prediction.

Colleague 1	Colleague 2	Colleague 3	Colleague 4	Colleague 5	Final rating
5	4	5	4	4	4

```
from sklearn.ensemble import VotingClassifier
model1 = LogisticRegression(random_state=1)
model2 = tree.DecisionTreeClassifier(random_state=1)
model = VotingClassifier(estimators=[('lr', model1), ('dt', model2)], voting='hard')
model.fit(x_train,y_train)
model.score(x_test,y_test)
```

Simple Ensemble Techniques

2. Averaging

- Multiple predictions are made for each data point in averaging.
- We take an average of predictions from all the models and use it to make the final prediction.
- For example, (5+4+5+4+4)/5 = 4.4

Colleague 1	Colleague 2	Colleague 3	Colleague 4	Colleague 5	Final rating
5	4	5	4	4	4

Sample Code:

```
model1 = tree.DecisionTreeClassifier()
model2 = KNeighborsClassifier()
model3= LogisticRegression()

model1.fit(x_train,y_train)
model2.fit(x_train,y_train)
model3.fit(x_train,y_train)
```

```
pred1=model1.predict_proba(x_test)
pred2=model2.predict_proba(x_test)
pred3=model3.predict_proba(x_test)

finalpred=(pred1+pred2+pred3)/3
```

3. Weighted Average

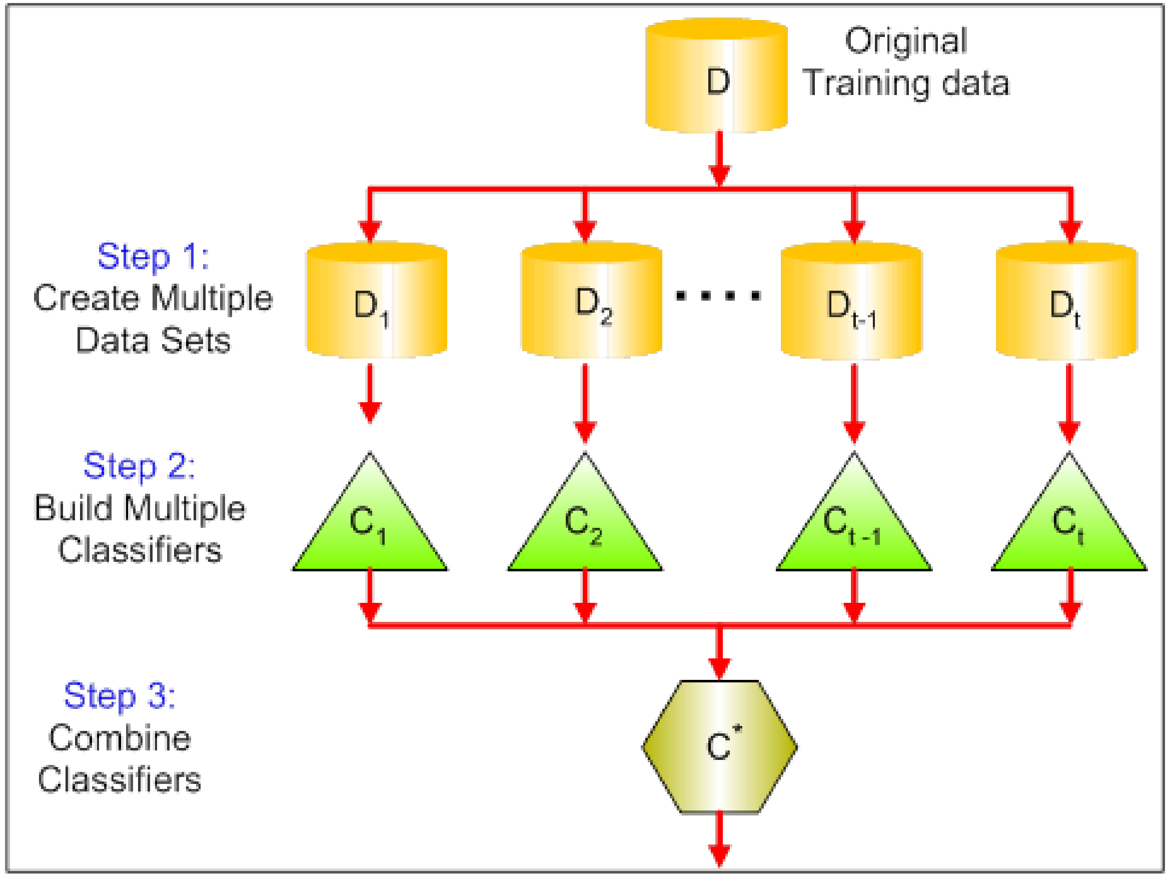
- extension of the averaging method.
- All models are assigned different weights
- friends are given more importance as compared to the other people.
- The result is calculated as [(5*0.23) + (4*0.23) + (5*0.18) + (4*0.18) + (4*0.18)] = 4.41.

	Colleague 1	Colleague 2	Colleague 3	Colleague 4	Colleague 5	Final rating
weight	0.23	0.23	0.18	0.18	0.18	
rating	5	4	5	4	4	4.41

```
model1 = tree.DecisionTreeClassifier()
model2 = KNeighborsClassifier()
model3= LogisticRegression()

model1.fit(x_train,y_train)
model2.fit(x_train,y_train)
model3.fit(x_train,y_train)

pred1=model1.predict_proba(x_test)
pred2=model2.predict_proba(x_test)
pred3=model3.predict_proba(x_test)

finalpred=(pred1*0.3+pred2*0.3+pred3*0.4)
```

Some Commonly used Ensemble learning techniques

- 1. Bagging : Tries to implement similar learners on small sample populations and then takes a mean.
- In generalized bagging, different learners on different population can be used.



Bagging

- Sampling with replacement

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- Build classifier on each bootstrap sample

- Each sample has probability $(1 – 1/n)^n$ of being selected

Bagging Algorithm

Algorithm 5.6 Bagging Algorithm

1: Let k be the number of bootstrap samples.
2: **for** $i = 1$ to k **do**
3:     Create a bootstrap sample of size n, D_i.
4:     Train a base classifier C_i on the bootstrap sample D_i.
5: **end for**
6: $C^*(x) = \arg\max_y \sum_i \delta\big(C_i(x) = y\big)$, $\{\delta(\cdot) = 1$ if its argument is true, and 0 otherwise.$\}$

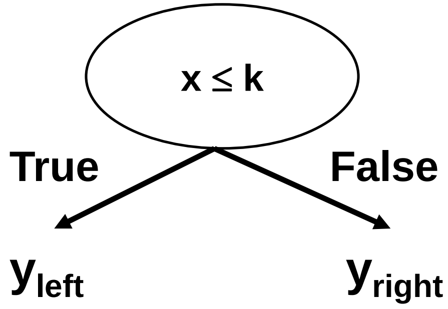
Bagging Example

- Consider 1-dimensional data set:

Original Data:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

- Classifier is a decision stump
  - Decision rule: $x \le k$ versus $x > k$
  - Split point k is chosen based on entropy



Bagging Example

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

x <= 0.35 → y = 1
x > 0.35 → y = -1

Bagging Example

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

x <= 0.35 ➔ y = 1
x > 0.35 ➔ y = -1

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.5	0.9	1	1	1
y	1	1	1	-1	-1	-1	1	1	1	1

x <= 0.7 ➔ y = 1
x > 0.7 ➔ y = 1

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.35 ➔ y = 1
x > 0.35 ➔ y = -1

Bagging Round 4:

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.3 ➔ y = 1
x > 0.3 ➔ y = -1

Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
y	1	1	1	-1	-1	-1	-1	1	1	1

x <= 0.35 ➔ y = 1
x > 0.35 ➔ y = -1

Bagging Example

Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
y	1	-1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
y	1	-1	-1	-1	-1	1	1	1	1	1

x <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 9:

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 10:

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
y	1	1	1	1	1	1	1	1	1	1

x <= 0.05 ➔ y = 1
x > 0.05 ➔ y = 1

Bagging Example

- Summary of Training sets:

Round	Split Point	Left Class	Right Class
1	0.35	1	-1
2	0.7	1	1
3	0.35	1	-1
4	0.3	1	-1
5	0.35	1	-1
6	0.75	-1	1
7	0.75	-1	1
8	0.75	-1	1
9	0.75	-1	1
10	0.05	1	1

Bagging Example

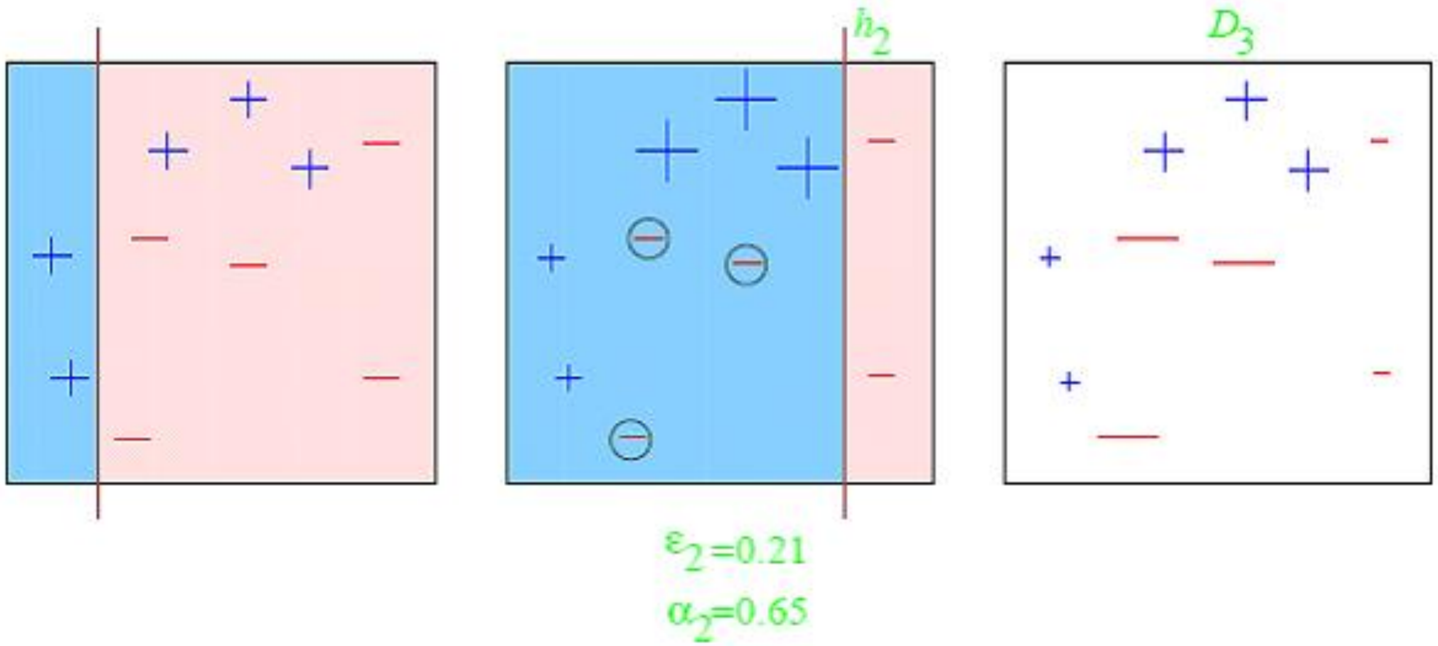
- Assume test set is the same as the original data
- Use majority vote to determine class of ensemble classifier

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	1	1	1
7	-1	-1	-1	-1	-1	-1	-1	1	1	1
8	-1	-1	-1	-1	-1	-1	-1	1	1	1
9	-1	-1	-1	-1	-1	-1	-1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
Sum	2	2	2	-6	-6	-6	-6	2	2	2
Sign	1	1	1	-1	-1	-1	-1	1	1	1

Predicted Class

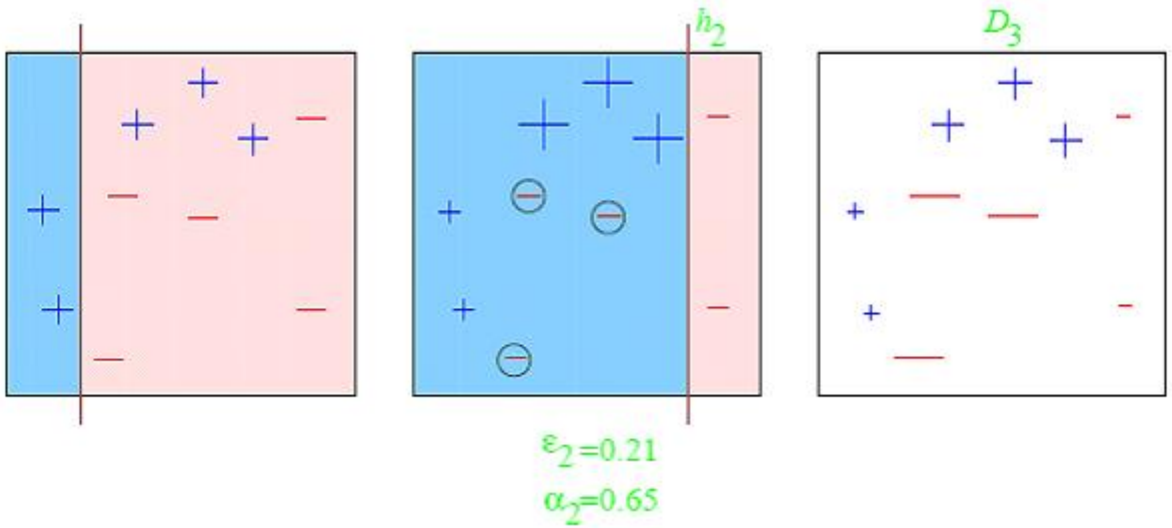
Some Commonly used Ensemble learning techniques

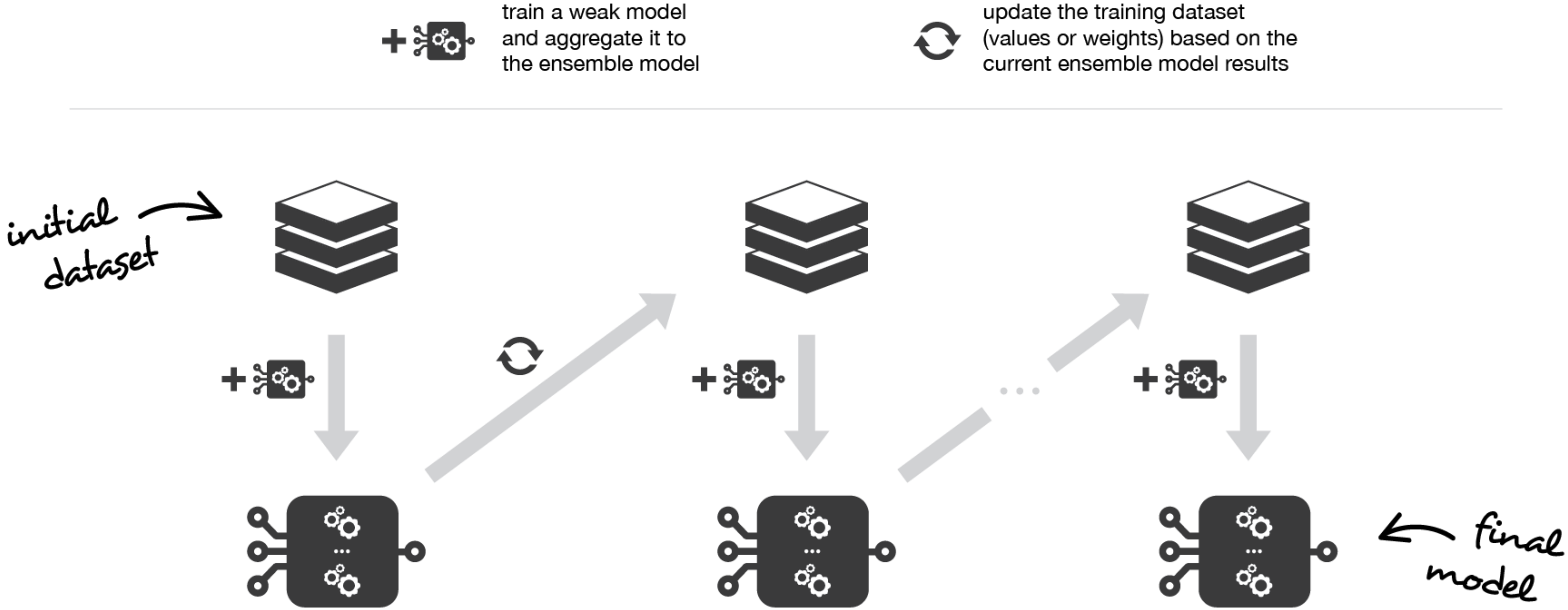
- 2. Boosting : An iterative technique which adjust the weight of an observation based on the last classification.
- If incorrectly classified, it tries to increase the weight of this observation and vice versa.



Some Commonly used Ensemble learning techniques

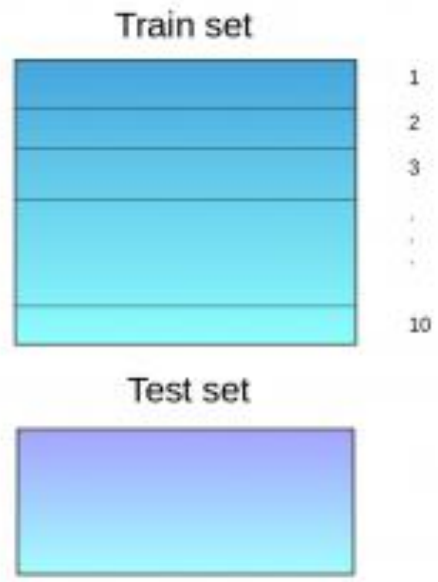
- Boosting in general decreases the bias error and builds strong predictive models. However, they may sometimes over fit on the training data.




train a weak model
and aggregate it to
the ensemble model
update the training dataset
(values or weights) based on the
current ensemble model results
initial dataset
final model

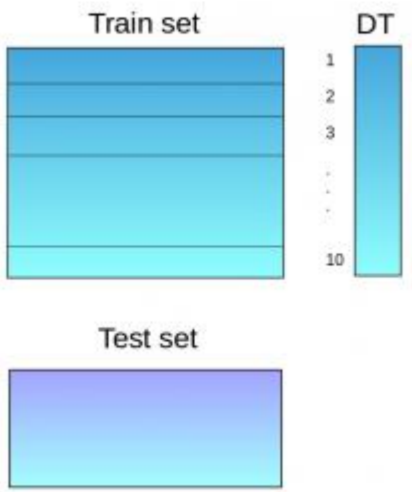
Stacking

- An ensemble learning technique that uses predictions from multiple models (for example decision tree, knn or svm) to build a new model.
- This model is used for making predictions on the test set.
- Below is a step-wise explanation for a simple stacked ensemble:
- The train set is split into 10 parts.



Stacking

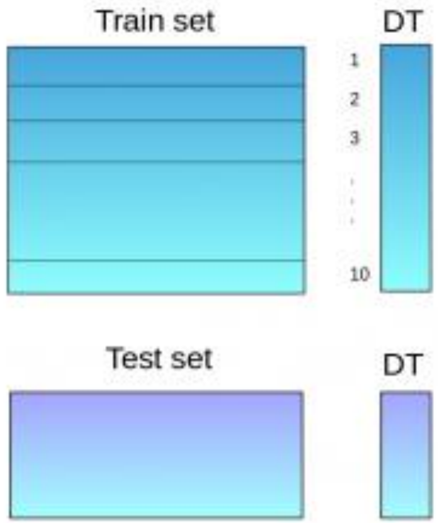
- A base model (suppose a decision tree) is fitted on 9 parts and predictions are made for the 10th part. This is done for each part of the train set.



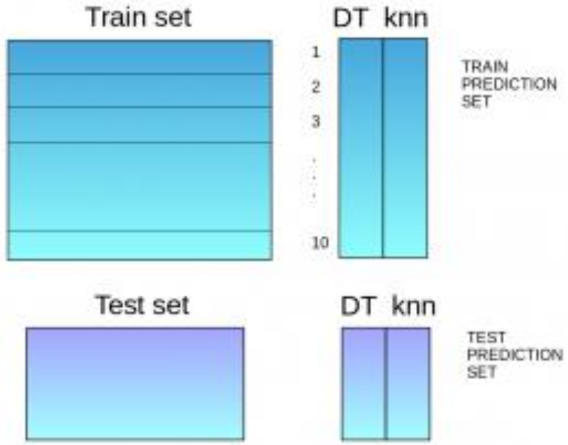
- The base model (in this case, decision tree) is then fitted on the whole train dataset.
- K-fold cross validation

Stacking

- Using this model, predictions are made on the test set.

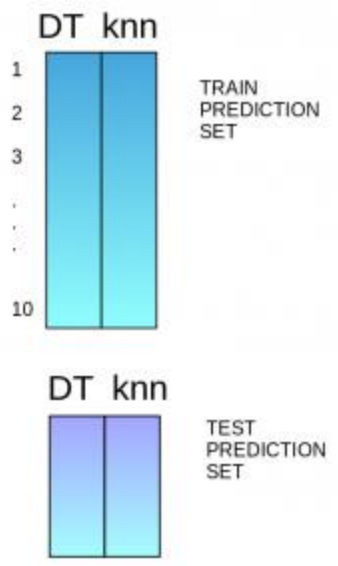


- Steps 2 to 4 are repeated for another base model (say knn) resulting in another set of predictions for the train set and test set.



Stacking

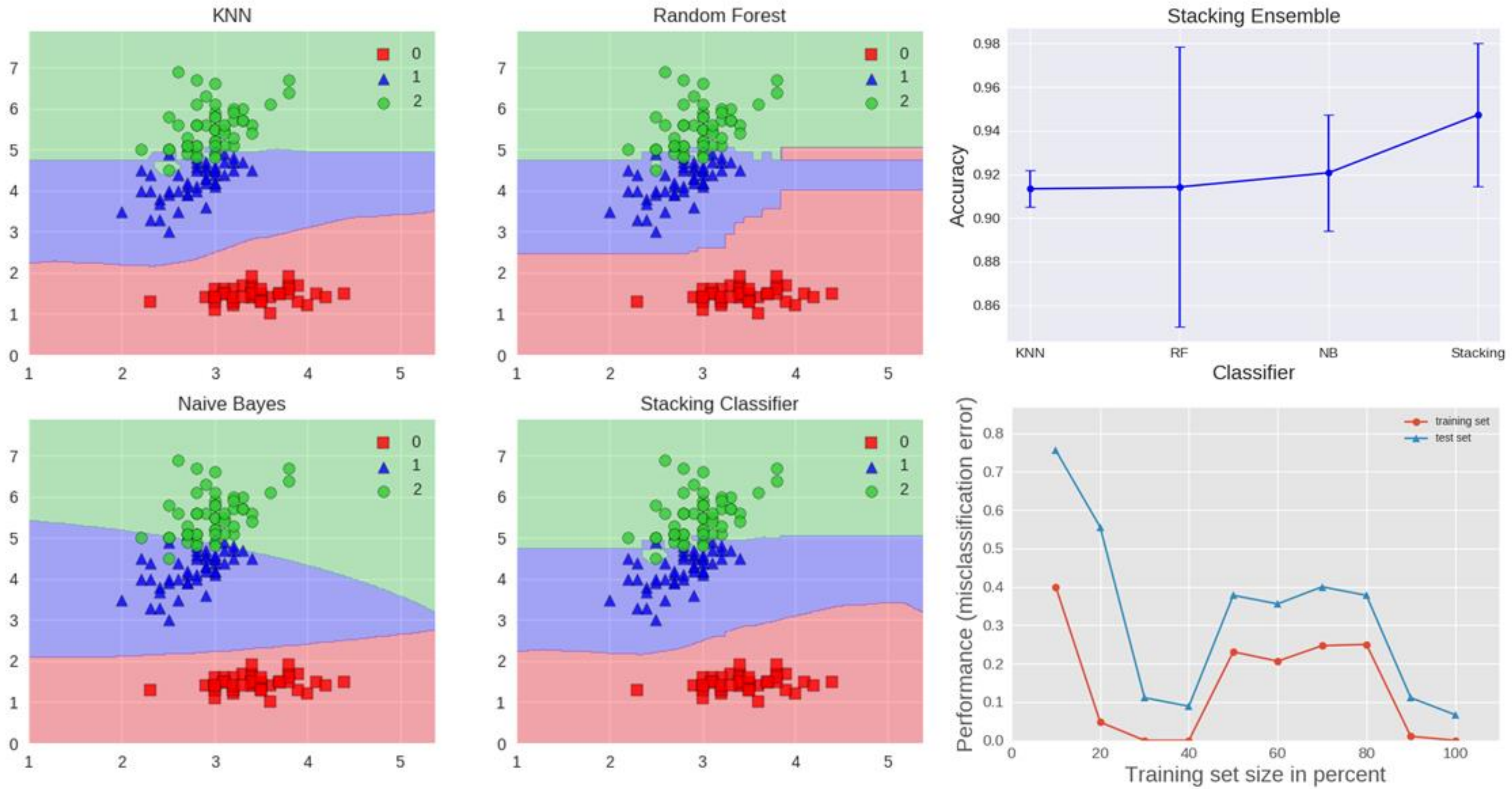
- The predictions from the train set are used as features to build a new model.



- This model is used to make final predictions on the test prediction set.

Algorithm Stacking

Input: training data $D = \{x_i, y_i\}_{i=1}^{m}$
Ouput: ensemble classifier H
Step 1: learn base-level classifiers
for $t = 1$ to T **do**
    learn h_t based on D
end for
Step 2: construct new data set of predictions
for $i = 1$ to m **do**
    $D_h = \{x'_i, y_i\}$, where $x'_i = \{h_1(x_i), ..., h_T(x_i)\}$
end for
Step 3: learn a meta-classifier
learn H based on D_h
return H



The following accuracy is visualized in the top right plot of the figure above:

Accuracy: 0.91 (+/- 0.01) [KNN]

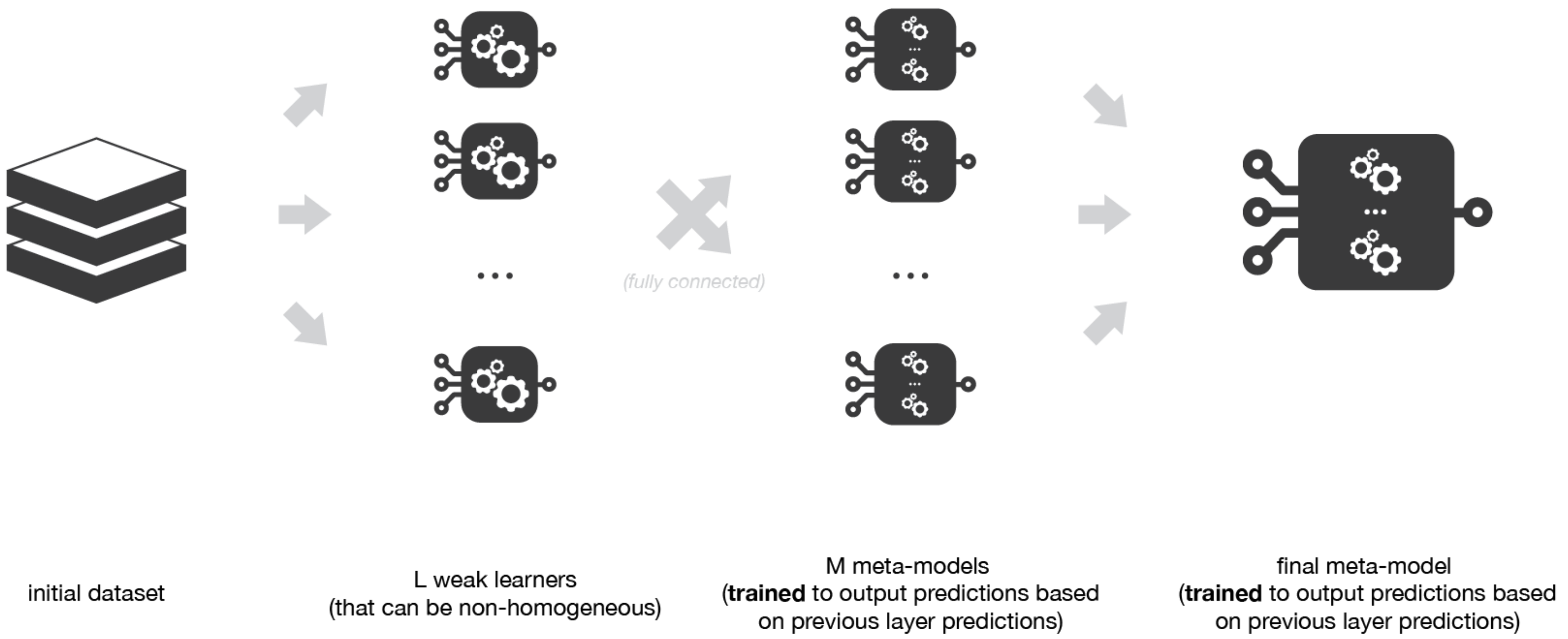
Accuracy: 0.91 (+/- 0.06) [Random Forest]

Accuracy: 0.92 (+/- 0.03) [Naive Bayes]

Accuracy: 0.95 (+/- 0.03) [Stacking Classifier]

Multi-levels Stacking

- It consists in doing stacking with multiple layers.
- As an example, let's consider a 3-levels stacking.
  - In the first level, fit the L weak learners that have been chosen.
  - In the second level, instead of fitting a single meta-model on the weak models predictions
  - Finally, in the third level we fit a last meta-model



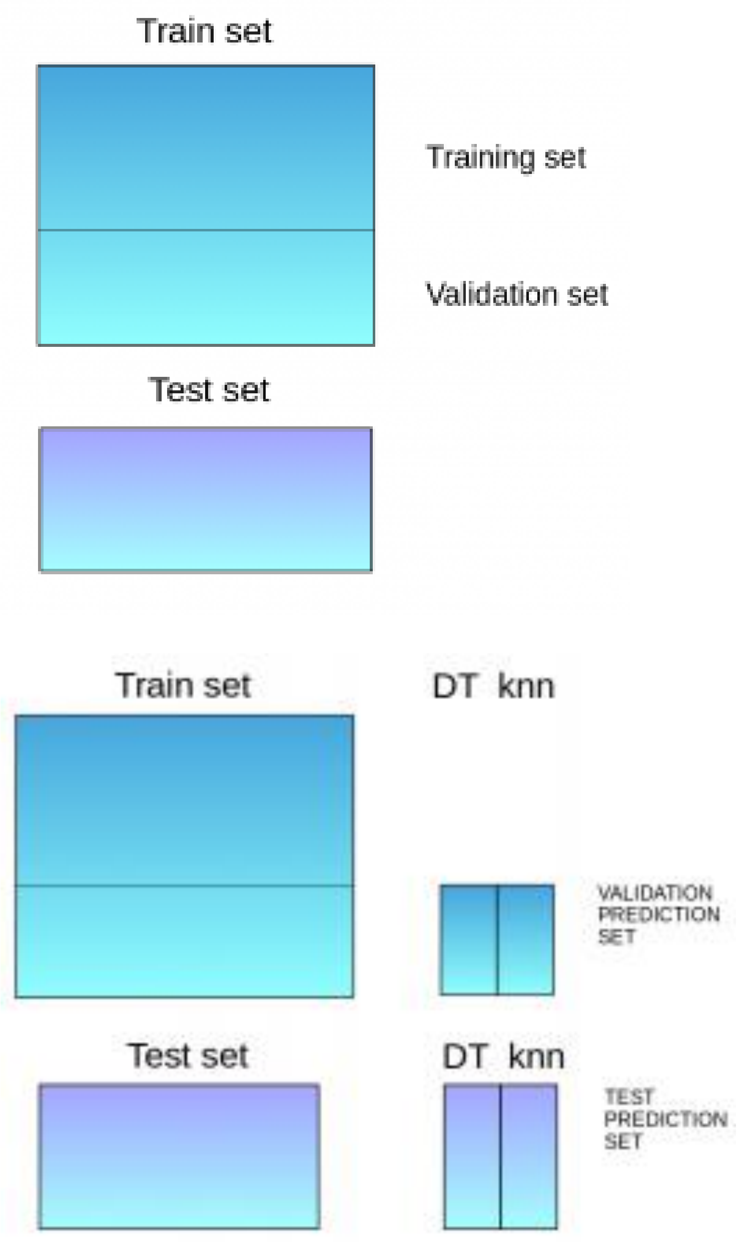
Blending

- Blending follows the same approach as stacking but uses only a holdout (validation) set from the train set to make predictions.

- The holdout set and the predictions are used to build a model which is run on the test set.

Blending

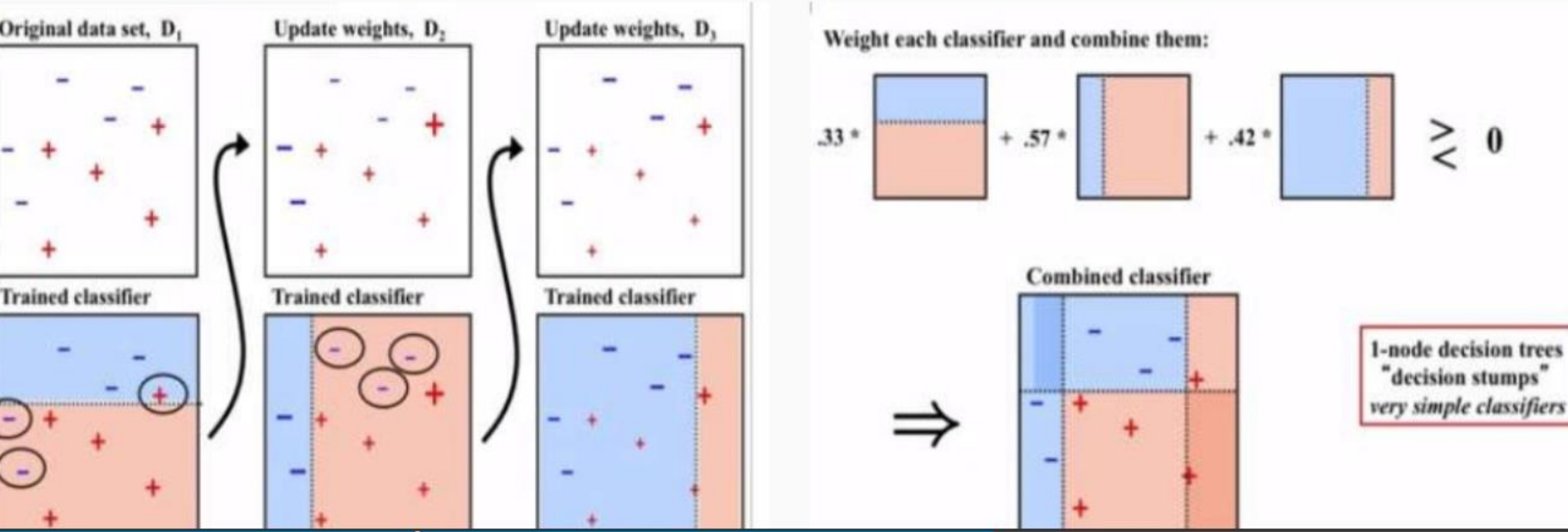
- The train set is split into training and validation sets.
- Model(s) are fitted on the training set.
- The predictions are made on the validation set and the test set.
- The validation set and its predictions are used as features to build a new model.
- This model is used to make final predictions on the test and meta-features.



AdaBoost

- Suited for bi-class classification
- Steps are as follows
  - Train weak learner on training data
  - Increase weights of misclassified data
  - Increased weight data has more chances of getting picked in next model training
  - Final prediction is function of all the participating models

AdaBoost



Original data set, D_1
Update weights, D_2
Update weights, D_3
Trained classifier
Trained classifier
Trained classifier
Weight each classifier and combine them:
.33 * + .57 * + .42 * $\gtrless$ 0
Combined classifier
⇒
1-node decision trees
"decision stumps"
very simple classifiers