Ensemble Learning

Nipun Batra and teaching staff

January 13, 2020

IIT Gandhinagar

Ensemble Methods

Use multiple models for prediction.
Most winning entries of Kaggle competition using ensemble learning.

Ensemble Methods

Use multiple models for prediction.
Most winning entries of Kaggle competition using ensemble learning.

Example:
Classifier 1 - Good
Classifier 2 - Good
Classifier 3 - Bad

Using Majority Voting, we predict Good.

Ensemble Methods

Use multiple models for prediction.
Most winning entries of Kaggle competition using ensemble learning.

Example:
Regressor 1 - 20
Regressor 2 - 30
Regressor 3 - 30

Using Average, we predict $\frac{80}{3}$

Intuition

Three reasons why ensembles make sense:

Intuition

Three reasons why ensembles make sense:

1) Statistical: Sometimes if data is less, many competing hypothesis can be learnt.

Intuition

Three reasons why ensembles make sense:

1) Statistical: Sometimes if data is less, many competing hypothesis can be learnt.

Eg. Depending on criteria and initialisation, we can learn many decision trees for the same data.

Intuition

Three reasons why ensembles make sense:

1) Statistical: Sometimes if data is less, many competing hypothesis can be learnt.

Eg. Depending on criteria and initialisation, we can learn many decision trees for the same data.

2) Computational: Some classifiers/regressors can get stuck in local optima. Computationally learning the “best” hypothesis can be non-trivial.

Intuition

Three reasons why ensembles make sense:

1) Statistical: Sometimes if data is less, many competing hypothesis can be learnt.

Eg. Depending on criteria and initialisation, we can learn many decision trees for the same data.

2) Computational: Some classifiers/regressors can get stuck in local optima. Computationally learning the “best” hypothesis can be non-trivial.

Eg. Decision Trees employ greedy critera

Intuition

Three reasons why ensembles make sense:

1) Statistical: Sometimes if data is less, many competing hypothesis can be learnt.

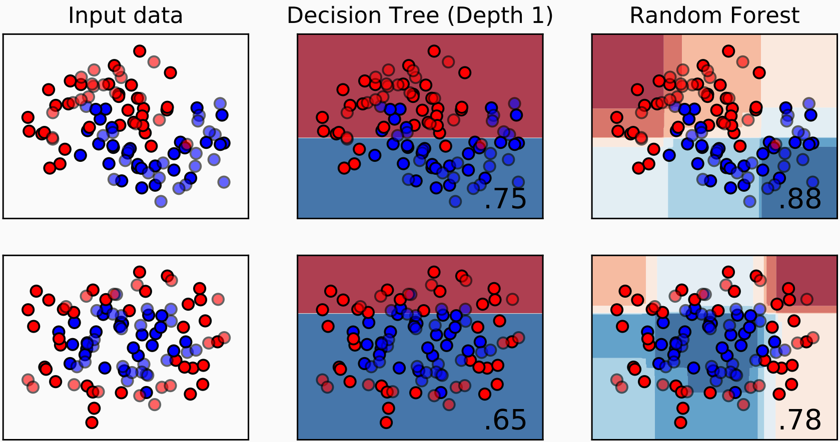
Eg. Depending on criteria and initialisation, we can learn many decision trees for the same data.

2) Computational: Some classifiers/regressors can get stuck in local optima. Computationally learning the "best" hypothesis can be non-trivial.

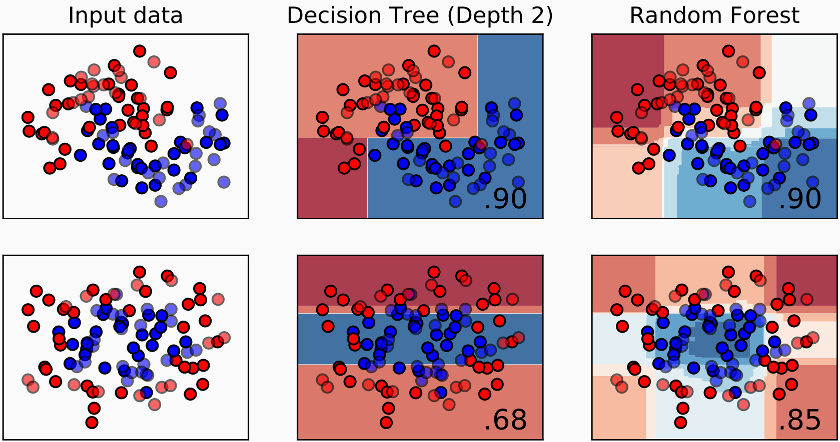
Eg. Decision Trees employ greedy critera

3) Representational: Some classifiers/regressors can not learn the true form/representation.

Representation of Limited Depth DTs vs RFs



Representation of Limited Depth DTs vs RFs



Necessary and Sufficient Conditions

Necessary and Sufficient Conditions

1)A necessary and sufficient condition for an ensemble of classiers to be more accurate than any of its individual members is if the classiers are accurate and diverse.

Necessary and Sufficient Conditions

1)A necessary and sufficient condition for an ensemble of classiers to be more accurate than any of its individual members is if the classiers are accurate and diverse.

2) An accurate classier:

Necessary and Sufficient Conditions

1)A necessary and sufficient condition for an ensemble of classiers to be more accurate than any of its individual members is if the classiers are accurate and diverse.

2) An accurate classier: is one that has an error rate of better than random guessing on new x values.

Necessary and Sufficient Conditions

1)A necessary and sufficient condition for an ensemble of classiers to be more accurate than any of its individual members is if the classiers are accurate and diverse.

2) An accurate classier: is one that has an error rate of better than random guessing on new x values.

3) Two classifiers are diverse:

Necessary and Sufficient Conditions

1)A necessary and sufficient condition for an ensemble of classiers to be more accurate than any of its individual members is if the classiers are accurate and diverse.

2) An accurate classier: is one that has an error rate of better than random guessing on new x values.

3) Two classifiers are diverse: if they make different errors on new data points

Necessary and Sufficient Conditions

Imagine that we have an ensemble of three classifiers (h_1, h_2, h_3) and consider a new case x.

Necessary and Sufficient Conditions

Imagine that we have an ensemble of three classifiers (h_1, h_2, h_3) and consider a new case x.

If the three classifiers are identical, i.e. not diverse, then when $h_1(x)$ is wrong $h_2(x)$ and $h_3(x)$ will also be wrong.

Necessary and Sufficient Conditions

Imagine that we have an ensemble of three classifiers (h_1, h_2, h_3) and consider a new case x.

If the three classifiers are identical, i.e. not diverse, then when $h_1(x)$ is wrong $h_2(x)$ and $h_3(x)$ will also be wrong.

However, if the errors made by the classifiers are uncorrelated, then when $h_1(x)$ is wrong, $h_2(x)$ and $h_3(x)$ may be correct, so that a majority vote will correctly class.

Intuition for Ensemble Methods from Quantitative Perspective

Intuition for Ensemble Methods from Quantitative Perspective

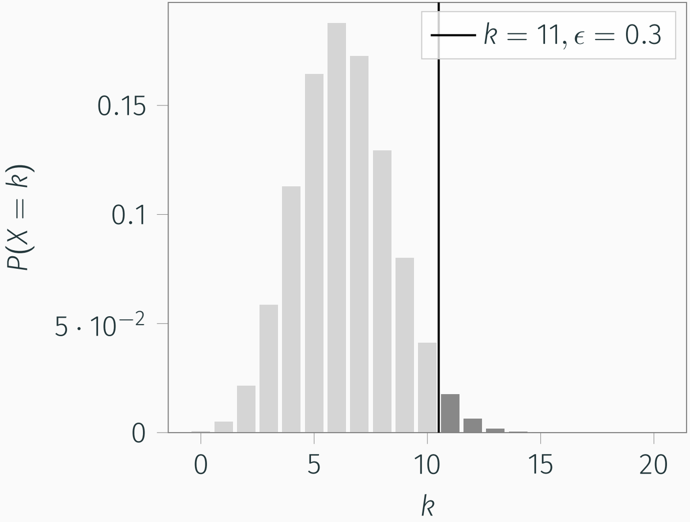
Error Probability of each model = ε = 0.3

$$Pr(\text{ensemble being wrong}) = {}^3C_2(\varepsilon^2)(1-\varepsilon)^{3-2} + {}^3C_3(\varepsilon^3)(1-\varepsilon)^{3-3}$$

$$= 0.19 \leq 0.3$$

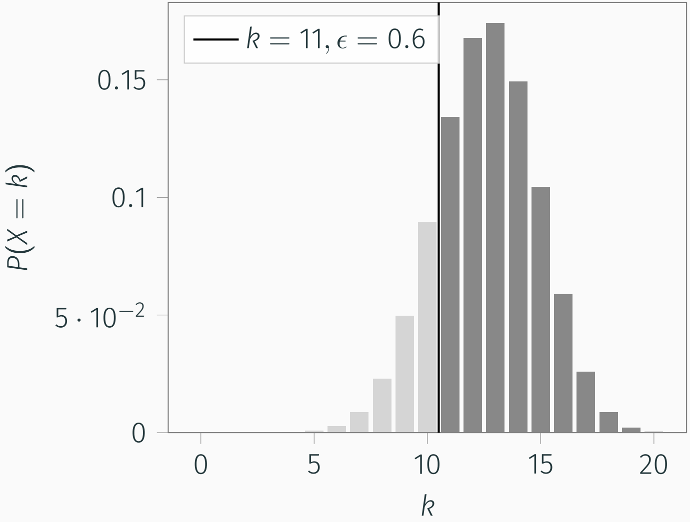
Some calculations

Probability that majority vote (11 out of 21) is wrong = 0.026



Some calculations

Probability that majority vote (11 out of 21) is wrong = 0.826



Where does ensemble learning not work well?

Where does ensemble learning not work well?

- The base model is bad.
- All models give similar prediction or the models are highly correlated.

Bagging

Also known as *Bootstrap Aggregation*.

Bagging

Also known as *Bootstrap Aggregation*.

Key idea : Reduce Variance

Bagging

Also known as *Bootstrap Aggregation*.
Key idea : Reduce Variance

How to learn different classifiers while feeding in the same data?

Bagging

Also known as *Bootstrap Aggregation*.
Key idea : Reduce Variance

How to learn different classifiers while feeding in the same data?

Think about cross-validation!

Bagging

Also known as *Bootstrap Aggregation*.
Key idea : Reduce Variance

How to learn different classifiers while feeding in the same data?

Think about cross-validation!

We will create multiple datasets from our single dataset using "*sampling with replacement*".

Bagging

Consider our dataset has n samples, $D_1, D_2, D_3, \ldots, D_n$.
For each model in the ensemble, we create a new dataset of size n by sampling uniformly with replacement.

Bagging

Consider our dataset has n samples, $D_1, D_2, D_3, \ldots, D_n$.
For each model in the ensemble, we create a new dataset of size n by sampling uniformly with replacement.

$$
\begin{aligned}
&\text{Round 1}: D_1, D_3, D_6, D_1, \ldots, D_n \\
&\text{Round 2}: D_2, D_4, D_1, D_{80}, \ldots, D_3 \\
&\vdots
\end{aligned}
$$

Bagging

Consider our dataset has n samples, $D_1, D_2, D_3, \ldots, D_n$.
For each model in the ensemble, we create a new dataset of size n by sampling uniformly with replacement.

$$\text{Round 1}: D_1, D_3, D_6, D_1, \ldots, D_n$$
$$\text{Round 2}: D_2, D_4, D_1, D_{80}, \ldots, D_3$$
$$\vdots$$

Repetition of samples is possible.

Bagging

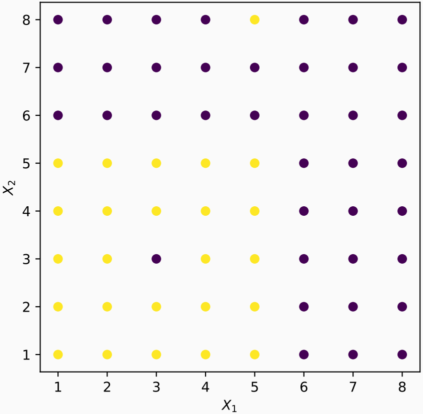
Consider our dataset has n samples, $D_1, D_2, D_3, \ldots, D_n$.
For each model in the ensemble, we create a new dataset of size n by sampling uniformly with replacement.

$$\text{Round 1}: D_1, D_3, D_6, D_1, \ldots, D_n$$
$$\text{Round 2}: D_2, D_4, D_1, D_{80}, \ldots, D_3$$
$$\vdots$$

Repetition of samples is possible.
We can train the same classifier/models on each of these different "Bagging Rounds".

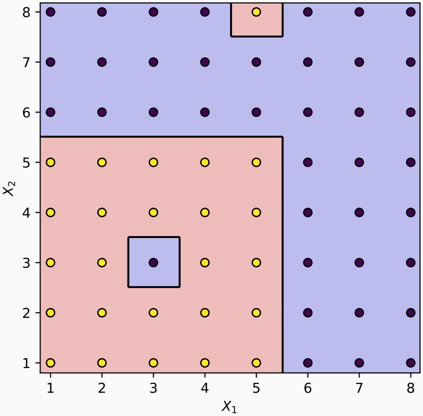
Bagging : Classification Example

Consider the dataset below. Points (3,3) and (5,8) are anomalies.



Bagging : Classification Example

Decision Boundary for decision tree with depth 6.



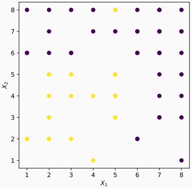
Bagging : Classification Example

Lets use bagging with ensemble of 5 trees.

Bagging : Classification Example

Lets use bagging with ensemble of 5 trees.

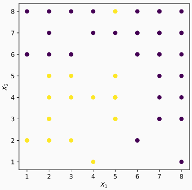
Round - 1



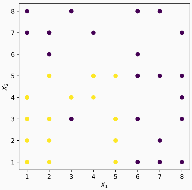
Bagging : Classification Example

Lets use bagging with ensemble of 5 trees.

Round - 1



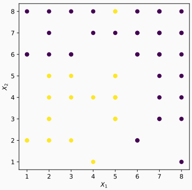
Round - 2



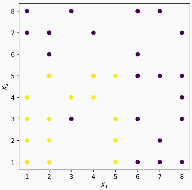
Bagging : Classification Example

Lets use bagging with ensemble of 5 trees.

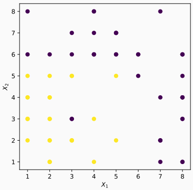
Round - 1



Round - 2



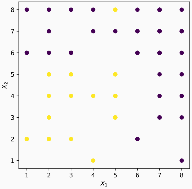
Round - 3



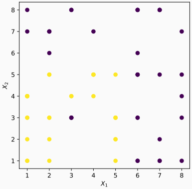
Bagging : Classification Example

Lets use bagging with ensemble of 5 trees.

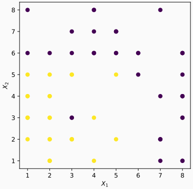
Round - 1



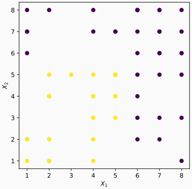
Round - 2



Round - 3



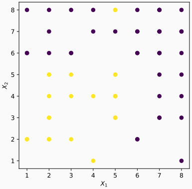
Round - 4



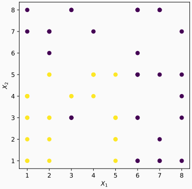
Bagging : Classification Example

Lets use bagging with ensemble of 5 trees.

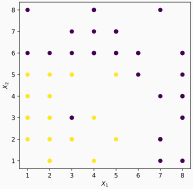
Round - 1



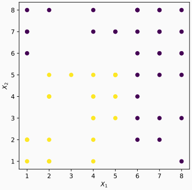
Round - 2



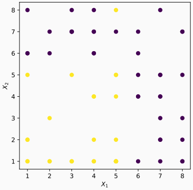
Round - 3



Round - 4



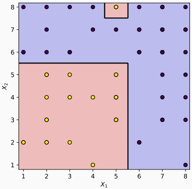
Round - 5



Bagging : Classification Example

Bagging : Classification Example

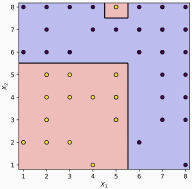
Round - 1



Tree

Depth = 4

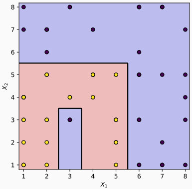
Round - 1



Tree

Depth = 4

Round - 2

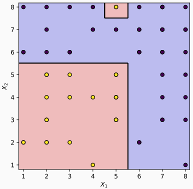


Tree

Depth = 5

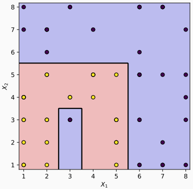
Bagging : Classification Example

Round - 1



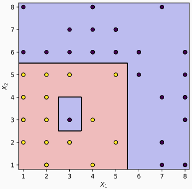
Tree Depth = 4

Round - 2



Tree Depth = 5

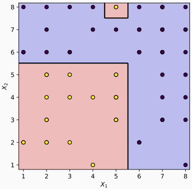
Round - 3



Tree Depth = 5

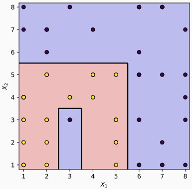
Bagging : Classification Example

Round - 1



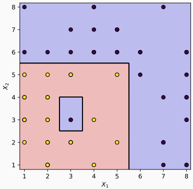
Tree Depth = 4

Round - 2



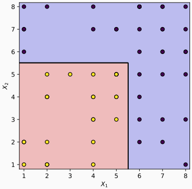
Tree Depth = 5

Round - 3



Tree Depth = 5

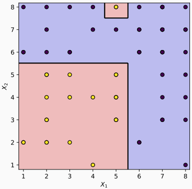
Round - 4



Tree Depth = 2

Bagging : Classification Example

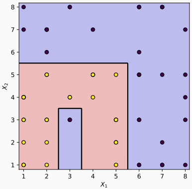
Round - 1



Tree

Depth = 4

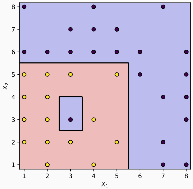
Round - 2



Tree

Depth = 5

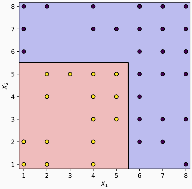
Round - 3



Tree

Depth = 5

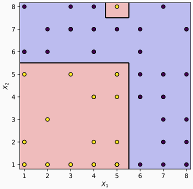
Round - 4



Tree

Depth = 2

Round - 5

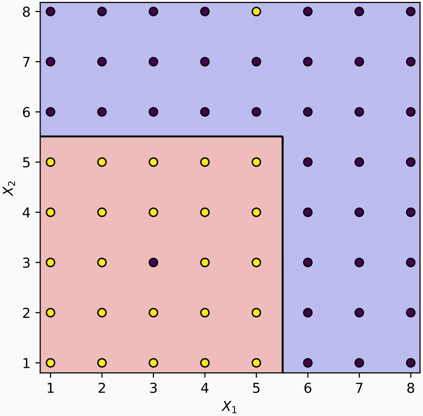


Tree

Depth = 4

Bagging : Classification Example

Using majority voting to combine all predictions, we get the decision boundary below.



Bagging

Summary

- We take “strong” learners and combine them to reduce variance.
- All learners are independent of each other.

Boosting

- We take "weak" learners and combine them to reduce bias.

Boosting

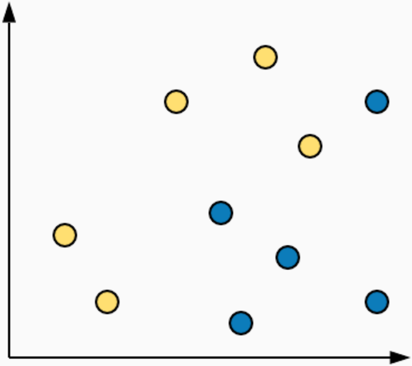
- We take “weak” learners and combine them to reduce bias.
- All learners are incrementally built.

Boosting

- We take “weak” learners and combine them to reduce bias.
- All learners are incrementally built.
- Incremental building: Incrementally try to classify “harder” samples correctly.

Consider we have a dataset of N samples.

Sample i has weight w_i. There are M classifers in ensemble.

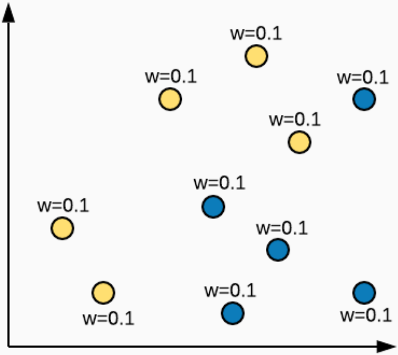


Boosting : AdaBoost

Consider we have a dataset of N samples.

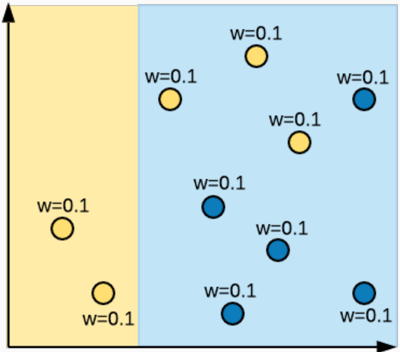
Sample i has weight w_i. There are M classifers in ensemble.

1. Initialize weights of data samples, $w_i = \dfrac{1}{N}$



Consider we have a dataset of N samples.
Sample i has weight w_i. There are M classifers in ensemble.

1. Initialize weights of data samples, $w_i = \dfrac{1}{N}$
2. For $m = 1 \ldots M$
   2.1 Learn classifier using current weights $w_i's$

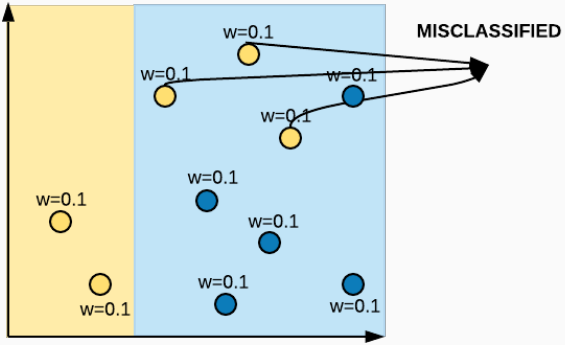


Boosting : AdaBoost

Consider we have a dataset of N samples.

Sample i has weight w_i. There are M classifers in ensemble.

1. Initialize weights of data samples, $w_i = \dfrac{1}{N}$
2. For $m = 1 \dots M$
    2.1 Learn classifier using current weights $w_i' s$

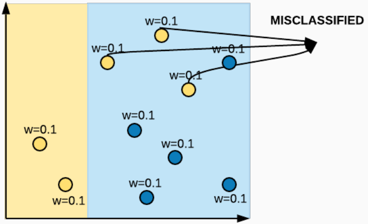


Boosting : AdaBoost

Consider we have a dataset of N samples.

Sample i has weight w_i. There are M classifers in ensemble.

1. Initialize weights of data samples, $w_i = \dfrac{1}{N}$
2. For $m = 1 \ldots M$
   2.1 Learn classifier using current weights $w_i's$
   2.2 Compute the weighted error, $err_m = \dfrac{\sum_i w_i(incorrect)}{\sum_i w_i}$



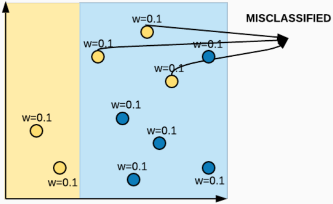
$$err_1 = \frac{0.3}{1}$$

Boosting : AdaBoost

Consider we have a dataset of N samples.

Sample i has weight w_i. There are M classifers in ensemble.

1. Initialize weights of data samples, $w_i = \dfrac{1}{N}$
2. For $m = 1 \ldots M$
   2.1 Learn classifier using current weights $w_i's$
   2.2 Compute the weighted error, $err_m = \dfrac{\sum_i w_i(incorrect)}{\sum_i w_i}$
   2.3 Compute $\alpha_m = \dfrac{1}{2} log_e \left(\dfrac{1 - err_m}{err_m} \right)$



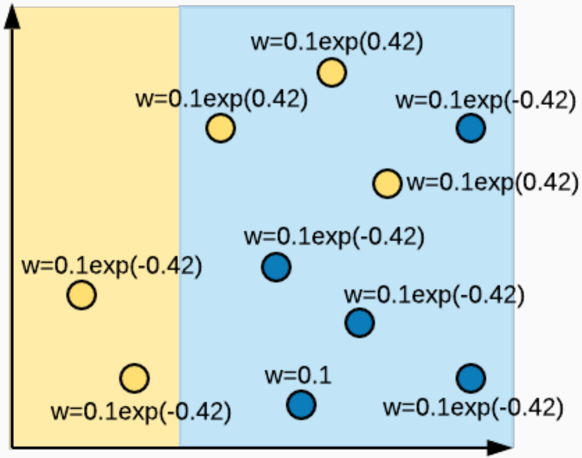
$$err_1 = \frac{0.3}{1}$$

$$\alpha_1 = \frac{1}{2} log \left(\frac{1 - 0.3}{0.3} \right) = 0.42$$

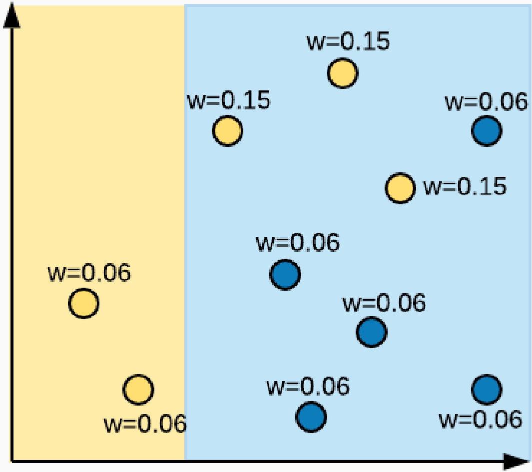
Boosting : AdaBoost

Consider we have a dataset of N samples.
Sample i has weight w_i. There are M classifers in ensemble.

1. Initialize weights of data samples, $w_i = \dfrac{1}{N}$
2. For $m = 1 \dots M$
   2.1 Learn classifier using current weights $w_i's$
   2.2 Compute the weighted error, $err_m = \dfrac{\sum_i w_i(incorrect)}{\sum_i w_i}$
   2.3 Compute $\alpha_m = \dfrac{1}{2} log_e \left(\dfrac{1 - err_m}{err_m} \right)$
   2.4 For samples which were predicted correctly, $w_i = w_i e^{-\alpha_m}$
   2.5 For samples which were predicted incorrectly, $w_i = w_i e^{\alpha_m}$


w=0.1exp(0.42)
w=0.1exp(0.42)
w=0.1exp(-0.42)
w=0.1exp(0.42)
w=0.1exp(-0.42)
w=0.1exp(-0.42)
w=0.1exp(-0.42)
w=0.1
w=0.1exp(-0.42)
w=0.1exp(-0.42)

Boosting : AdaBoost

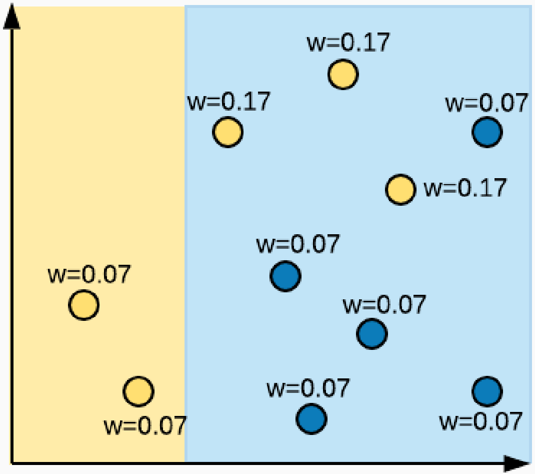


Boosting : AdaBoost

Consider we have a dataset of N samples.
Sample i has weight w_i. There are M classifers in ensemble.

1. Initialize weights of data samples, $w_i = \dfrac{1}{N}$
2. For $m = 1 \dots M$
    2.1 Learn classifier using current weights $w_i's$
    2.2 Compute the weighted error, $err_m = \dfrac{\sum_i w_i(incorrect)}{\sum_i w_i}$
    2.3 Compute $\alpha_m = \frac{1}{2} log_e \left(\dfrac{1 - err_m}{err_m} \right)$
    2.4 For samples which were predicted correctly, $w_i = w_i e^{-\alpha_m}$
    2.5 For samples which were predicted incorrectly, $w_i = w_i e^{\alpha_m}$
    2.6 Normalize $w_i's$ to sum up to 1.

Boosting : AdaBoost

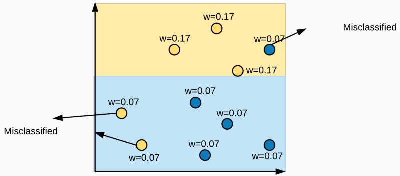


Boosting : AdaBoost

Consider we have a dataset of N samples.

Sample i has weight w_i. There are M classifers in ensemble.

1. Initialize weights of data samples, $w_i = \dfrac{1}{N}$
2. For $m = 1 \dots M$
    2.1 Learn classifier using current weights $w_i's$
    2.2 Compute the weighted error, $err_m = \dfrac{\sum_i w_i(incorrect)}{\sum_i w_i}$
    2.3 Compute $\alpha_m = \dfrac{1}{2} log_e \left(\dfrac{1 - err_m}{err_m} \right)$



$$err_2 = \frac{0.21}{1}$$

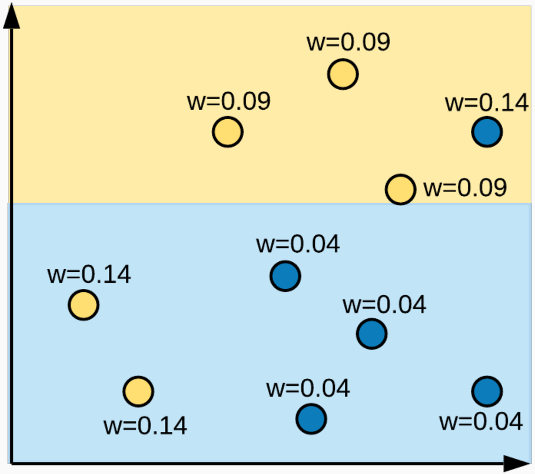
$$\alpha_2 = \frac{1}{2} log \left(\frac{1 - 0.21}{0.21} \right) = 0.66$$

Boosting : AdaBoost

Consider we have a dataset of N samples.
Sample i has weight w_i. There are M classifers in ensemble.

1. Initialize weights of data samples, $w_i = \dfrac{1}{N}$
2. For $m = 1 \dots M$
    2.1 Learn classifier using current weights $w_i's$
    2.2 Compute the weighted error, $err_m = \dfrac{\sum_i w_i(incorrect)}{\sum_i w_i}$
    2.3 Compute $\alpha_m = \frac{1}{2} log_e \left(\dfrac{1 - err_m}{err_m} \right)$
    2.4 For samples which were predicted correctly, $w_i = w_i e^{-\alpha_m}$
    2.5 For samples which were predicted incorrectly, $w_i = w_i e^{\alpha_m}$

Boosting : AdaBoost

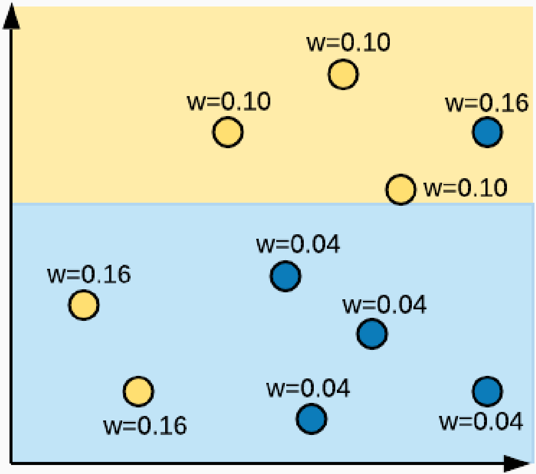


Boosting : AdaBoost

Consider we have a dataset of N samples.
Sample i has weight w_i. There are M classifers in ensemble.

1. Initialize weights of data samples, $w_i = \dfrac{1}{N}$
2. For $m = 1 \ldots M$
   2.1 Learn classifier using current weights $w_i's$
   2.2 Compute the weighted error, $err_m = \dfrac{\sum_i w_i(incorrect)}{\sum_i w_i}$
   2.3 Compute $\alpha_m = \dfrac{1}{2} log_e \left(\dfrac{1 - err_m}{err_m} \right)$
   2.4 For samples which were predicted correctly, $w_i = w_i e^{-\alpha_m}$
   2.5 For samples which were predicted incorrectly, $w_i = w_i e^{\alpha_m}$
   2.6 Normalize $w_i's$ to sum up to 1.

Boosting : AdaBoost

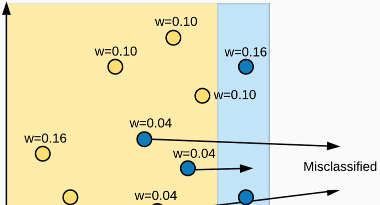


Boosting : AdaBoost

Consider we have a dataset of N samples.

Sample i has weight w_i. There are M classifers in ensemble.

1. Initialize weights of data samples, $w_i = \dfrac{1}{N}$
2. For $m = 1 \ldots M$
   - 2.1 Learn classifier using current weights $w_i's$
   - 2.2 Compute the weighted error, $err_m = \dfrac{\sum_i w_i(incorrect)}{\sum_i w_i}$
   - 2.3 Compute $\alpha_m = \dfrac{1}{2} log_e \left(\dfrac{1 - err_m}{err_m} \right)$



$$err_3 = \frac{0.12}{1}$$

$$\alpha_3 = \frac{1}{2} log \left(\frac{1 - 0.12}{0.12} \right) = 0.99$$

Intuitively, after each iteration, importance of wrongly classified samples is increased by increasing their weights and importance of correctly classified samples is decreased by decreasing their weights.

Testing

Testing

Final Prediction = SIGN(α_1(Pred. of Clf. 1) + α_2(Pred. Clf. 2) + ...
+ α_M(Pred. Clf M))

Testing

Final Prediction = SIGN(α_1(Pred. of Clf. 1) + α_2(Pred. Clf. 2) + . . . + α_M(Pred. Clf M))

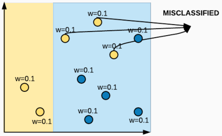


Figure 1: $\alpha_1 = 0.42$

Testing

Final Prediction = SIGN(α_1(Pred. of Clf. 1) + α_2(Pred. Clf. 2) + ...
+ α_M(Pred. Clf M))

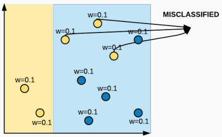


Figure 1: $\alpha_1 = 0.42$

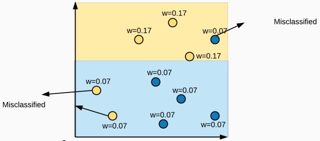


Figure 2: $\alpha_2 = 0.66$

Testing

Final Prediction = SIGN(α_1(Pred. of Clf. 1) + α_2(Pred. Clf. 2) + . . .
+ α_M(Pred. Clf M))

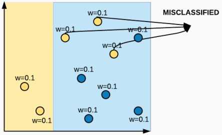


Figure 1: $\alpha_1 = 0.42$

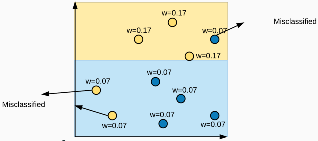


Figure 2: $\alpha_2 = 0.66$

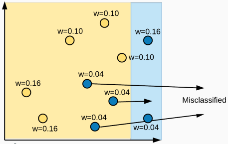


Figure 3: $\alpha_3 = 0.99$

Testing

Final Prediction = SIGN(α_1(Pred. of Clf. 1) + α_2(Pred. Clf. 2) + ... + α_M(Pred. Clf M))

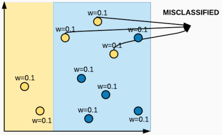


Figure 1: $\alpha_1 = 0.42$

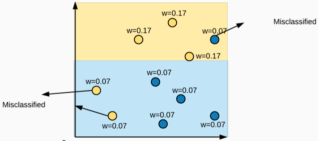


Figure 2: $\alpha_2 = 0.66$

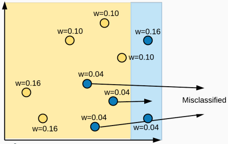
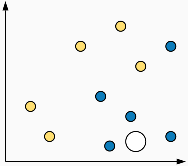


Figure 3: $\alpha_3 = 0.99$



Boosting: Adaboost

Testing

Final Prediction = SIGN(α_1(Pred. of Clf. 1) + α_2(Pred. Clf. 2) + . . .
+ α_M(Pred. Clf M))

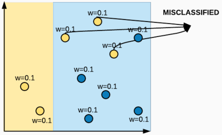


Figure 1: $\alpha_1 = 0.42$

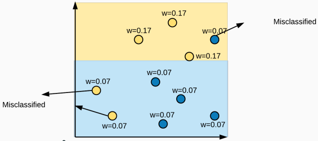


Figure 2: $\alpha_2 = 0.66$

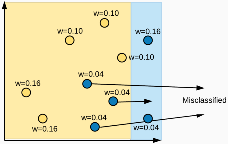
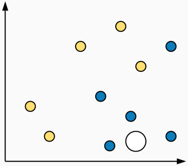


Figure 3: $\alpha_3 = 0.99$



Let us say, yellow class is +1
and blue class is -1

Testing

Final Prediction = SIGN(α_1(Pred. of Clf. 1) + α_2(Pred. Clf. 2) + ... + α_M(Pred. Clf M))

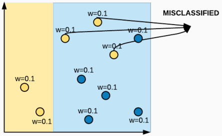


Figure 1: $\alpha_1 = 0.42$

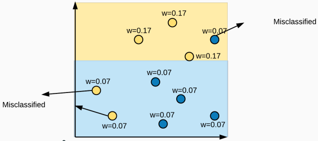


Figure 2: $\alpha_2 = 0.66$

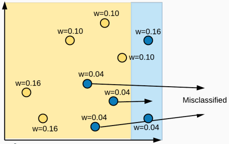
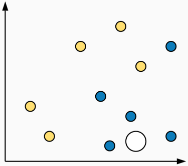


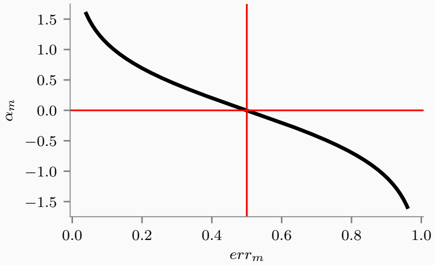
Figure 3: $\alpha_3 = 0.99$



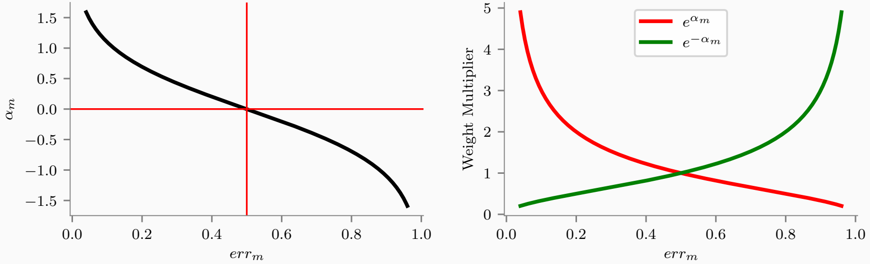
Let us say, yellow class is +1 and blue class is -1
Prediction = SIGN(0.42*-1 + 0.66*-1 + 0.99*+1) = Negative = blue

Intuition behind weight update formula

Intuition behind weight update formula



Intuition behind weight update formula



Random Forest

It is an ensemble of decision trees, where each tree is trained on randomly-selected features.

As features are randomly selected, we learn decorrelated trees and helps in reducing variance.

Random Forest

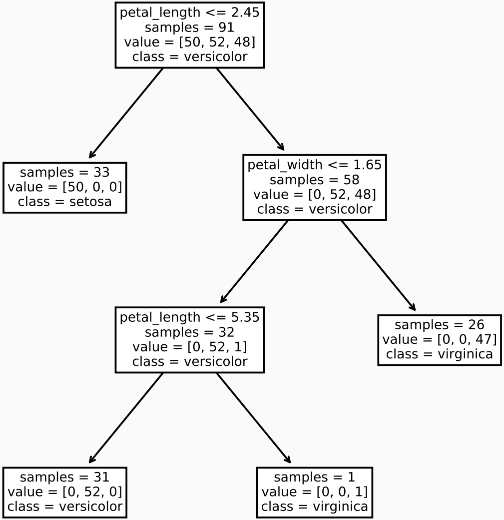
- for *tree* in $[1, \ldots, \mathit{number\ of\ trees}]$
    - For each split, select "m" features from total available M features and train a decision tree on selected features

Dataset

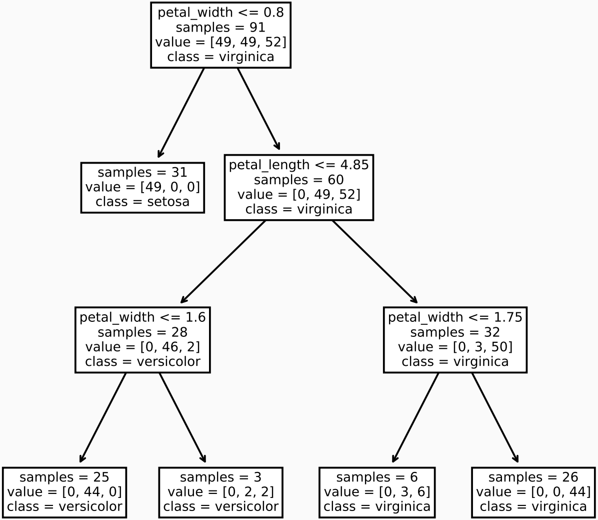
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

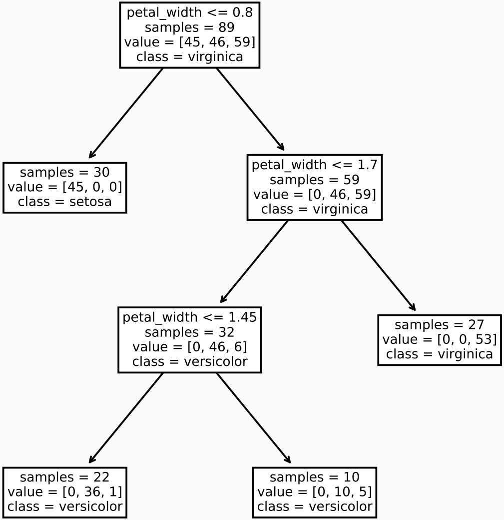
Decision Tree # 0



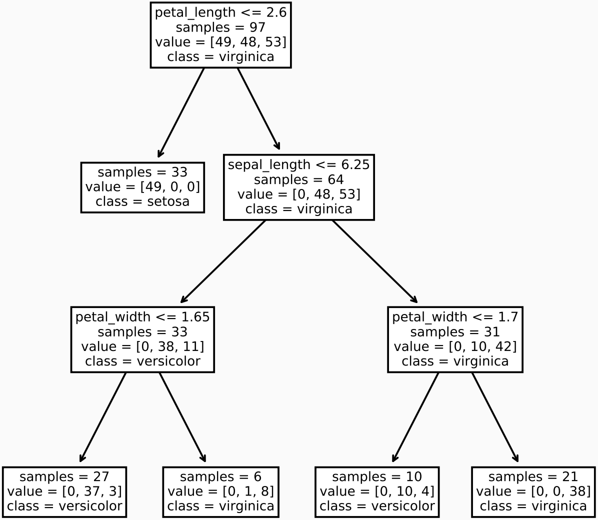
Decision Tree # 1



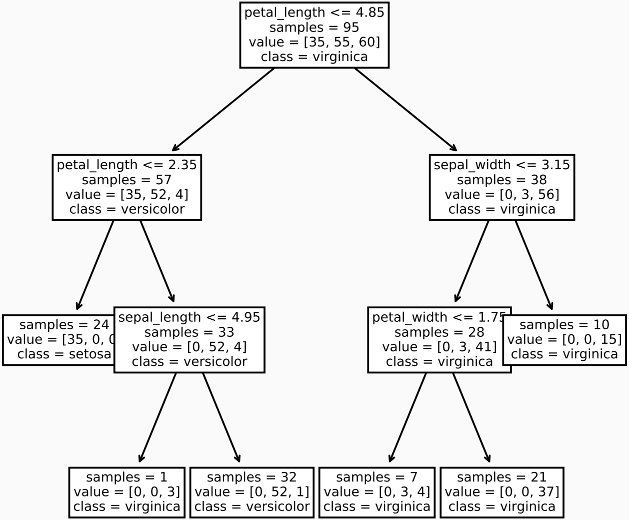
Decision Tree # 2



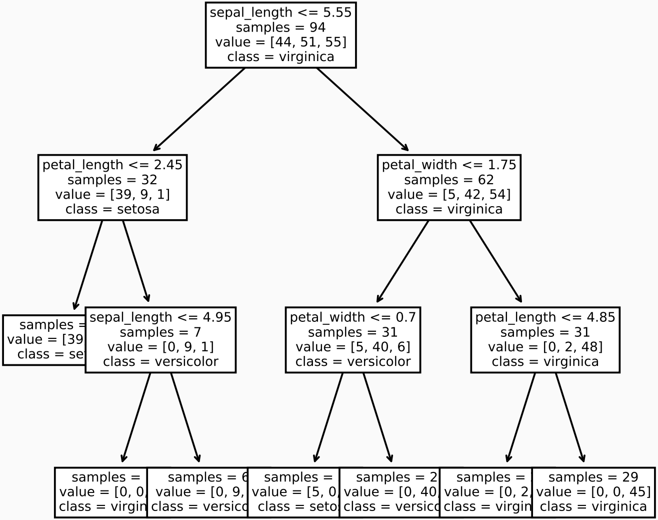
Decision Tree # 3



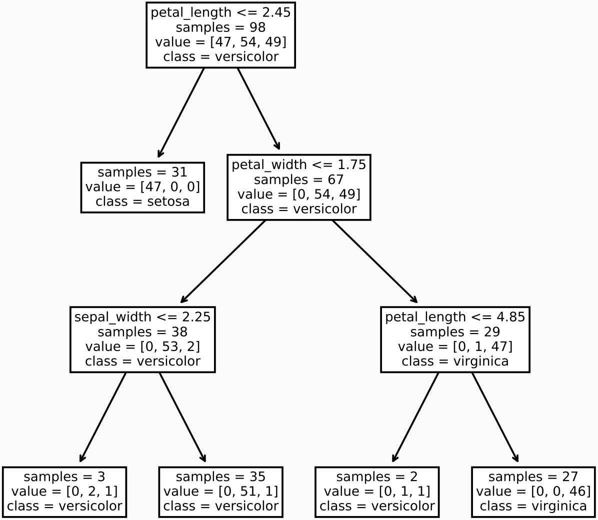
Decision Tree # 4



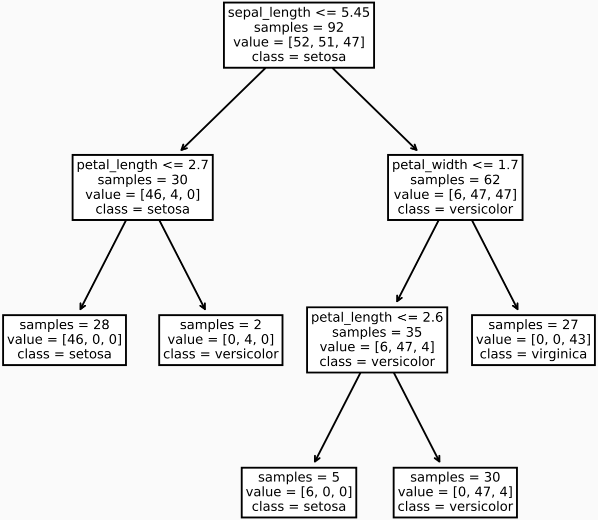
Decision Tree # 5



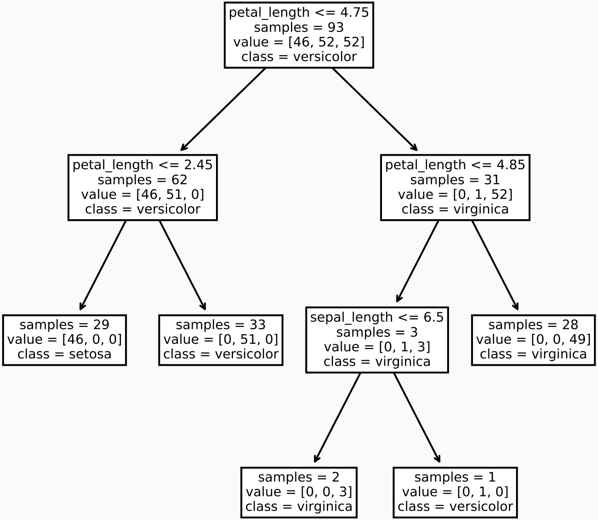
Decision Tree # 6



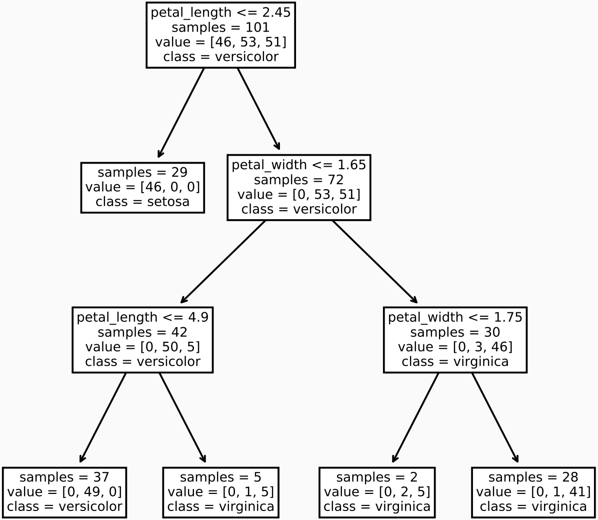
Decision Tree # 7



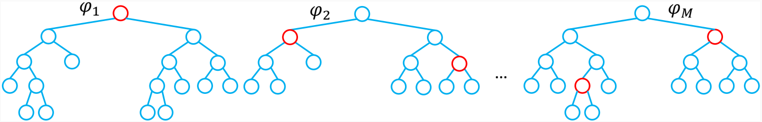
Decision Tree # 8



Decision Tree # 9



Feature Importance[1]



Importance of variable X_j for an ensemble of M trees φ_m is:

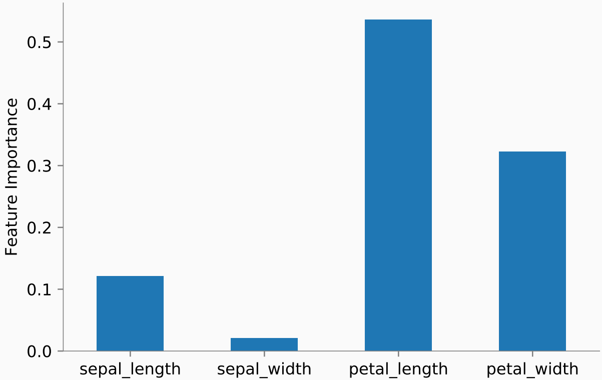
$$\text{Imp}(X_j) = \frac{1}{M}\sum_{m=1}^{M}\sum_{t\in\varphi_m} 1(j_t = j)\Big[p(t)\Delta i(t)\Big],$$

where j_t denotes the variable used at node t, $p(t) = N_t/N$ and $\Delta i(t)$ is the impurity reduction at node t:

$$\Delta i(t) = i(t) - \frac{N_{t_L}}{N_t} i(t_L) - \frac{N_{t_r}}{N_t} i(t_R)$$

[1]Slide Courtesy Gilles Louppe

Computed Feature Importance



Code for Examples

Google Colab