آکادمی یوتک

Machine Learning Course

Vahid Reza Khazaie

Classifiers

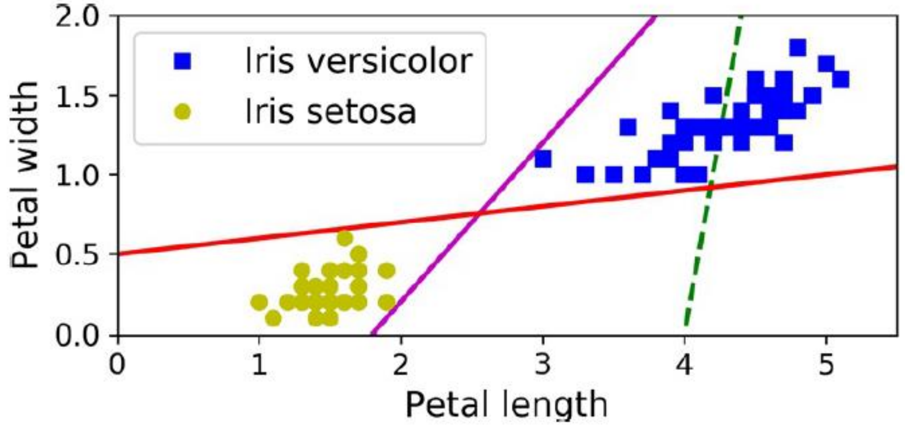
- **SVM**
- **Decision Tree**
- **Random Forrest**
- **Gaussian Naïve Bayes**

Support Vector Machine

- Perform linear or nonlinear **classification** or **regression**
- Well suited for classification of complex **small or medium-sized** datasets

Support Vector Machine

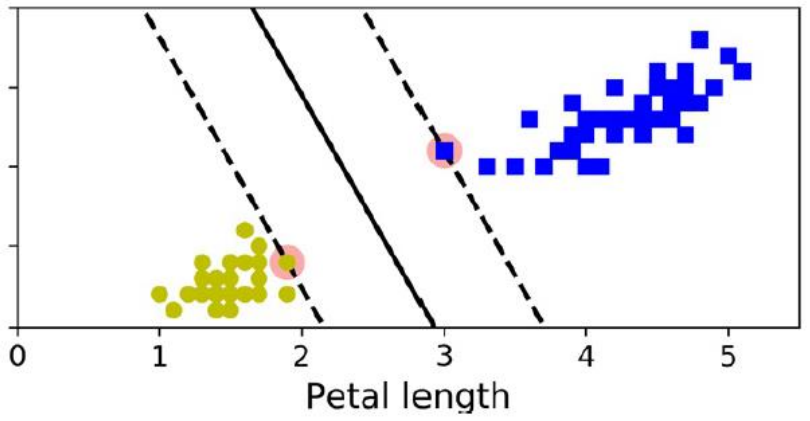
The model whose decision boundary is represented by the dashed line is so bad that it does not even separate the classes properly. The other two models work perfectly on this training set, but their decision boundaries come so close to the instances that these models will probably not perform as well on new instances.



Support Vector Machine

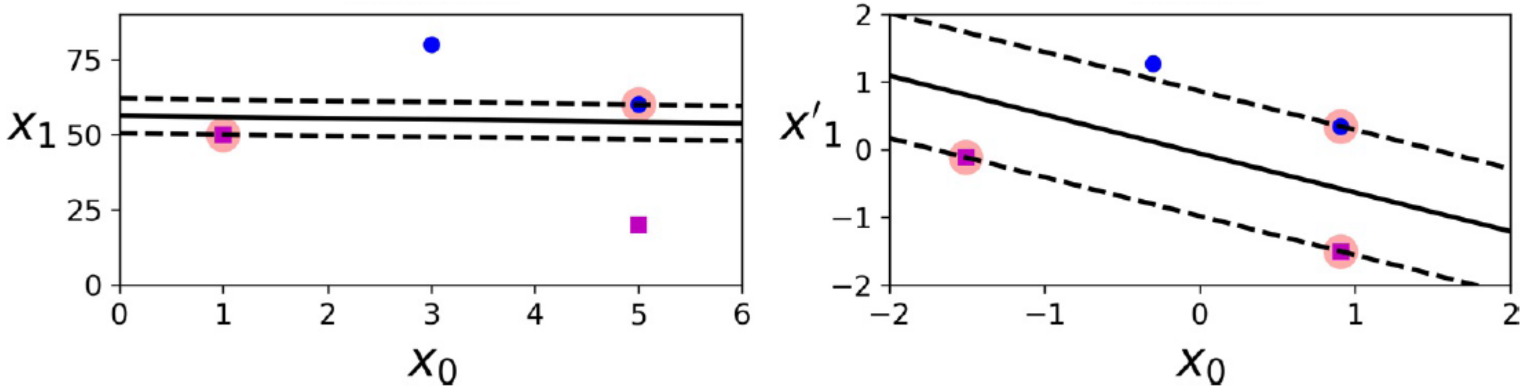
In contrast, the solid line in the plot represents the decision boundary of an SVM classifier; this line not only separates the two classes but also stays as far away from the closest training instances as possible.

This is called **large margin classification**.



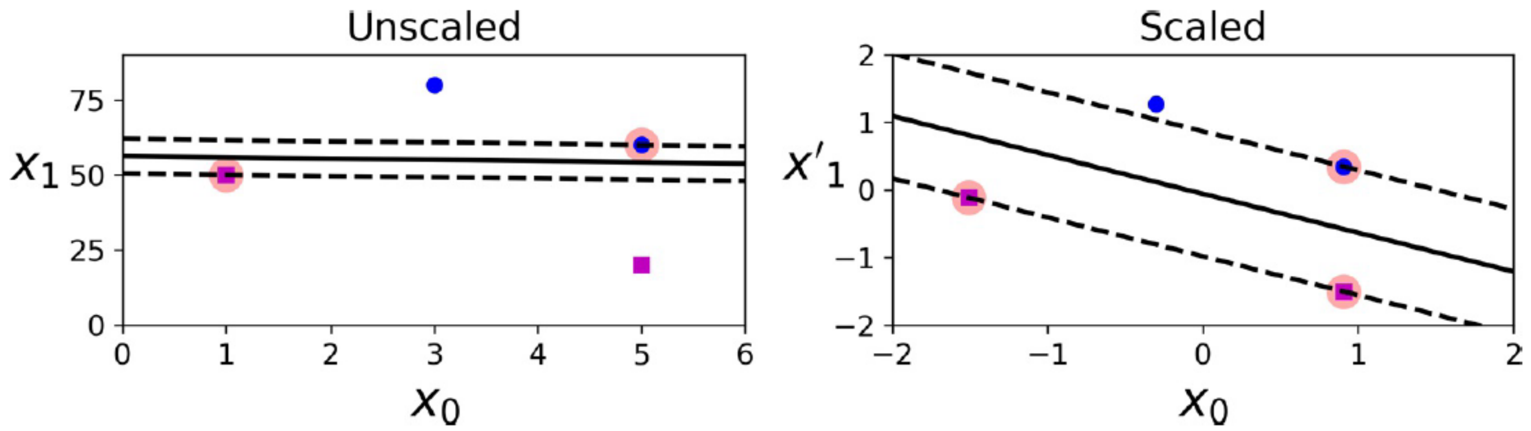
Support Vector Machine

Support Vectors



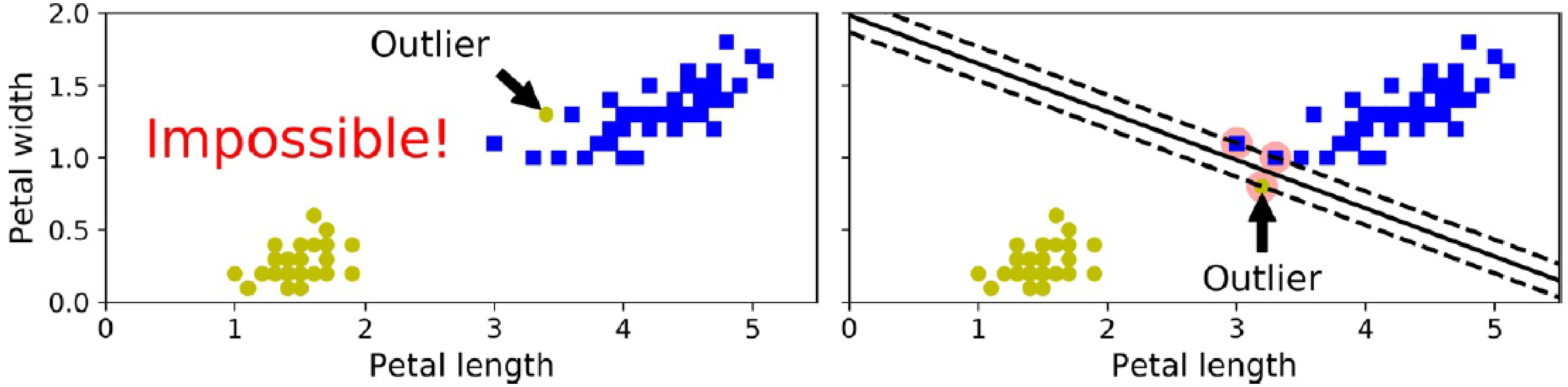
Support Vector Machine

Support Vectors



Support Vector Machine

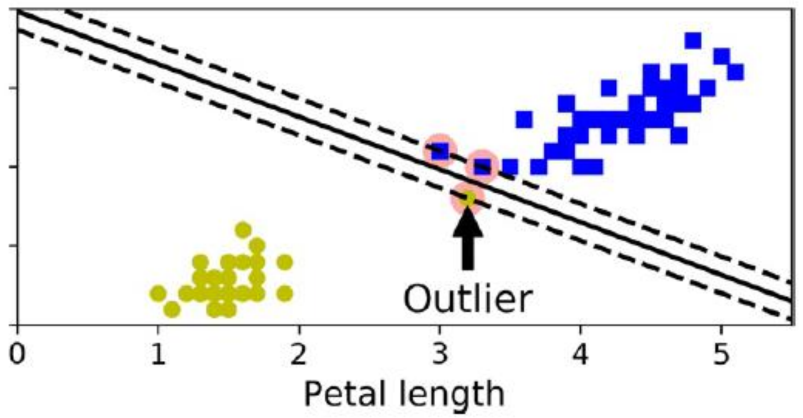
Soft Margin Vs. Hard Margin Classification



Support Vector Machine

Hard Margin Classification

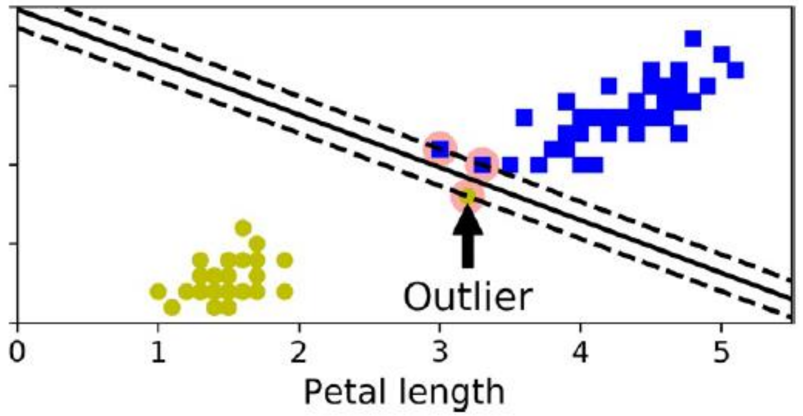
Strictly impose that all instances must be off the street and on the right side.



Support Vector Machine

Hard Margin Classification

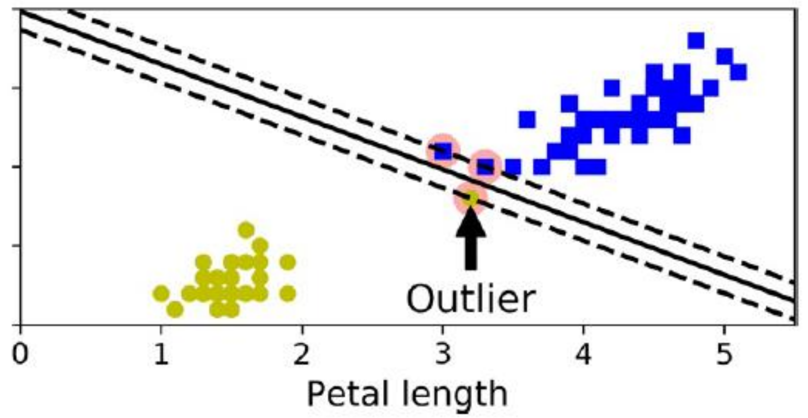
First, it only works if the data is linearly separable. Second, it is sensitive to outliers and it will probably not generalize as well.



Support Vector Machine

Soft Margin Classification

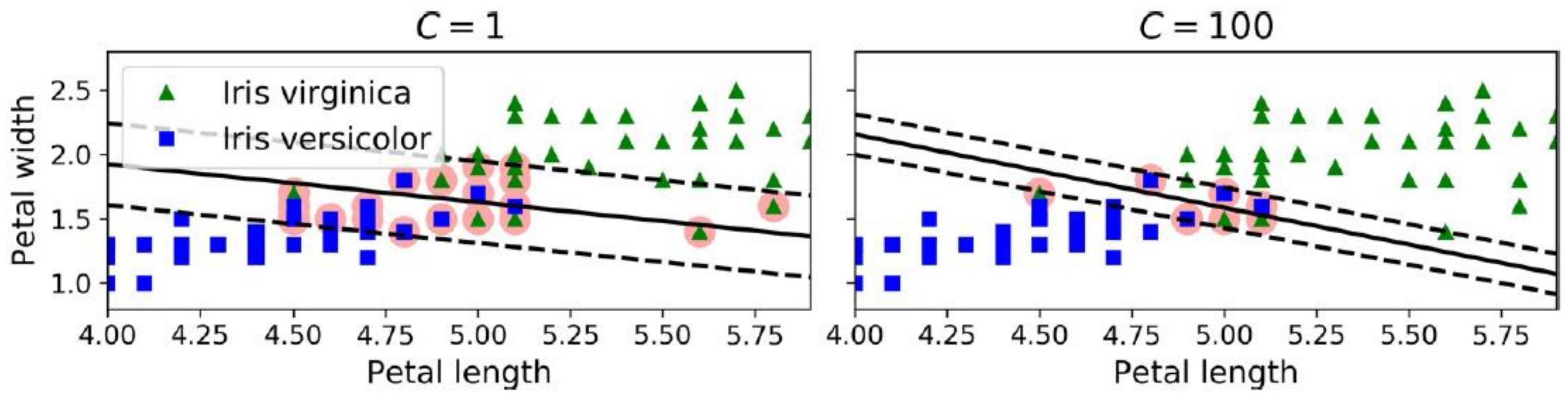
To avoid these issues, use a more flexible model. The objective is to find a good balance between keeping the street as large as possible and limiting the ***margin violations*** (i.e., instances that end up in the middle of the street or even on the wrong side). This is called *soft margin classification*.



Support Vector Machine

Margin and C parameter

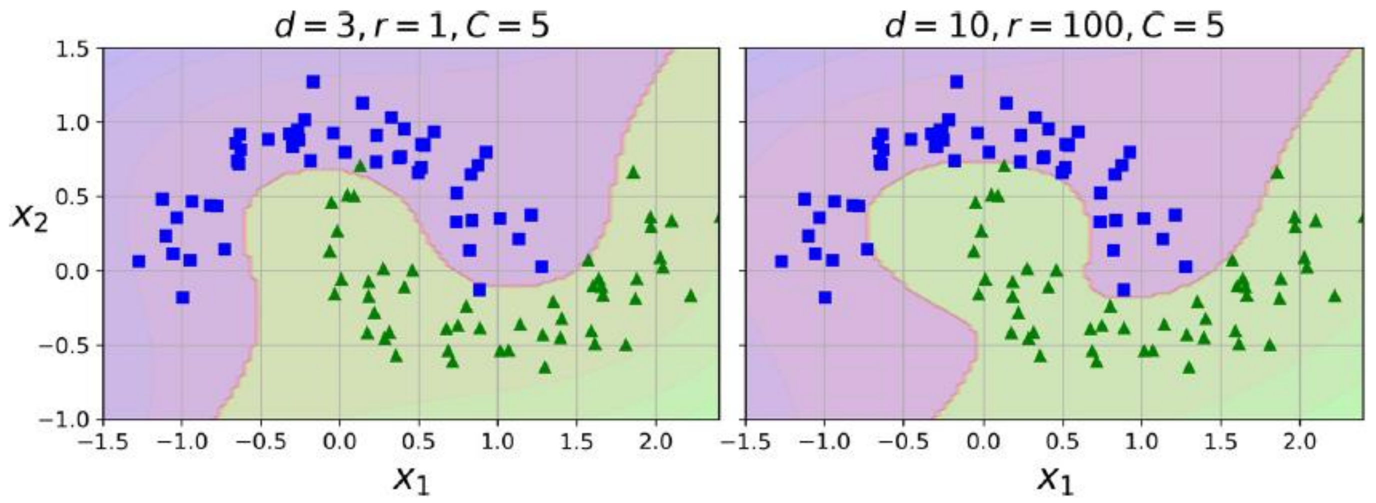
If we set it to a low value, then we end up with the model on the left. With a high value, we get the model on the right. Margin violations are bad. It’s usually better to have few of them. However, in this case the model on the left has a lot of margin violations but will probably generalize better.



Support Vector Machine

Nonlinear SVM Classification

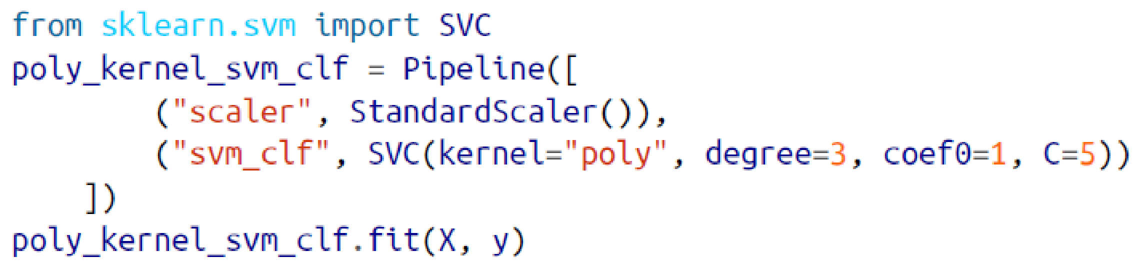
an SVM classifier using a third-degree polynomial kernel represented on the left. On the right is another SVM classifier using a 10th-degree polynomial kernel. Obviously, if your model is overfitting, you might want to reduce the polynomial degree.



Support Vector Machine

Nonlinear SVM Classification

The hyperparameter coef0 controls how much the model is influenced by high-degree polynomials versus low-degree polynomials.



```
from sklearn.svm import SVC
poly_kernel_svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("svm_clf", SVC(kernel="poly", degree=3, coef0=1, C=5))
    ])
poly_kernel_svm_clf.fit(X, y)
```

Support Vector Machine

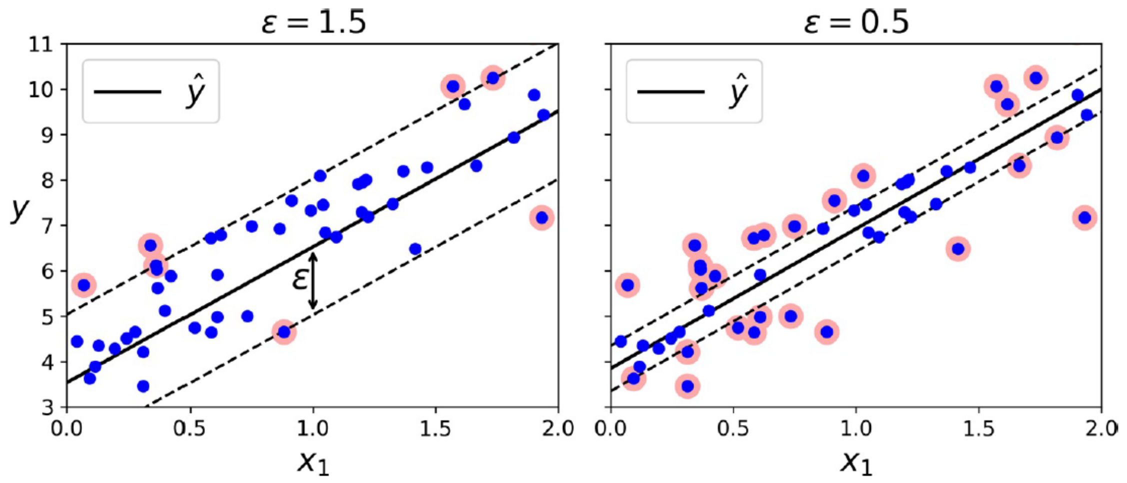
SVM Kernels

$$\begin{aligned}
\text{Linear:} \quad & K(\mathbf{a},\mathbf{b}) = \mathbf{a}^\intercal \mathbf{b} \\
\text{Polynomial:} \quad & K(\mathbf{a},\mathbf{b}) = (\gamma \mathbf{a}^\intercal \mathbf{b} + r)^d \\
\text{Gaussian RBF:} \quad & K(\mathbf{a},\mathbf{b}) = \exp\left(-\gamma \|\mathbf{a} - \mathbf{b}\|^2\right) \\
\text{Sigmoid:} \quad & K(\mathbf{a},\mathbf{b}) = \tanh\left(\gamma \mathbf{a}^\intercal \mathbf{b} + r\right)
\end{aligned}$$

Support Vector Machine

SVM Regression

SVM Regression tries to fit as many instances as possible *on* the street while limiting margin violations (i.e., instances *off* the street). The width of the street is controlled by a hyperparameter, ϵ. It is shown two linear SVM Regression models trained on some random linear data, one with a large margin (ϵ = 1.5) and the other with a small margin (ϵ = 0.5).



Decision Tree

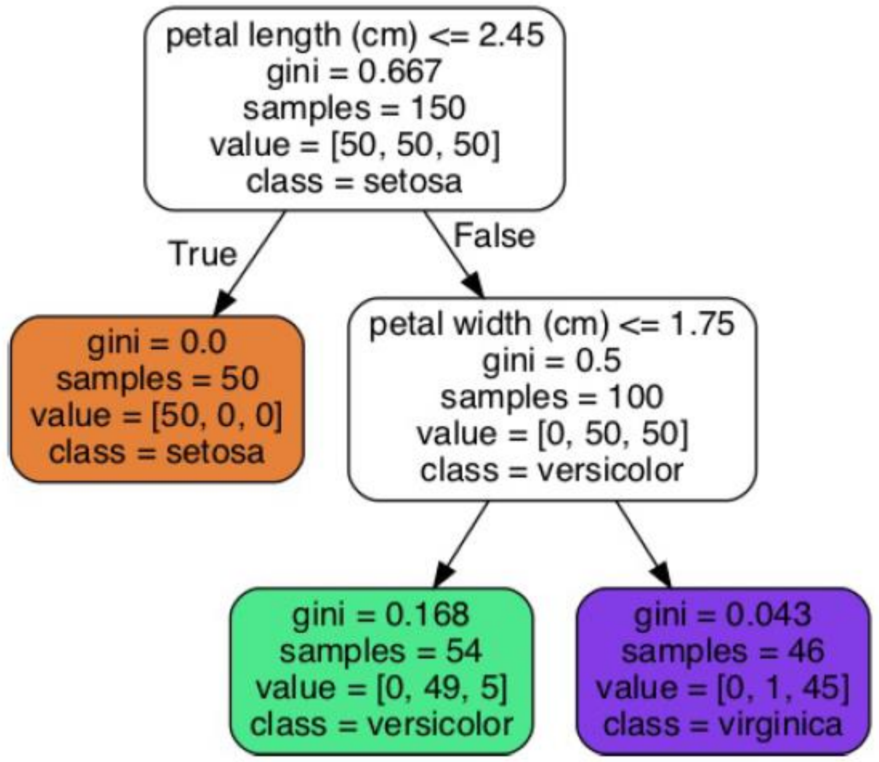
Like SVMs, *Decision Trees* are versatile Machine Learning algorithms that can perform both classification and regression tasks. They are powerful algorithms, capable of fitting complex datasets.

Decision Trees are also the fundamental components of Random Forests.

Decision Tree

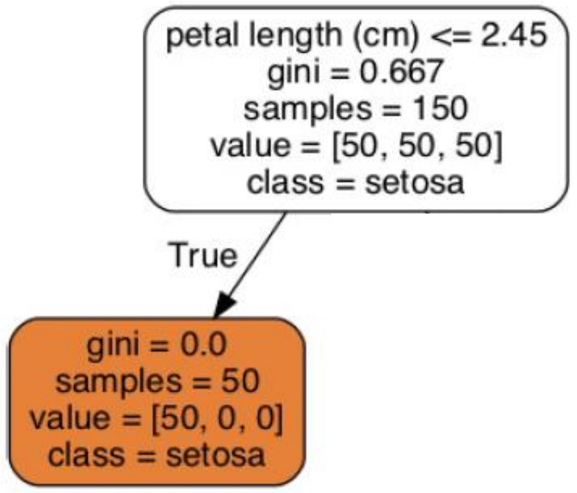
Decision Tree

One of the many qualities of Decision Trees is that they require very little data preparation. In fact, they don’t require feature scaling or centering at all.



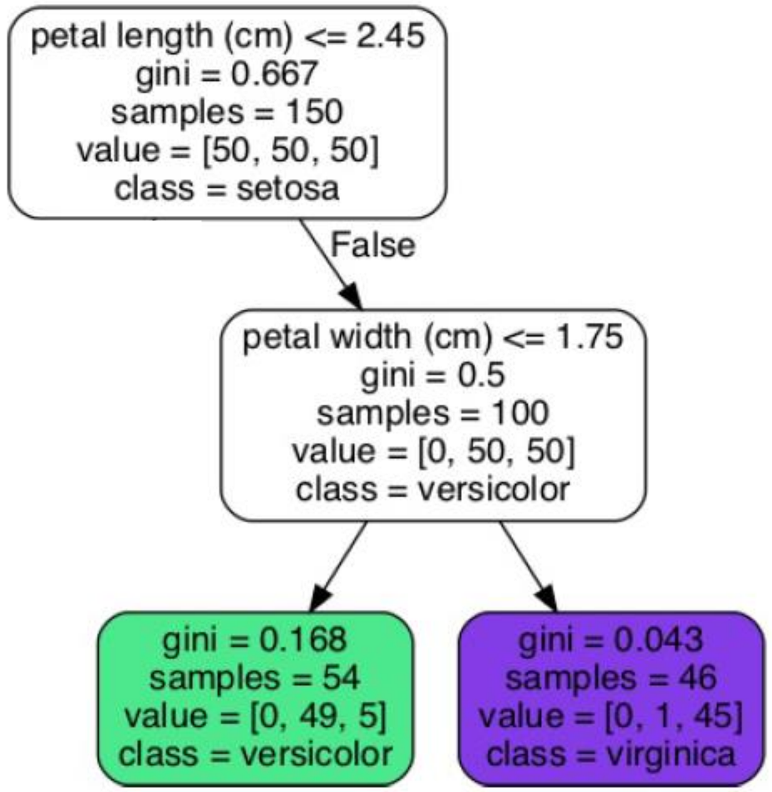
Decision Tree

Suppose you find an iris flower and you want to classify it. You start at the *root node* (depth 0, at the top): this node asks whether the flower's petal length is smaller than 2.45 cm. If it is, then you move down to the root's left child node (depth 1, left). In this case, it is a *leaf node* (i.e., it does not have any child nodes), so it does not ask any questions: simply look at the predicted class for that node, and the Decision Tree predicts that your flower is an *Iris setosa* (class=setosa).



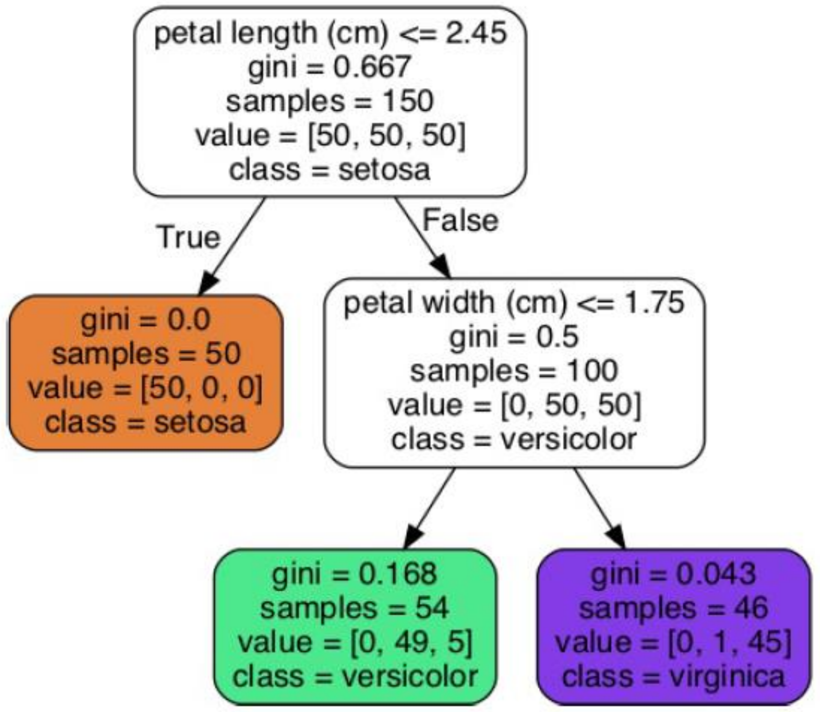
Decision Tree

Now suppose you find another flower, and this time the petal length is greater than 2.45 cm. You must move down to the root's right child node (depth 1, right), which is not a leaf node, so the node asks another question: is the petal width smaller than 1.75 cm? If it is, then your flower is most likely an *Iris versicolor* (depth 2, left). If not, it is likely an *Iris virginica* (depth 2, right).



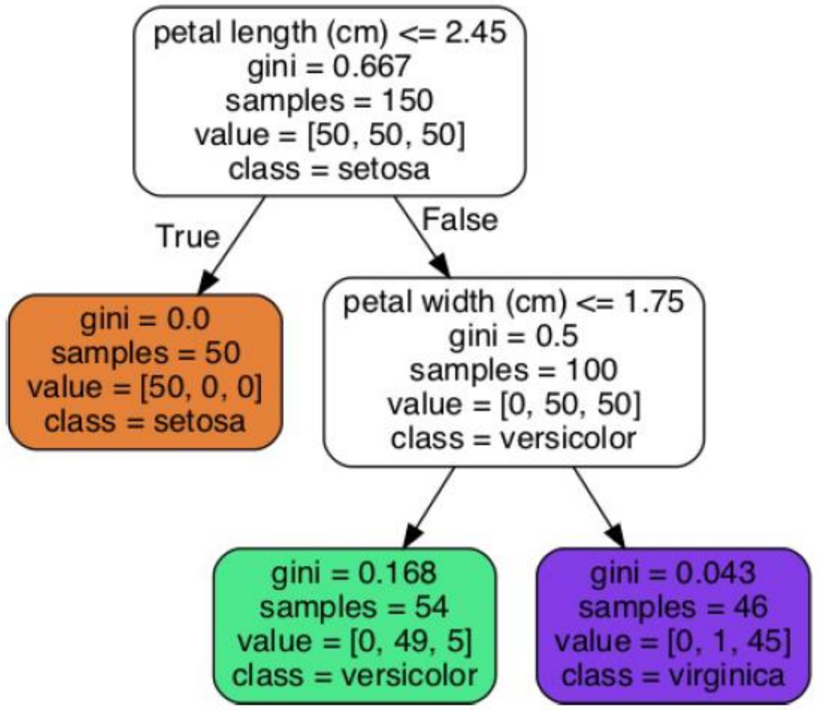
Decision Tree

A node's samples attribute counts how many training instances it applies to. For example, 100 training instances have a petal length greater than 2.45 cm (depth 1, right), and of those 100, 54 have a petal width smaller than 1.75 cm (depth 2, left). A node's value attribute tells you how many training instances of each class this node applies to: for example, the bottom-right node applies to 0 *Iris setosa*, 1 *Iris versicolor*, and 45 *Iris virginica*.



Decision Tree

Finally, a node's gini attribute measures its *impurity*: a node is "pure" (gini=0) if all training instances it applies to belong to the same class. For example, since the depth-1 left node applies only to *Iris setosa* training instances, it is pure and its gini score is 0. It is shown how the training algorithm computes the gini score *G* of the i^{th} node. The depth-2 left node has a gini score equal to 1 – (0/54) – (49/54) – (5/54) ≈ 0.168



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

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

Decision Tree

Classification and Regression Tree (CART) algorithm

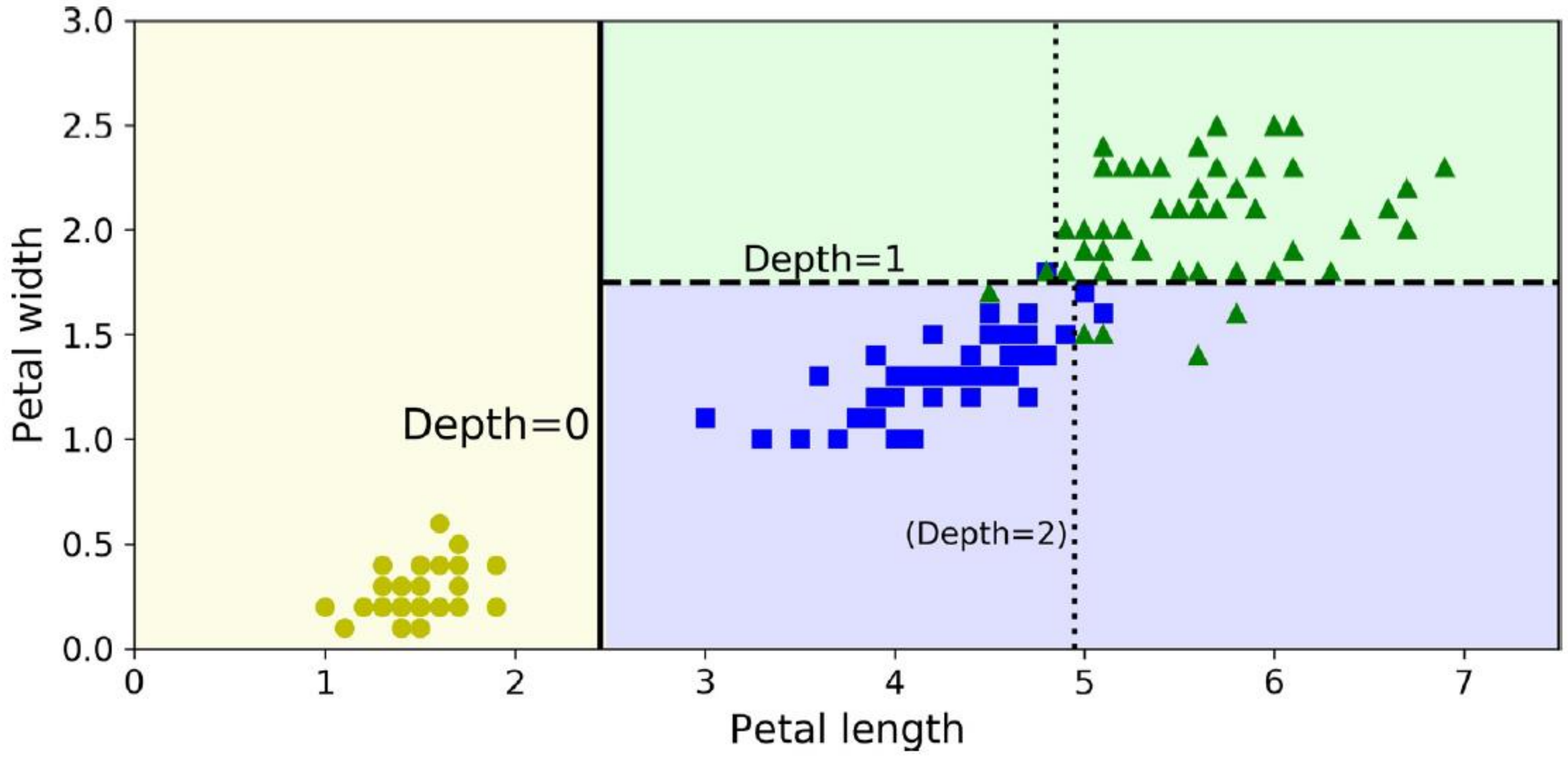
The algorithm works by first splitting the training set into two subsets using a single feature *k* and a threshold *t* (e.g., “petal length ≤ 2.45 cm”). How does it choose *k* and *t* ? It searches for the pair (*k*, *t*) that produces the purest subsets (weighted by their size).
Here is the cost function that the algorithm tries to minimize:

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

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

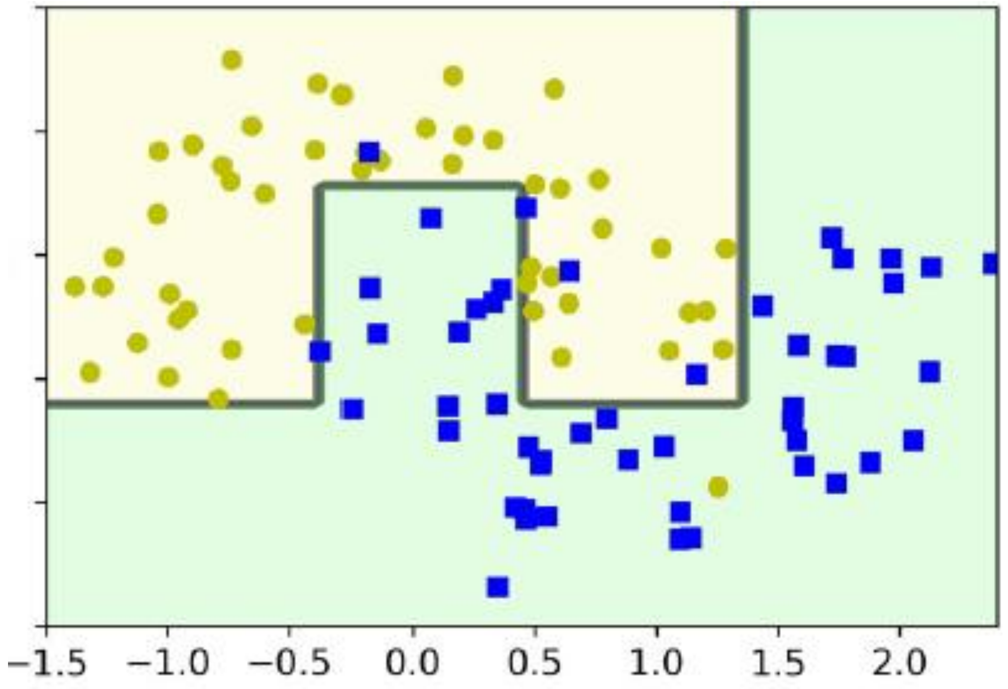
Decision Tree

Decision Boundary



Decision Tree

Decision Boundary



Random Forrest

Bagging*:* Using the same training algorithm for every predictor and train them on different random subsets of the training set with replacement.

The Random Forest algorithm introduces extra randomness when growing decision trees.

This algorithm instead of searching for the very best feature when splitting a node, searches for the best feature among a random subset of features. The algorithm results in greater tree diversity, which (again) trades a higher bias for a lower variance, generally yielding an overall better model.

Gaussian Naïve Bayes

Bayes’ Theorem

One of the easiest ways of selecting the most probable hypothesis given the data that we have that we can use as our prior knowledge about the problem. Bayes' Theorem provides a way that we can calculate the probability of a hypothesis given our prior knowledge. Bayes' Theorem is stated as:

$$P(h|d) = \frac{P(d|h) \times P(h)}{P(d)}$$

Where:

- $P(h|d)$ is the probability of hypothesis h given the data d. This is called the posterior probability.
- $P(d|h)$ is the probability of data d given that the hypothesis h was true.
- $P(h)$ is the probability of hypothesis h being true (regardless of the data). This is called the prior probability of h.
- $P(d)$ is the probability of the data (regardless of the hypothesis).

Gaussian Naïve Bayes

Naive Bayes Classier

The representation for naive Bayes is probabilities. A list of probabilities is stored to le for a learned naive Bayes model. This includes:

- Class Probabilities: The probabilities of each class in the training dataset.
- Conditional Probabilities: The conditional probabilities of each input value given each class value.

It is called naive Bayes because the calculation of the probabilities for each hypothesis are simplified to make their calculation tractable. They are assumed to be conditionally independent given the target value.

This is a very strong assumption that is most unlikely in real data, i.e. that the attributes do not interact. Nevertheless, the approach performs surprisingly well on data where this assumption does not hold.

Gaussian Naïve Bayes

Gaussian Naive Bayes Model

Probabilities of new x values are calculated using the Gaussian Probability Density Function (PDF). When making predictions these parameters can be plugged into the Gaussian PDF with a new input for the variable, and in return the Gaussian PDF will provide an estimate of the probability of that new input value for that class.

$$mean(x) = \frac{1}{n} \times \sum_{i=1}^{n} x_i$$

$$StandardDeviation(x) = \sqrt{\frac{1}{n} \times \sum_{i=1}^{n} (x_i - mean(x))^2}$$

$$pdf(x, mean, sd) = \frac{1}{\sqrt{2 \times \pi} \times sd} \times e^{-\left(\frac{(x-mean)^2}{2 \times sd^2}\right)}$$