RECOD
reasoning for complex data

Machine Learning and Pattern Recognition

A High Level Overview

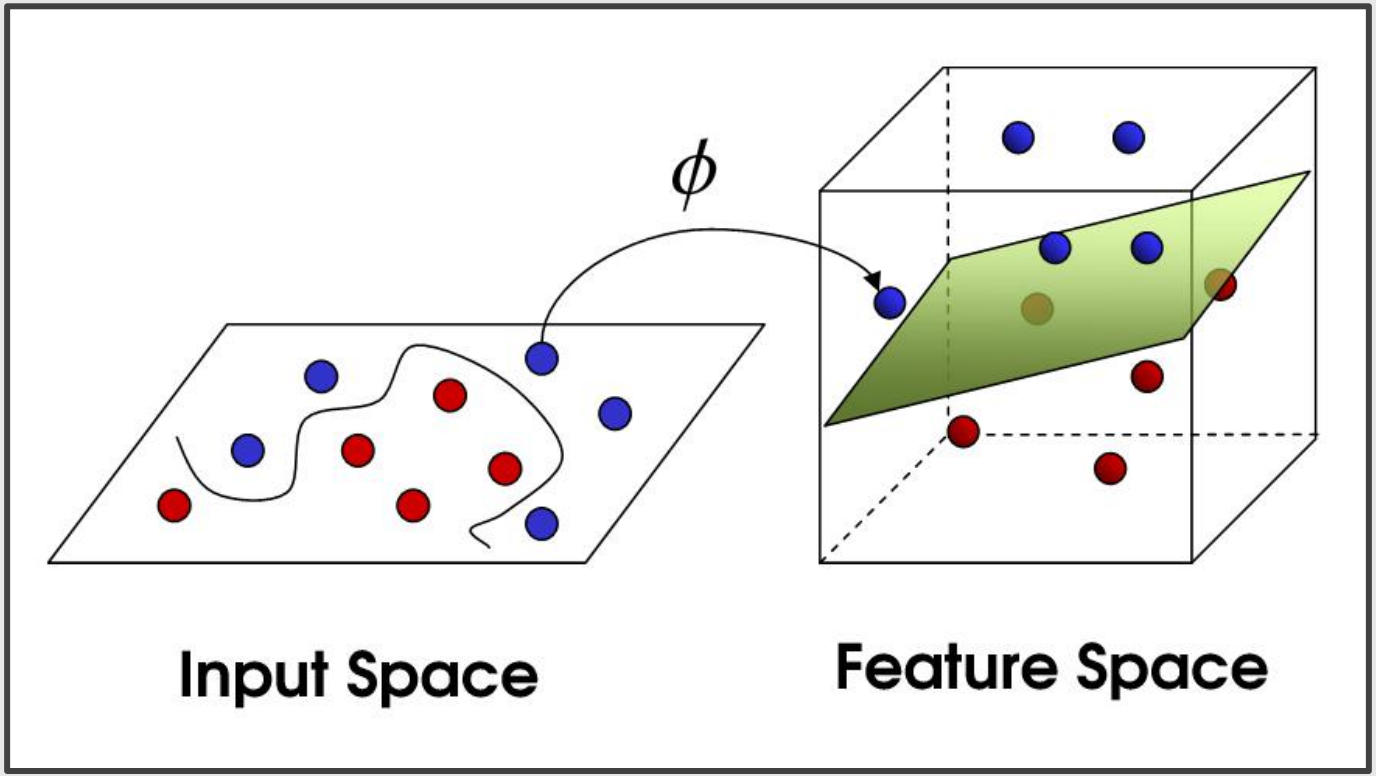
Prof. Anderson Rocha
(Main bulk of slides kindly provided by **Prof. Sandra Avila**)
Institute of Computing (IC/Unicamp)

MC886/MO444

Non-linear SVMs: Feature spaces

General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable.

$$\Phi : x \rightarrow \varphi(x)$$



Kernel Trick

- The linear classifier relies on inner product between vector
  $K(x_i, x_j) = x_i^T x_j$
- If every datapoint is mapped into high-dimensional space via some transformation $\Phi: x \rightarrow \varphi(x)$, the inner product becomes:

$$\boldsymbol{K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)}$$

- A kernel function is a function that is equivalent to an inner product in some feature space.

Kernelized SVM

Suppose you want to apply a 2^{nd} degree polynomial transformation to a 2-dimensional training set, then train a linear SVM classifier on the transformed training set.

$$\varphi(x) = \varphi((x_1\ x_2)) = (x_1^2\ \ \sqrt{2}x_1x_2\ \ x_2^2)$$

The transformed vector is 3-dimensional instead of 2-dimensional.

Kernelized SVM

Let's look at what happens to a couple 2-dimensional vectors **a** and **b** if we apply this 2nd degree polynomial mapping then compute the dot product of the transformed vectors.

$$\varphi(\mathbf{a})^T\varphi(\mathbf{b}) =$$

Kernelized SVM

Let's look at what happens to a couple 2-dimensional vectors **a** and **b** if we apply this 2nd degree polynomial mapping then compute the dot product of the transformed vectors.

$$\varphi(\mathbf{a})^T\varphi(\mathbf{b}) = (a_1^2 \;\; \sqrt{2}a_1a_2 \;\; a_2^2)^T(b_1^2 \;\; \sqrt{2}b_1b_2 \;\; b_2^2) =$$

Kernelized SVM

Let's look at what happens to a couple 2-dimensional vectors **a** and **b** if we apply this 2nd degree polynomial mapping then compute the dot product of the transformed vectors.

$$\varphi(\mathbf{a})^T\varphi(\mathbf{b}) = (a_1^2 \;\; \sqrt{2}a_1a_2 \;\; a_2^2)^T(b_1^2 \;\; \sqrt{2}b_1b_2 \;\; b_2^2) =$$
$$= a_1^2b_1^2 + 2a_1a_2b_1b_2 + a_2^2b_2^2 =$$
$$= (a_1b_1 + a_2b_2)^2 =$$
$$= ((a_1 \; a_2)^T(b_1 \; b_2))^2 =$$
$$= (\mathbf{a}^T \cdot \mathbf{b})^2$$

Kernelized SVM

The dot product of the transformed vectors is equal to the square of the dot product of the original vectors: $\varphi(\mathbf{a})^T\varphi(\mathbf{b}) = (\mathbf{a}^T \cdot \mathbf{b})^2$

Kernelized SVM

The dot product of the transformed vectors is equal to the square of the dot product of the original vectors: $\varphi(\mathbf{a})^T \varphi(\mathbf{b}) = (\mathbf{a}^T \cdot \mathbf{b})^2$

So **you don't actually need to transform the training instances** at all: just replace the dot product by its square.

This is the essence of the kernel trick.

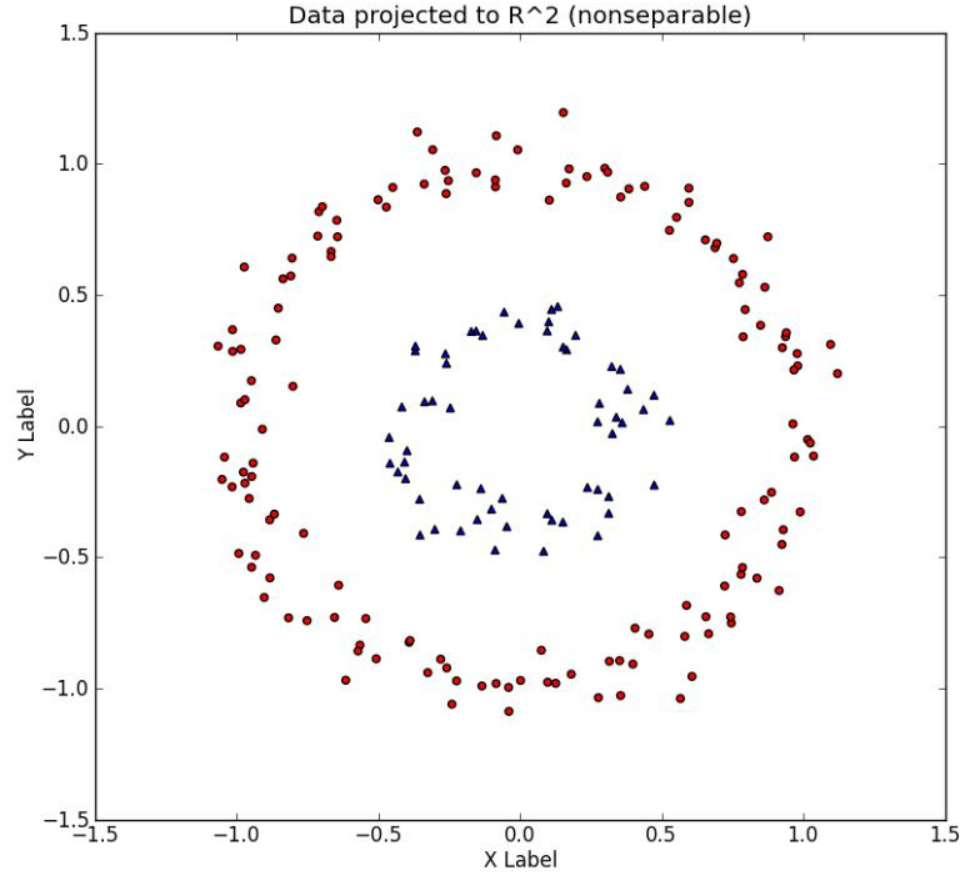
Kernelized SVM

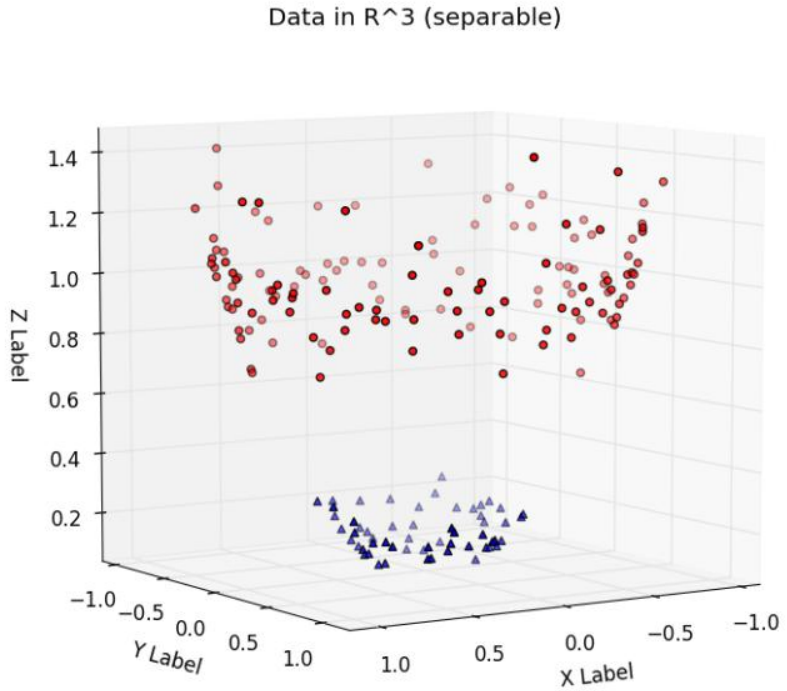
In Machine Learning, a ***kernel*** is a function capable of computing the dot product $\varphi(\mathbf{a})^T\varphi(\mathbf{b})$ based only on the original vectors **a** and **b**, without having to compute (or even to know about) the transformation φ.

Kernel Functions

- Linear SVM = Kernel SVM: $K(x_i, x_j) = x_i^{\mathrm{T}} x_j$
- Nonlinear SVM: $K(x_i, x_j) = \varphi(x_i) \cdot \varphi(x_j)$

- Gaussian kernel: $K(x_i, x_j) = \exp(-\|x_i - x_j\|^2/(2\sigma^2))$
- Polynomial kernel: $K(x_i, x_j) = (x_i \cdot x_j + 1)^d$, d degree
- Chi-square kernel, histogram intersection kernel, string kernel, ...

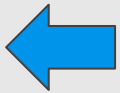

Data projected to R^2 (nonseparable)
Y Label
X Label
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


Data in R^3 (separable)
Z Label
Y Label
X Label
1.4
1.2
1.0
0.8
0.6
0.4
0.2
−1.0
−0.5
0.0
0.5
1.0
1.0
0.5
0.0
−0.5
−1.0

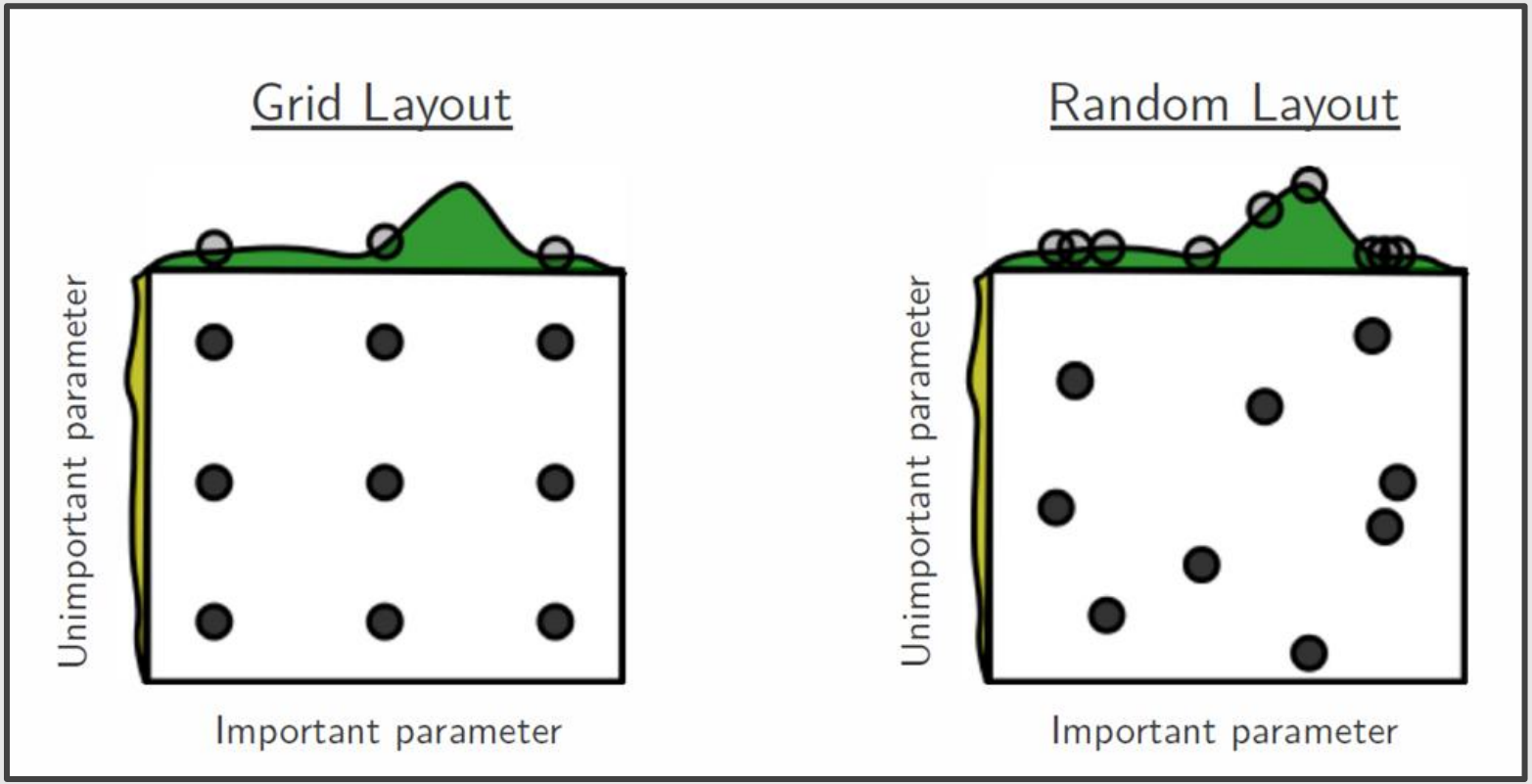
Important Parameters

Important parameters having higher impact on model performance, “kernel”, “gamma” and “C”.

The parameters can be tuned using grid-search.



Grid Search


Grid Layout
Unimportant parameter
Important parameter
Random Layout
Unimportant parameter
Important parameter

References

Machine Learning Books

- Hands-On Machine Learning with Scikit-Learn and TensorFlow, Chap. 5
- Pattern Recognition and Machine Learning, Chap. 6 & 7

Machine Learning Courses

- https://www.coursera.org/learn/machine-learning, Week 7
- http://cs229.stanford.edu/syllabus.html, http://cs229.stanford.edu/notes/cs229-notes3.pdf

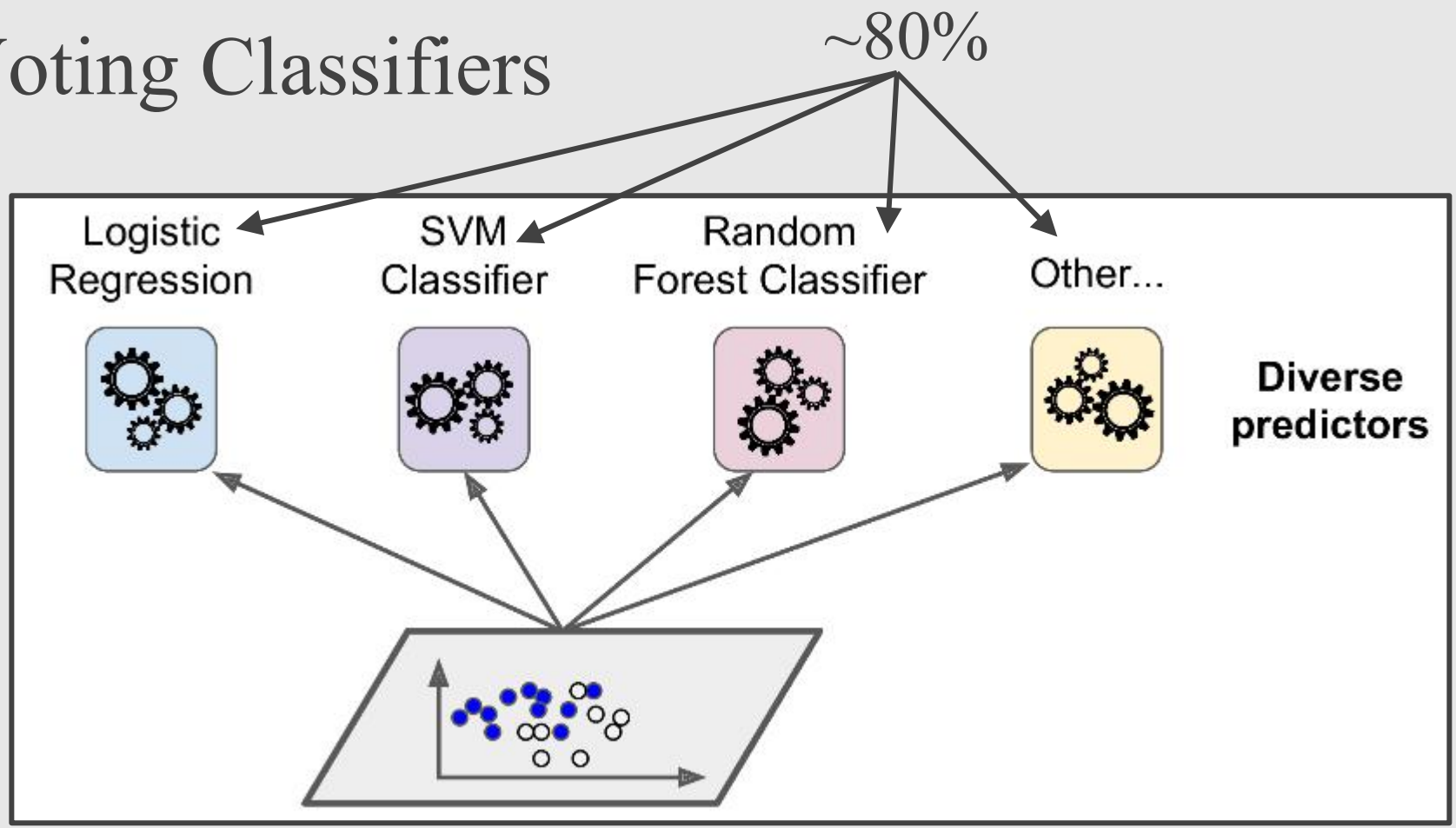
Ensemble Learning and Random Forests

Machine Learning and Pattern Recognition

Prof. Sandra Avila
Institute of Computing (IC/Unicamp)

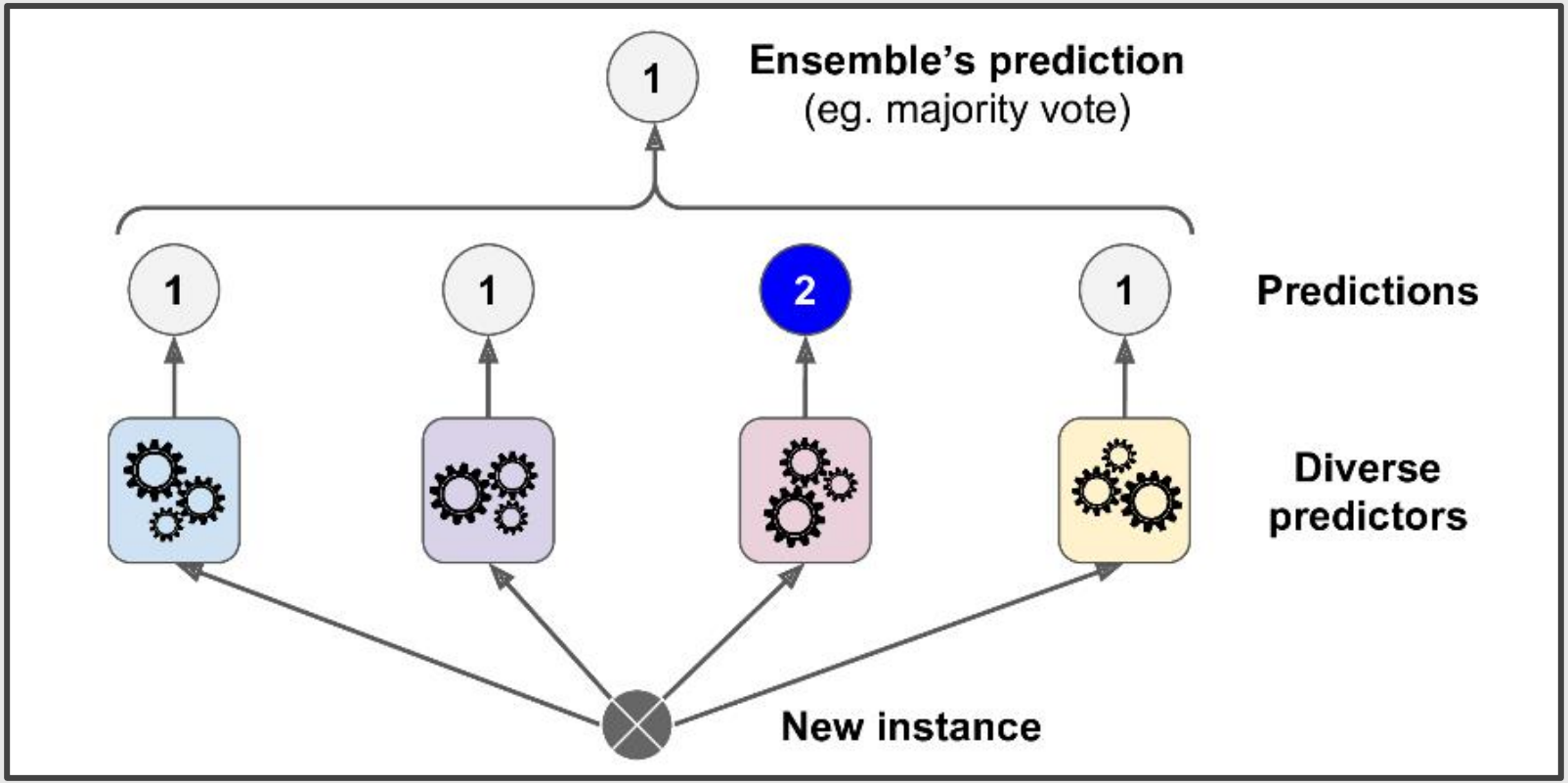
MC886/MO444, November 21, 2017

Voting Classifiers



Voting Classifiers

Hard voting classifier



Voting Classifiers

- Voting classifier often achieves a higher accuracy than the best classifier in the ensemble.

- Even if each classifier is a weak learner (meaning it does only slightly better than random guessing), the ensemble can still be a strong learner (achieving high accuracy).

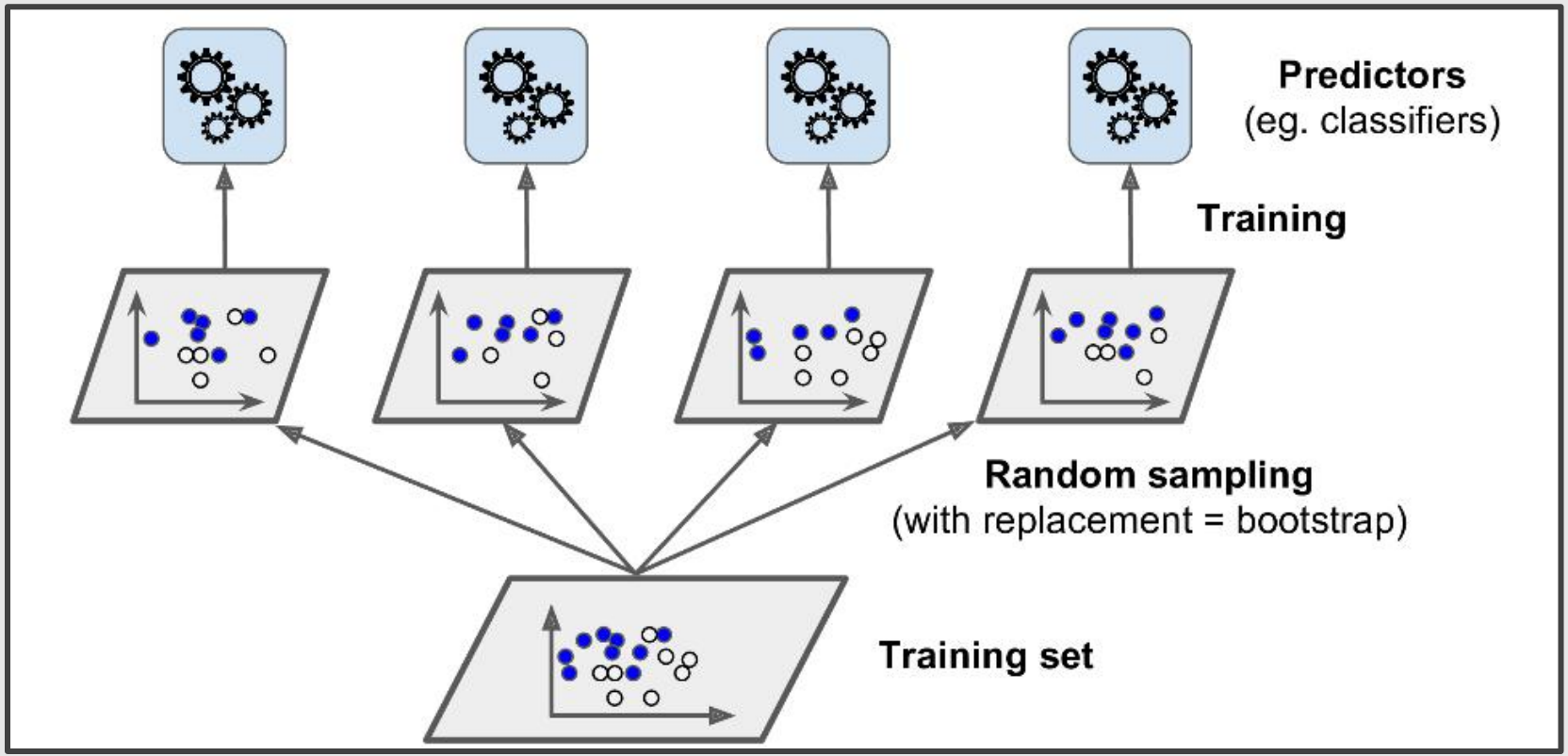
Bagging and Pasting

- Use **the same training algorithm** for every predictor, but to train them on different random subsets of the training set.

Bagging and Pasting

- Use **the same training algorithm** for every predictor, but to train them on different random subsets of the training set.
- Bagging (short for Bootstrap Aggregating): sampling is performed **with** replacement.
- Pasting: sampling is performed **without** replacement

Bagging and Pasting



Bagging and Pasting

- Once all predictors are trained, the ensemble can make a prediction for a new instance by simply aggregating the predictions of all predictors.

- Bagging and Pasting scale very well.

Bagging and Pasting

- Once all predictors are trained, the ensemble can make a prediction for a new instance by simply aggregating the predictions of all predictors.
- Bagging and Pasting scale very well.
- Out-of-bag evaluation: m% for training and (1-m)% for test

Decision Tree & Random Forest

Decision Tree & Random Forest

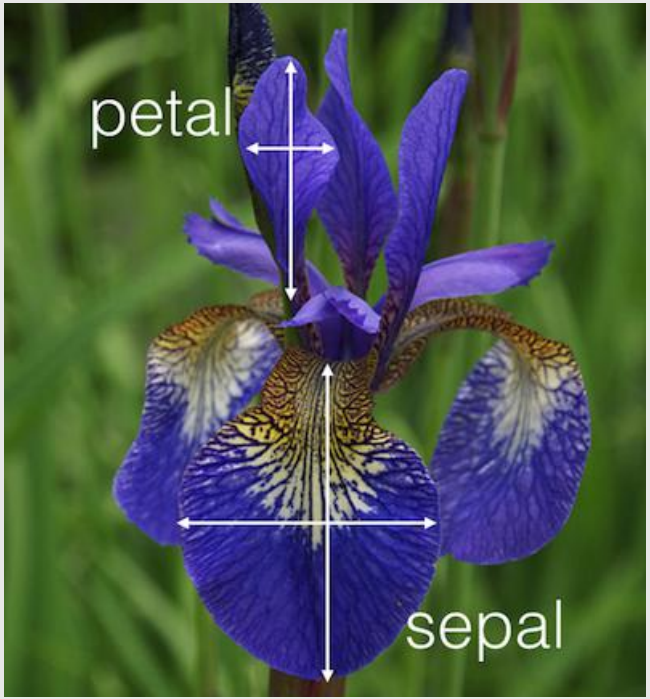
- Decision Trees are versatile Machine Learning algorithms that can perform both classification and regression tasks, and even multi-output tasks.

Decision Tree & Random Forest

- Decision Trees are versatile Machine Learning algorithms that can perform both classification and regression tasks, and even multi-output tasks.

- Random Forest is an ensemble of Decision Trees, generally trained using the Bagging method (or sometimes Pasting).

Decision Tree: Iris Dataset



http://sebastianraschka.com/Articles/2014_python_lda.html

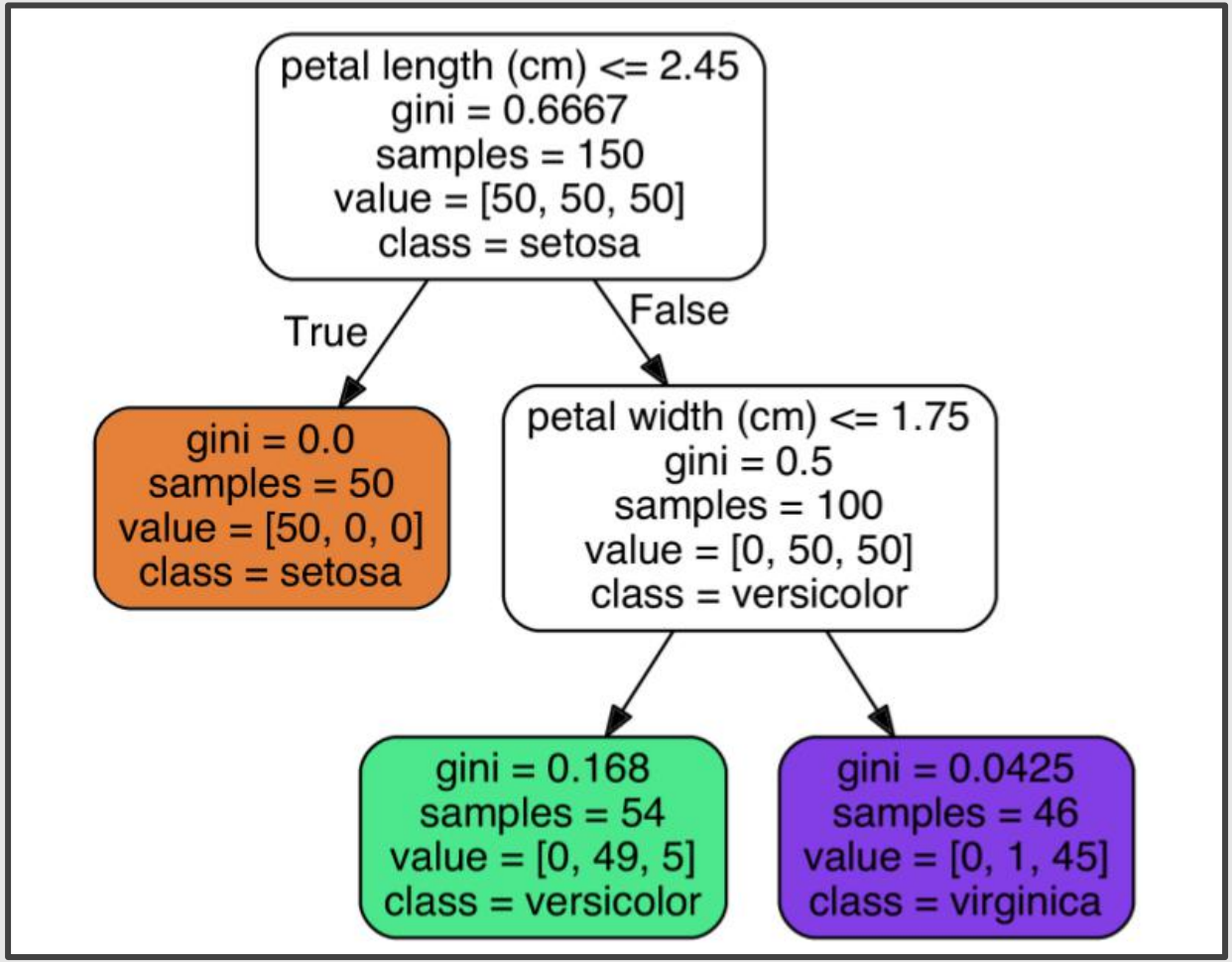
150 iris flowers from three different species.

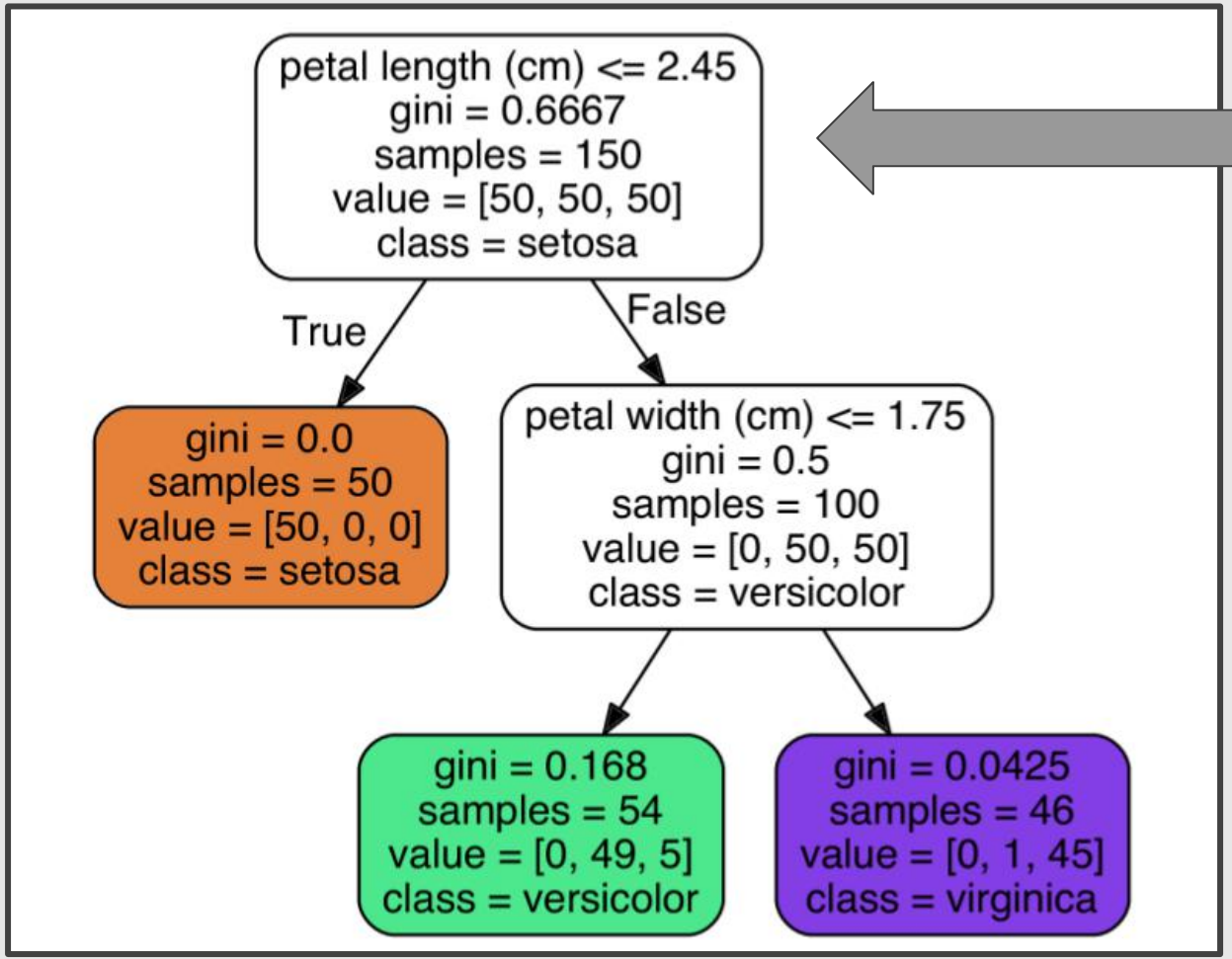
The three classes in the Iris dataset:

1. Iris-setosa (*n*=50)
2. Iris-versicolor (*n*=50)
3. Iris-virginica (*n*=50)

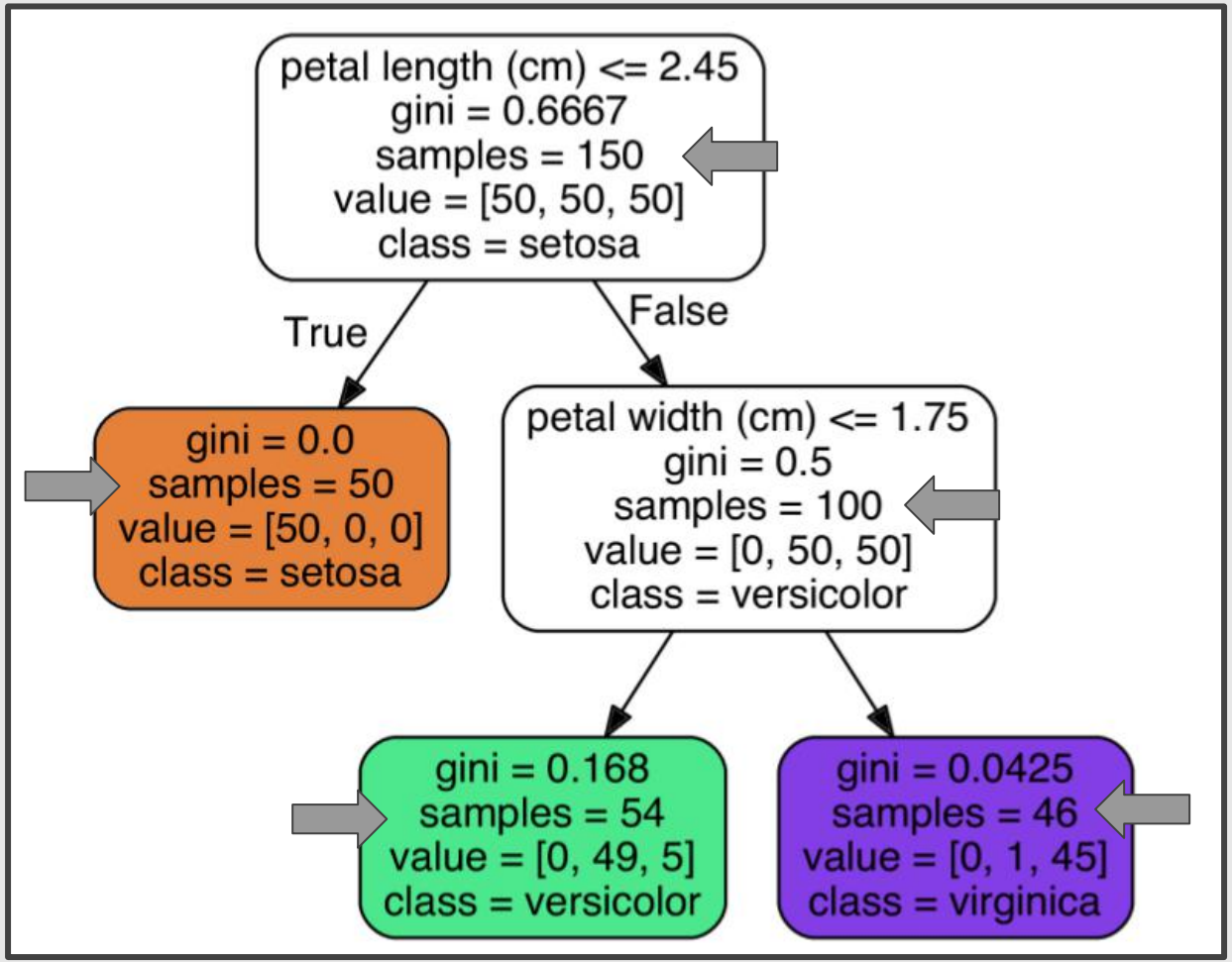
The four features of the Iris dataset:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm

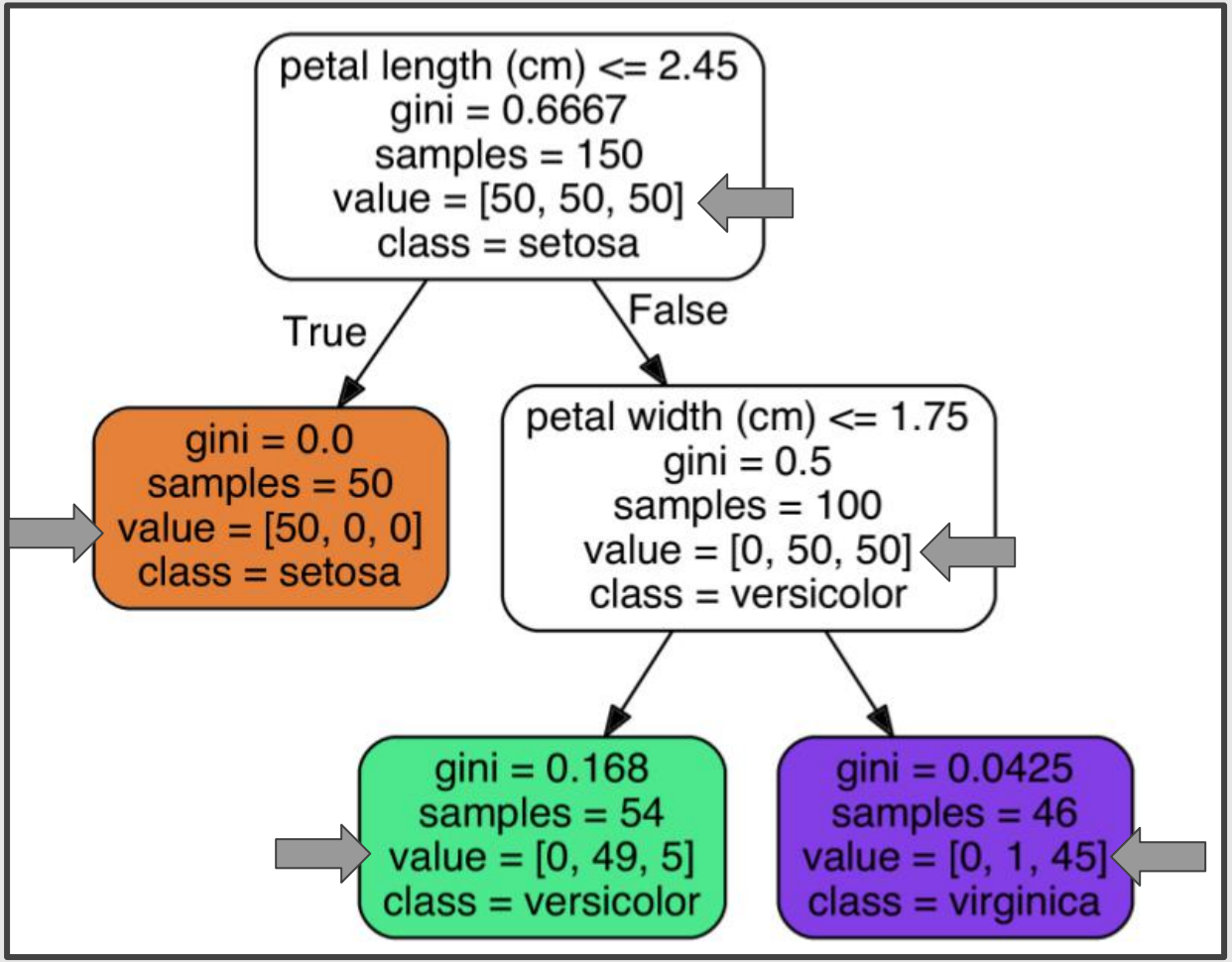

petal length (cm) <= 2.45
gini = 0.6667
samples = 150
value = [50, 50, 50]
class = setosa
True
False
gini = 0.0
samples = 50
value = [50, 0, 0]
class = setosa
petal width (cm) <= 1.75
gini = 0.5
samples = 100
value = [0, 50, 50]
class = versicolor
gini = 0.168
samples = 54
value = [0, 49, 5]
class = versicolor
gini = 0.0425
samples = 46
value = [0, 1, 45]
class = virginica



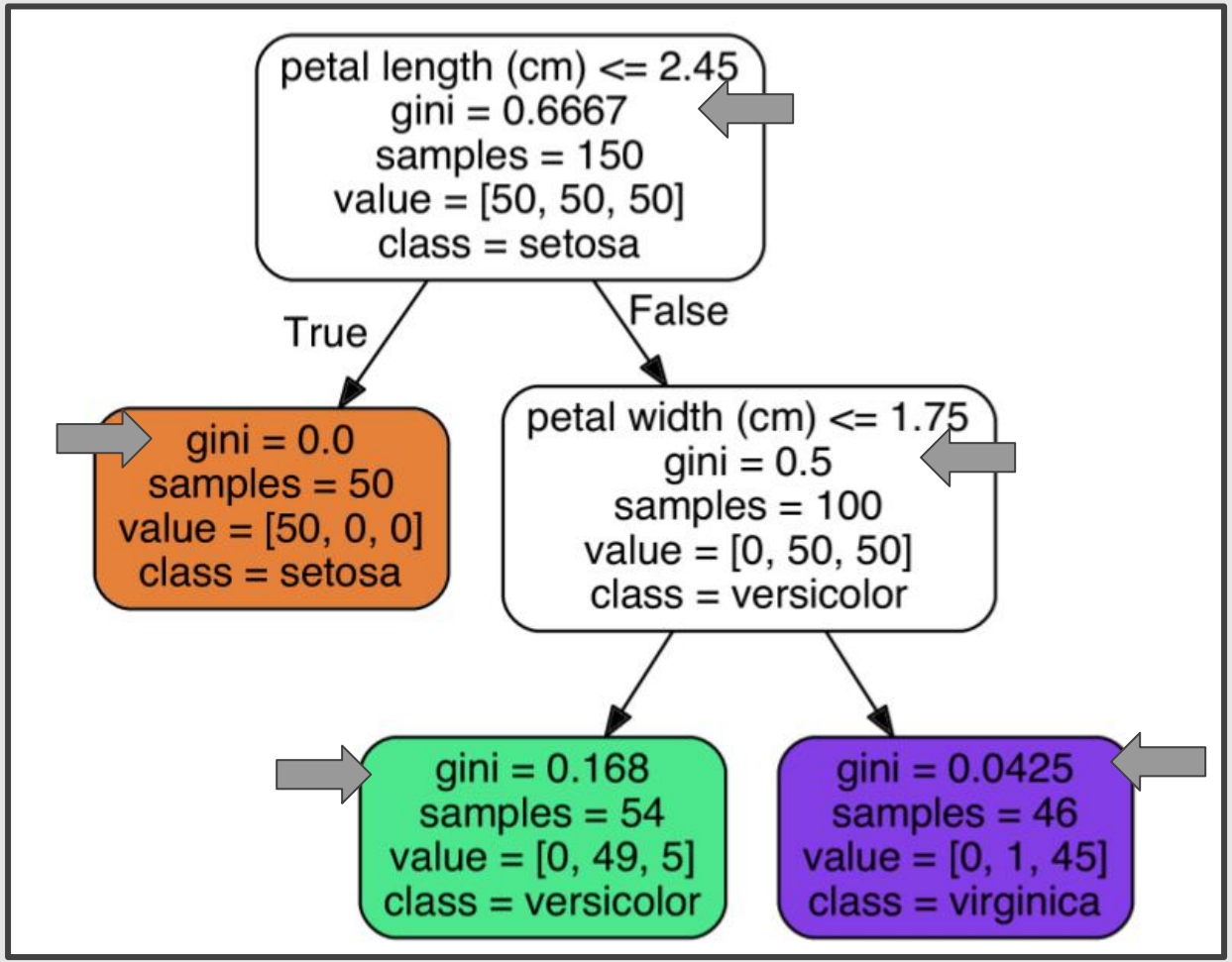
This node asks whether the flower's petal length is smaller than 2.45 cm



A **node's samples** attribute counts how many training instances it applies to.

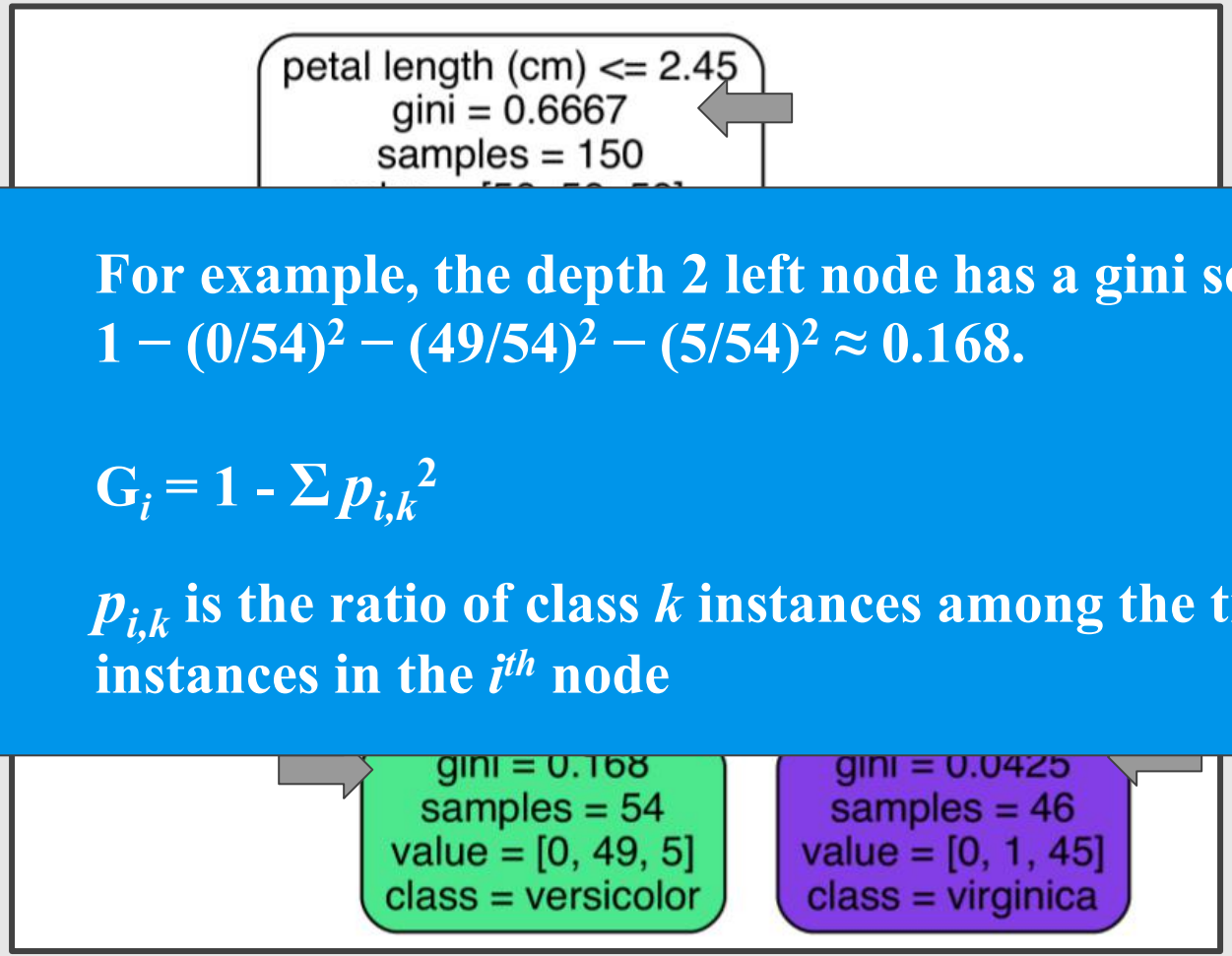


A **node's value** attribute tells you how many training instances of each class this node applies to.



A **node's gini** attribute measures its impurity.

"pure" (gini=0): all training instances belong to the same class.



For example, the depth 2 left node has a gini score equal to $1 - (0/54)^2 - (49/54)^2 - (5/54)^2 \approx 0.168$.

$$G_i = 1 - \Sigma p_{i,k}^2$$

$p_{i,k}$ is the ratio of class k instances among the training instances in the i^{th} node

i
sures
0): all
nces
same

The CART Algorithm

- Classification And Regression Tree (CART) algorithm.

The CART Algorithm

- Classification And Regression Tree (CART) algorithm.

- The idea is really quite simple: the algorithm first splits the training set in two subsets using a single feature k and a threshold t_k (e.g. “petal length $\leq$ 2.45 cm”).

The CART Algorithm

- Classification And Regression Tree (CART) algorithm.
- The idea is really quite simple: the algorithm first splits the training set in two subsets using a single feature k and a threshold t_k (e.g. “petal length $\leq$ 2.45 cm”).
- How does it choose k and t_k? The answer is that it searches for the pair (k, t_k) that produces the purest subsets (weighted by their size).

The CART Algorithm

$$J(k, t_k) = \frac{m_{\text{left}}}{m} G_{\text{left}} + \frac{m_{\text{right}}}{m} G_{\text{right}}$$

where $\begin{cases} G_{\text{left/right}} \text{ measures the impurity of the left/right subset,} \\ m_{\text{left/right}} \text{ is the number of instances in the left/right subset.} \end{cases}$

CART cost function for classification

The CART Algorithm

$$J(k, t_k) = \frac{m_{\text{left}}}{m} G_{\text{left}} + \frac{m_{\text{right}}}{m} G_{\text{right}}$$

where $\begin{cases} G_{\text{left/right}} \text{ measures the impurity of the left/right subset,} \\ m_{\text{left/right}} \text{ is the number of instances in the left/right subset.} \end{cases}$

CART cost function for classification

It stops recursing once it reaches the maximum depth (hyperparameter), or if it cannot find a split that will reduce impurity.

To be continued ...

References

Machine Learning Books

- Hands-On Machine Learning with Scikit-Learn and TensorFlow, Chap. 6 & 7
- Pattern Recognition and Machine Learning, Chap. 14
- Pattern Classification, Chap 8 & 9 (Sec. 9.5)