Machine Learning for Bioinformatics

Supervised learning

Irina Mohorianu (CSCI)

What is Machine Learning?

Machine Learning (ML) is generating abstract hypotheses (models) based on data which can be later used predict results on new data.

Types of ML:

[1] Unsupervised – Tue lecture
[2] Supervised – Thu lecture
[3] Semi-supervised
[4] Reinforcement learning

ML supervised. Outline

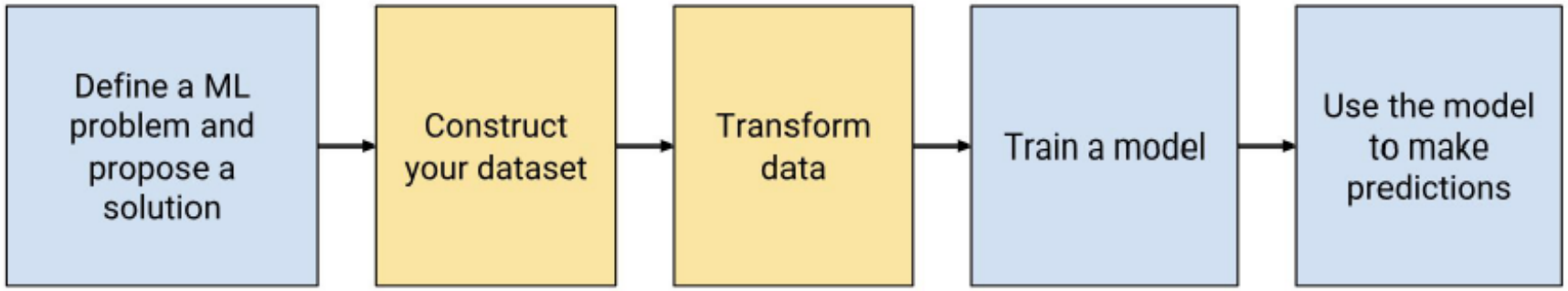
[1] Partitioning the data

[2] K nearest neighbours

[3] Decision Trees. Random Forests

[4] Support Vector Machines

Partitioning the data



To assess the efficiency of a classifier we need: a **training dataset** and a **testing dataset**.

Dev (development) set – dataset used to tune parameters, select features, and perform other decisions related to the classifier. Also named hold-out cross validation set.

Test dataset – used to evaluate the performance of the classifier, but not for any decisions regarding what learning algorithm or parameters to use.

The train, dev and test datasets should share the same statistical properties, and be derived from the same underlying distribution.

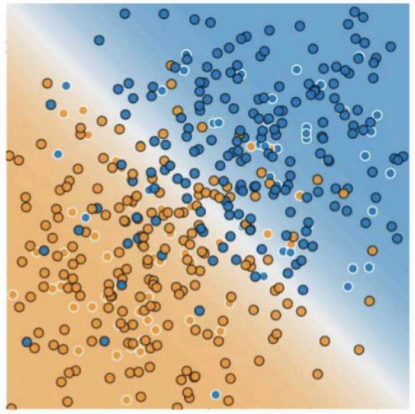
Partitioning the data

The dev set should be large enough to detect differences between algorithms that you are trying out.

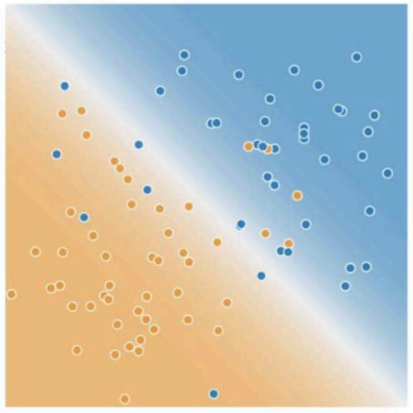
if classifier A has an accuracy of 95.0% and classifier B has an accuracy of 95.1%, then a dev set of 100 examples would not be able to detect this 0.1% difference.

Dev sets with sizes from 1,000 to 10,000 examples are common. With 10,000 examples, there is a good chance of detecting an improvement of 0.1%

There is no need to have excessively large test sets beyond what is needed to evaluate the performance of your algorithms. For example 30% is large in the big data world



Training Data

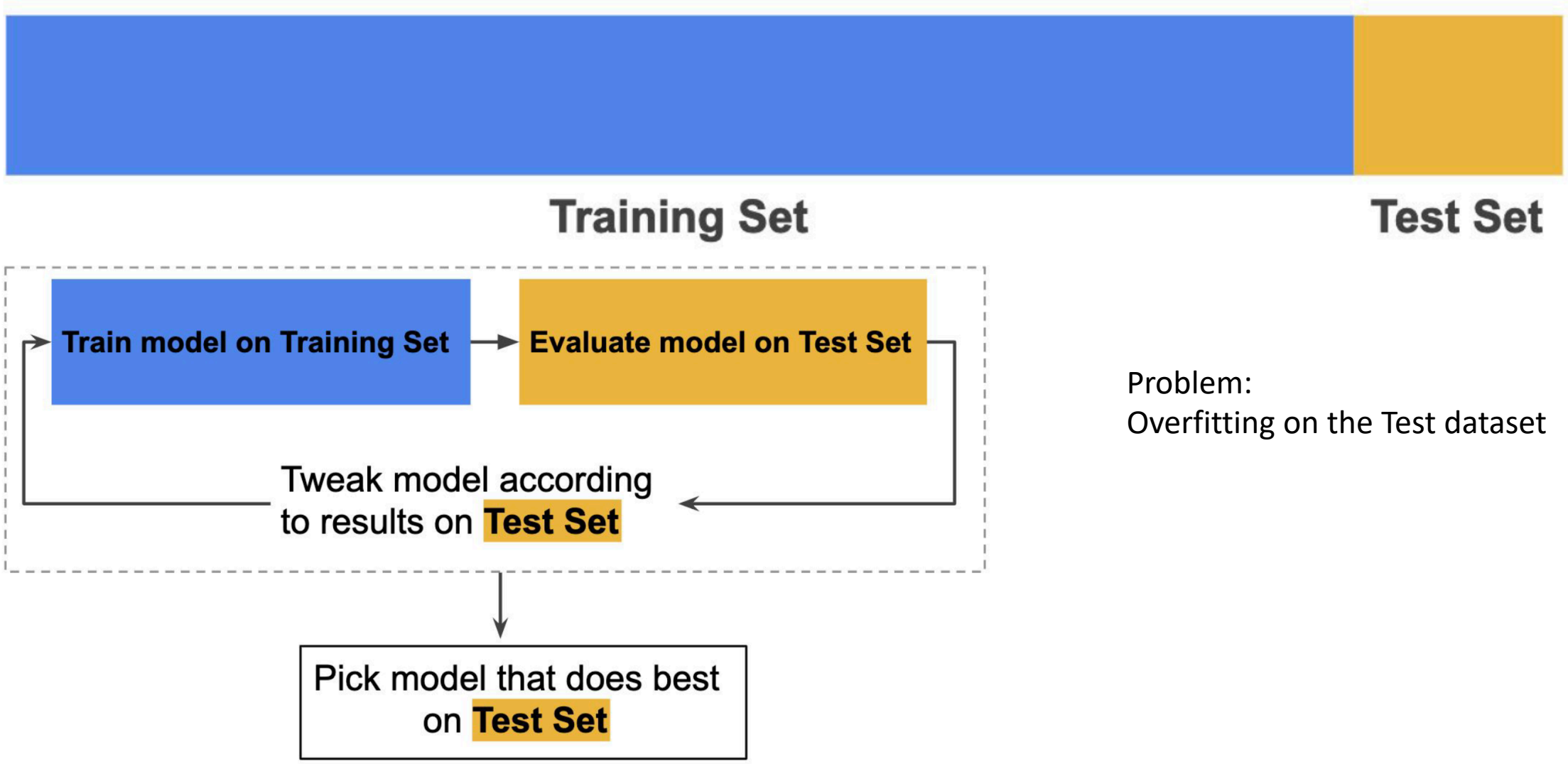


Test Data

More training data is better > better confidence intervals

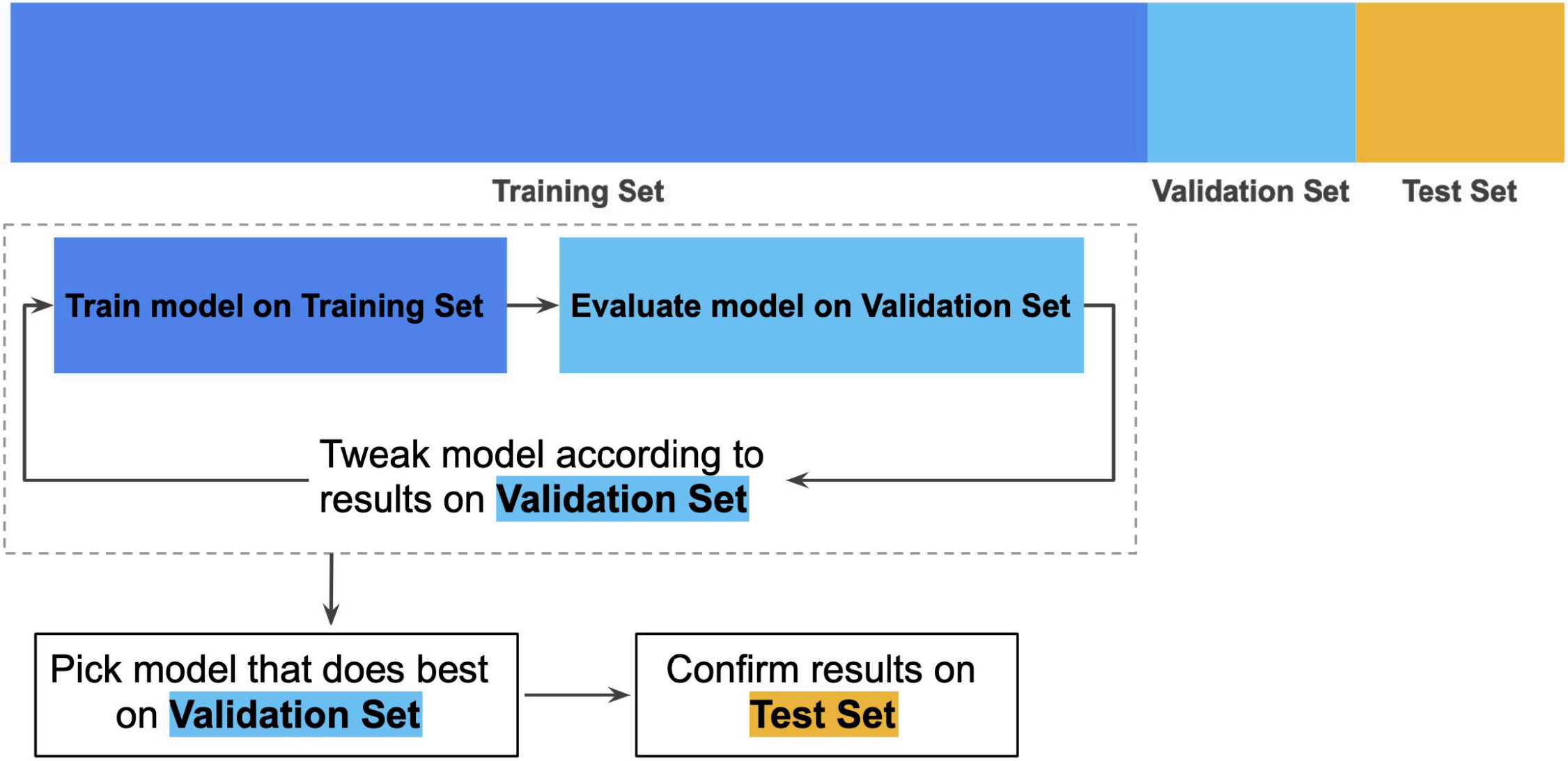
For a small dataset a cross-validation is preferred.

Partitioning the data



Problem:
Overfitting on the Test dataset

Partitioning the data



Evaluating the ML approach

sensitivity, recall, hit rate, or true positive rate (TPR)

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{P}} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} = 1 - \mathrm{FNR}$$

specificity, selectivity or true negative rate (TNR)

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{N}} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} = 1 - \mathrm{FPR}$$

precision or positive predictive value (PPV)

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} = 1 - \mathrm{FDR}$$

negative predictive value (NPV)

$$\mathrm{NPV} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FN}} = 1 - \mathrm{FOR}$$

miss rate or false negative rate (FNR)

$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{P}} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TP}} = 1 - \mathrm{TPR}$$

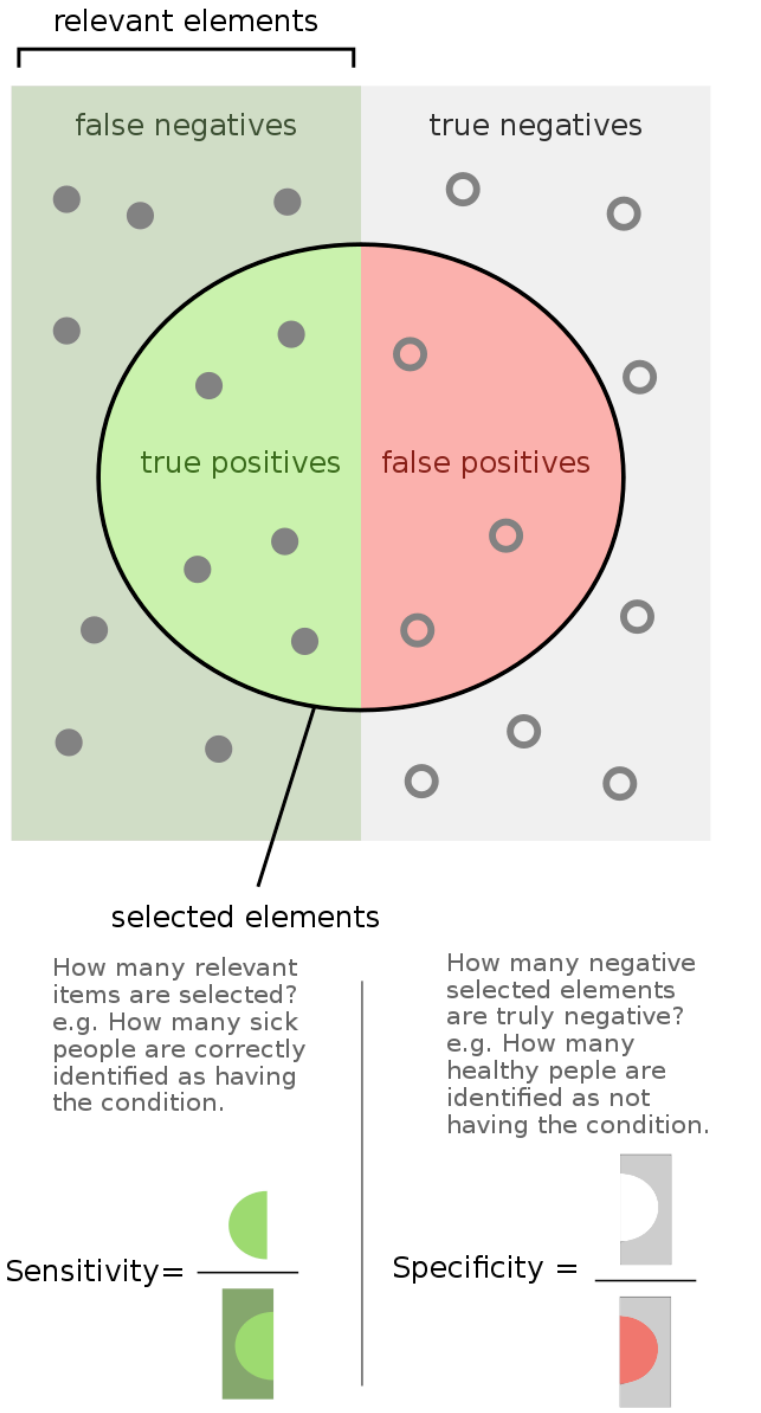
fall-out or false positive rate (FPR)

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{N}} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} = 1 - \mathrm{TNR}$$

false discovery rate (FDR)

$$\mathrm{FDR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TP}} = 1 - \mathrm{PPV}$$

https://en.wikipedia.org/wiki/Sensitivity_and_specificity

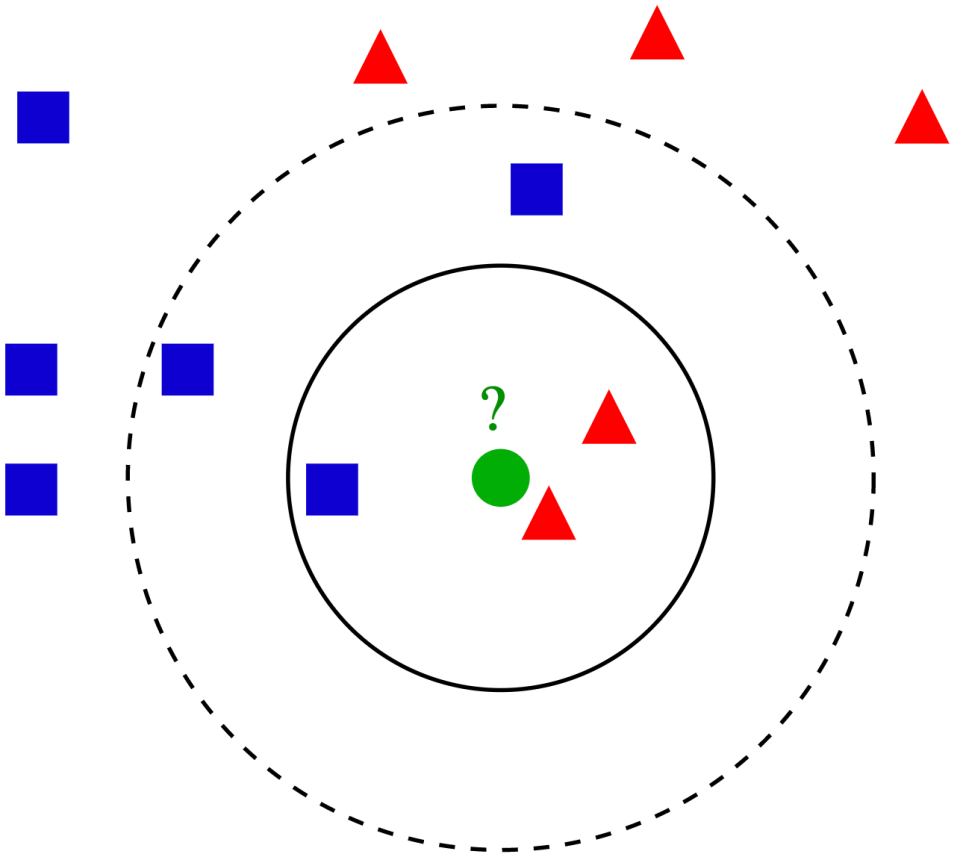


Evaluating the ML approach

		True condition				
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR−}}$	F_1 score = $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$
		False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$		

https://en.wikipedia.org/wiki/Sensitivity_and_specificity

K nearest neighbours

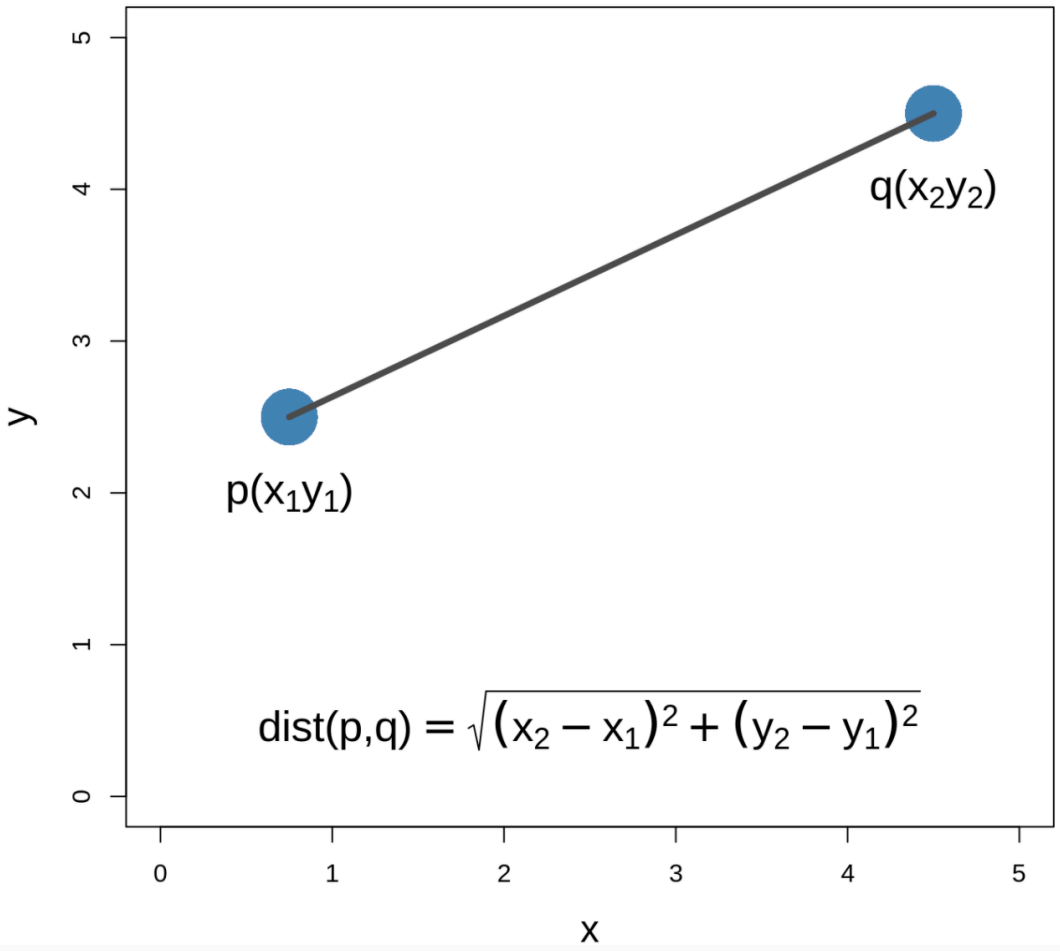


We are working with two classes:
[a] blue squares
[b] red triangles

Hyper - parameters for the knn:
[a] the number of neighbours
[b] the distance

KNN. Distances

$$distance\,(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$$



Manhattan distance:

$$distance\,(p,q) = \sum_{i=1}^{n}|p_i - q_i|$$

There are other metrics to measure the distance between observations. For example, the Minkowski distance is a generalization of the Euclidean and Manhattan distances and is defined as

Minkowski distance:

$$distance\,(p,q) = \sqrt[p]{\sum_{i=1}^{n}(p_i - q_i)^p}$$

where p>0 (Han, Pei, and Kamber 2011). When p=2 the Minkowski distance is the Euclidean distance and when q=1 it is the Manhattan distance

Alternative distances are correlation based distances.

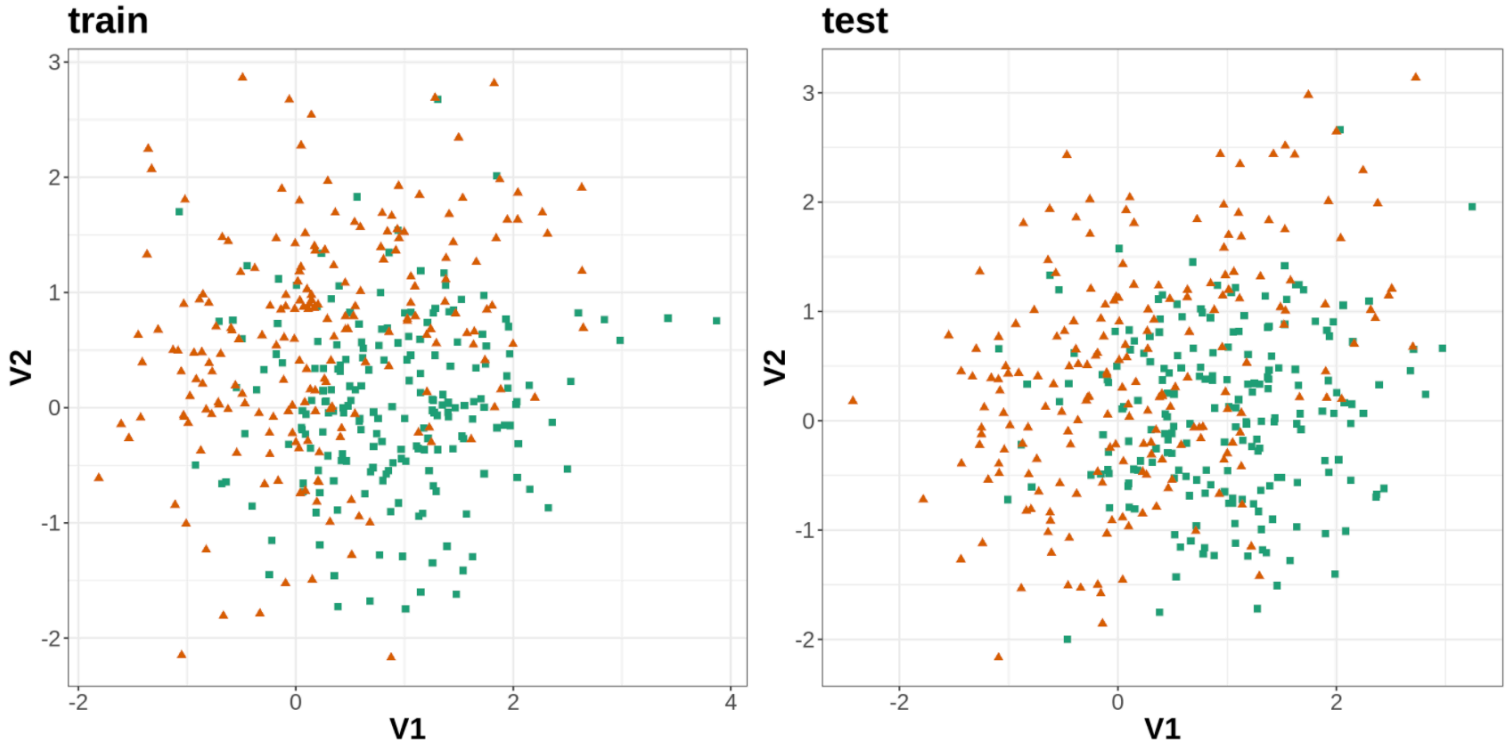
K nearest neighbours

Main points regarding the knn

[a] the knn function in R
plotting decision boundaries

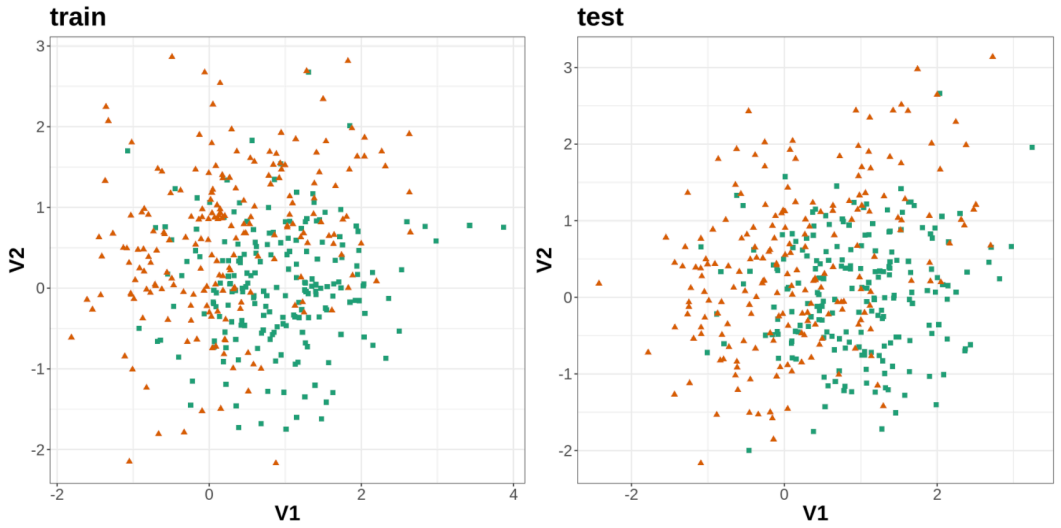
[b] bias-variance trade-off

[c] choosing the optimum value of *k*



Randomly generated data comprising two classes.

K nearest neighbours



The knn function in R has the following parameters:

train : matrix or data frame of training set cases.
test : matrix or data frame of test set cases.
cl : factor of true classifications of training set

k : number of neighbours considered.

l : minimum vote for definite decision, otherwise doubt. (More precisely, less than k-l dissenting votes are allowed, even if k is increased by ties.)

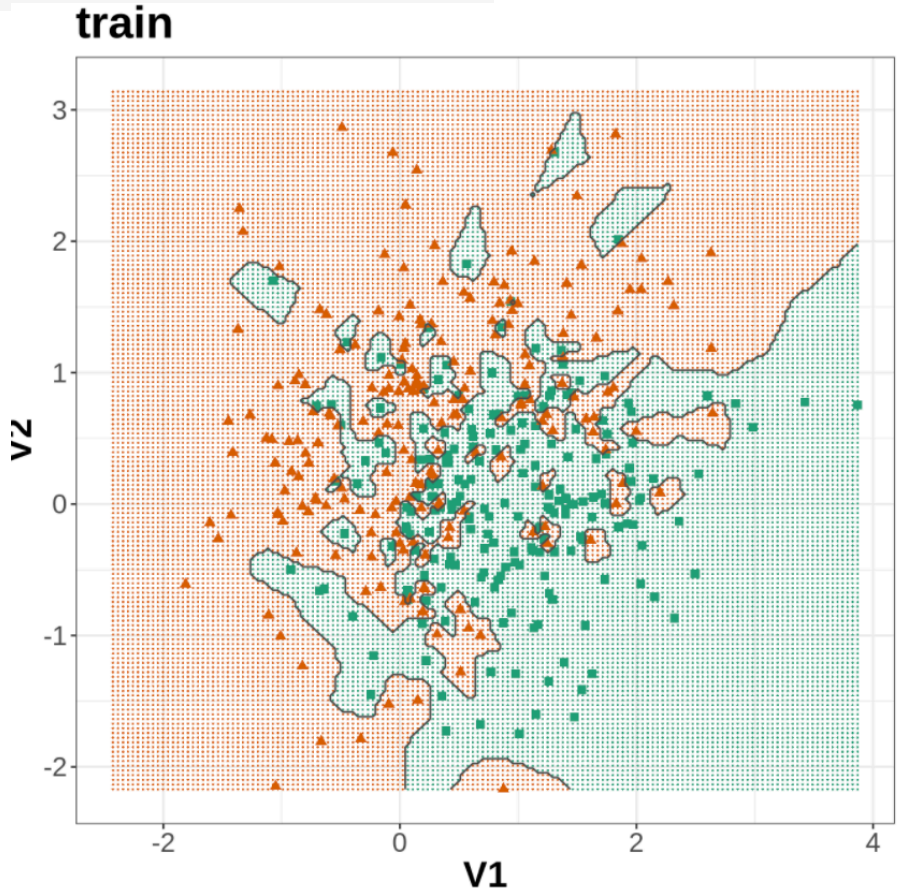
prob : If this is true, the proportion of the votes for the winning class are returned as attribute prob.

use.all : controls handling of ties. If true, all distances equal to the kth largest are included. If false, a random selection of distances equal to the kth is chosen to use exactly k neighbours.

K nearest neighbours. Choosing k

```
knn1train <- class::knn(train=xtrain, test=xtrain, cl=ytrain, k=1)
confusionMatrix(knn1train, as.factor(ytrain))
```

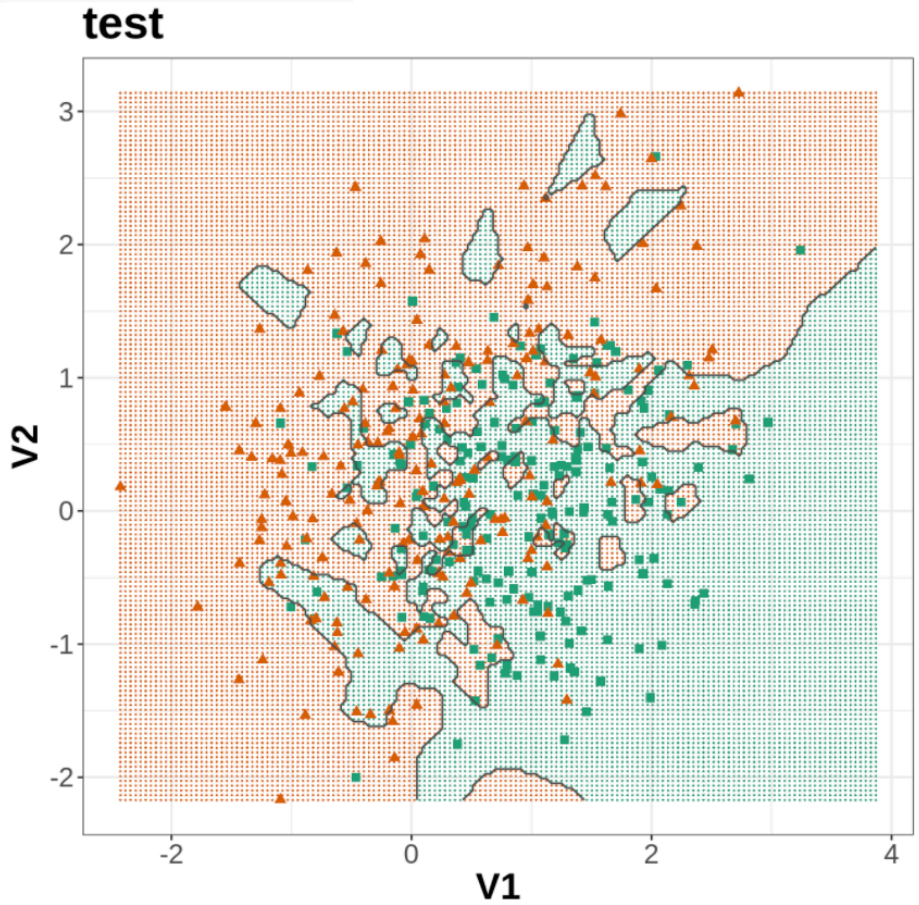
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 200   0
##          1   0 200
##
##                Accuracy : 1
##                  95% CI : (0.9908, 1)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
```



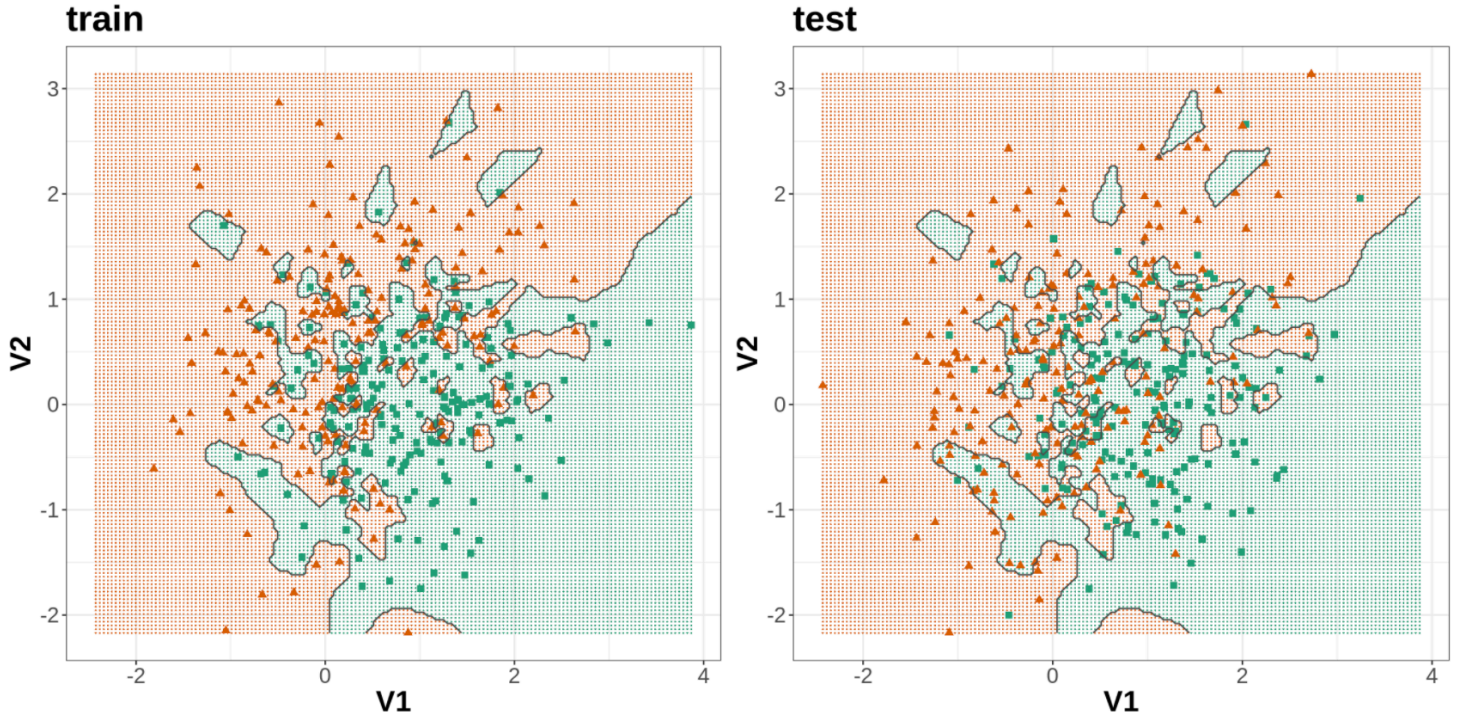
K nearest neighbours

```r
knn1test <- class::knn(train=xtrain, test=xtest, cl=ytrain, k=1)
confusionMatrix(knn1test, as.factor(ytest))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 131  81
##          1  69 119
##
##                Accuracy : 0.625
##                  95% CI : (0.5755, 0.6726)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 3.266e-07
```



K nearest neighbours

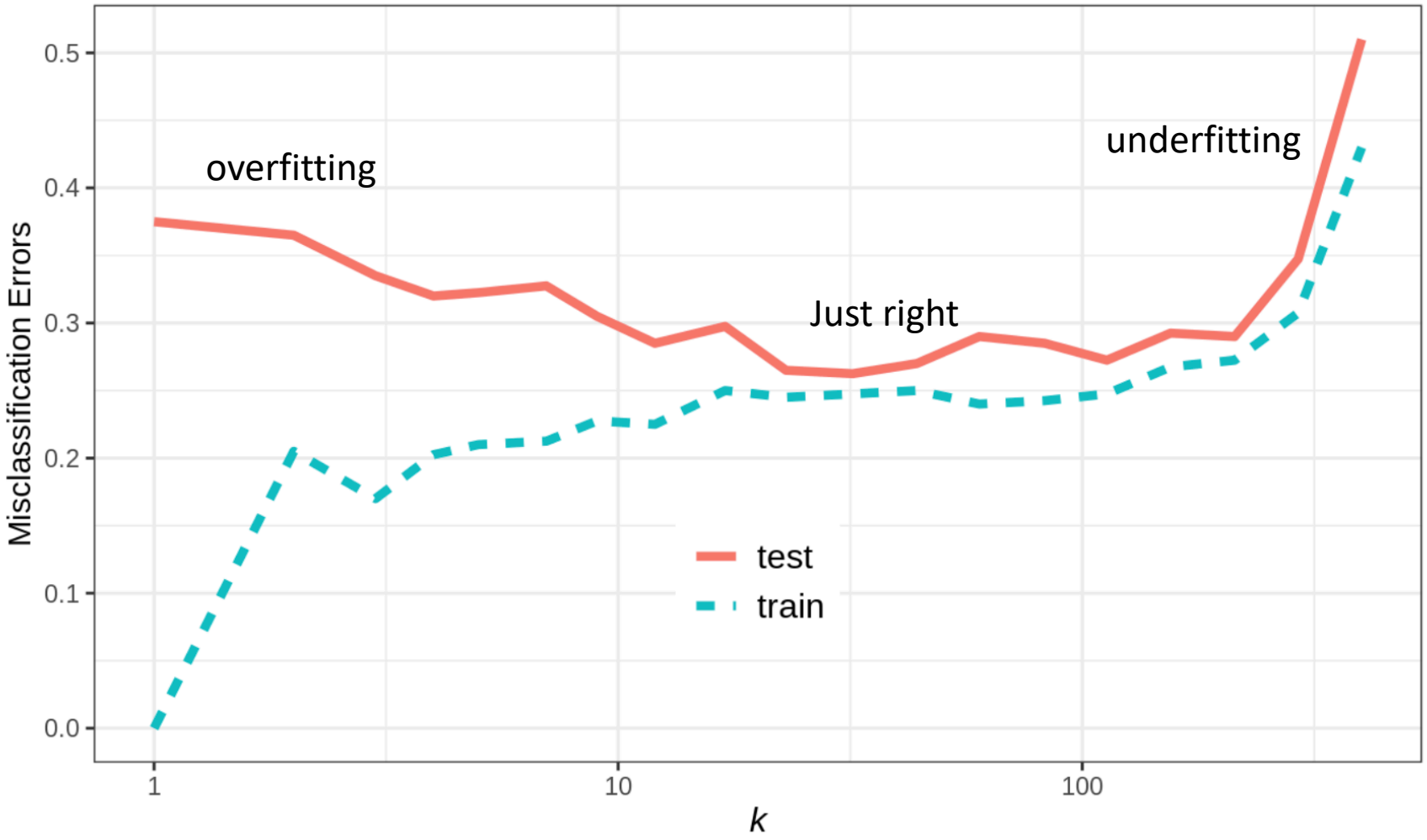


The **bias–variance trade-off** is the problem of simultaneously minimizing two sources of error that prevent supervised learning algorithms from generalizing beyond their training set:

The **bias** is error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting).

The **variance** is error from sensitivity to small fluctuations in the training set. High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs (overfitting).

K nearest neighbours



Misclassification errors as a function of neighbourhood size.

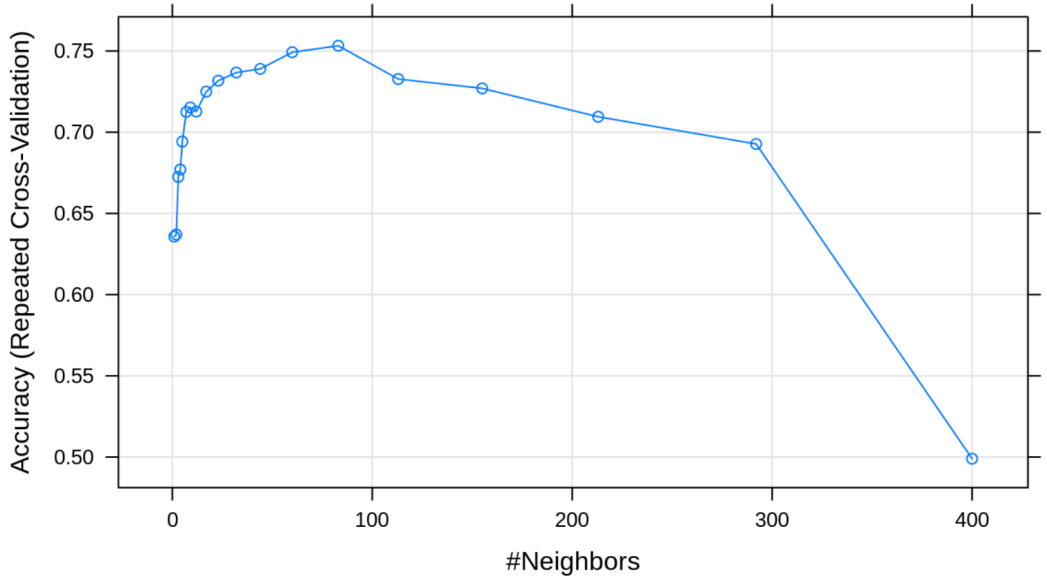
K nearest neighbours

k	Accuracy	Kappa
1	0.63575	0.2715
2	0.63700	0.2740
3	0.67250	0.3450
4	0.67700	0.3540
5	0.69425	0.3885
7	0.71250	0.4250
9	0.71525	0.4305
12	0.71275	0.4255
17	0.72500	0.4500
23	0.73175	0.4635
32	0.73675	0.4735
44	0.73900	0.4780
60	0.74925	0.4985
83	0.75325	0.5065
113	0.73275	0.4655
155	0.72700	0.4540
213	0.70950	0.4190
292	0.69275	0.3855
400	0.49900	-0.0020

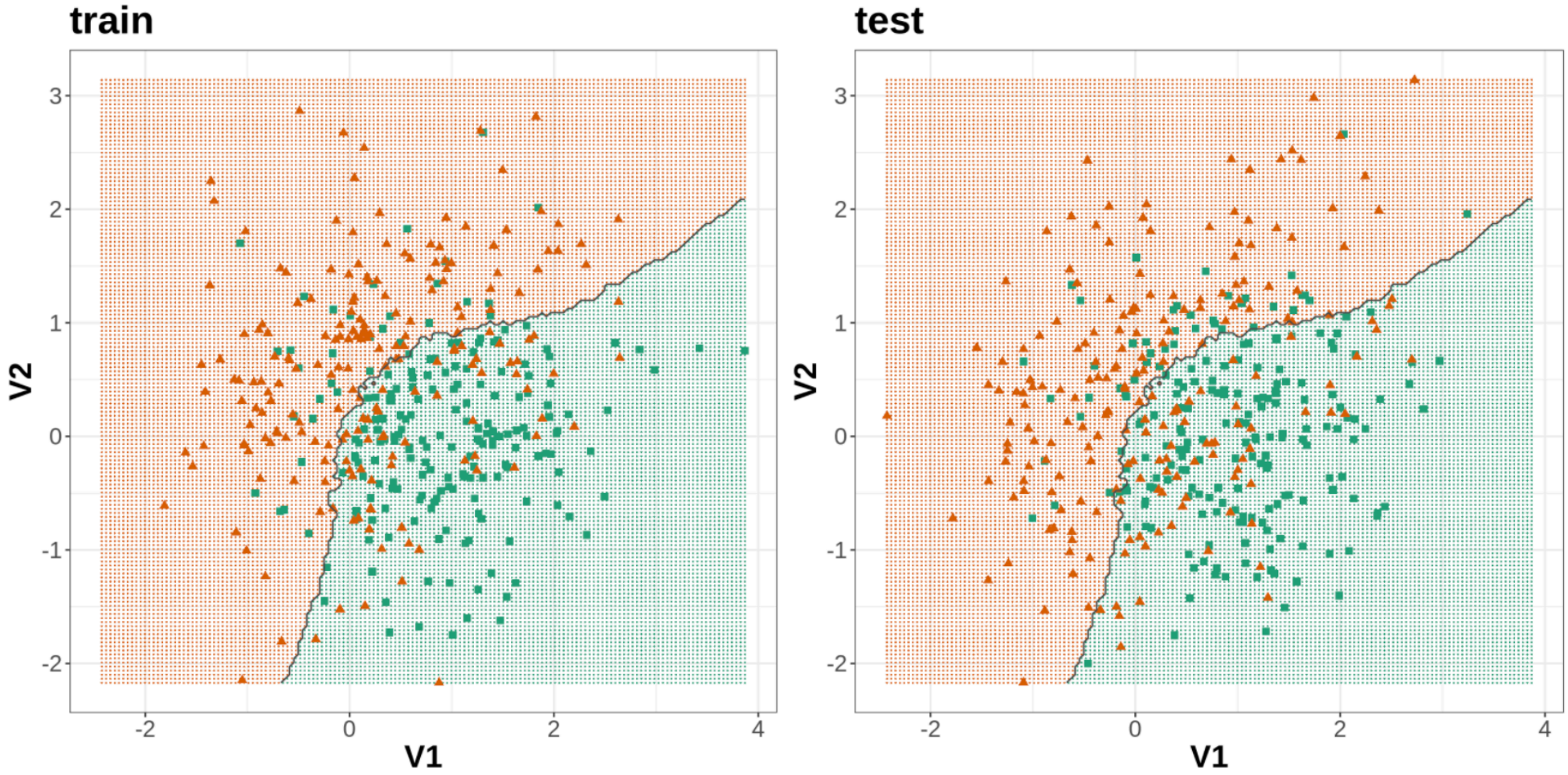
Cohen's Kappa:

$$Kappa = \frac{O - E}{1 - E} \tag{5.1}$$

where *O* is the observed accuracy and *E* is the expected accuracy based on the marginal totals of the confusion matrix. Cohen's Kappa takes values between -1 and 1; a value of zero indicates no agreement between the observed and predicted classes, while a value of one shows perfect concordance of the model prediction and the observed classes. If the prediction is in the opposite direction of the truth, a negative value will be obtained, but large negative values are rare in practice (Kuhn and Johnson 2013).



K nearest neighbours



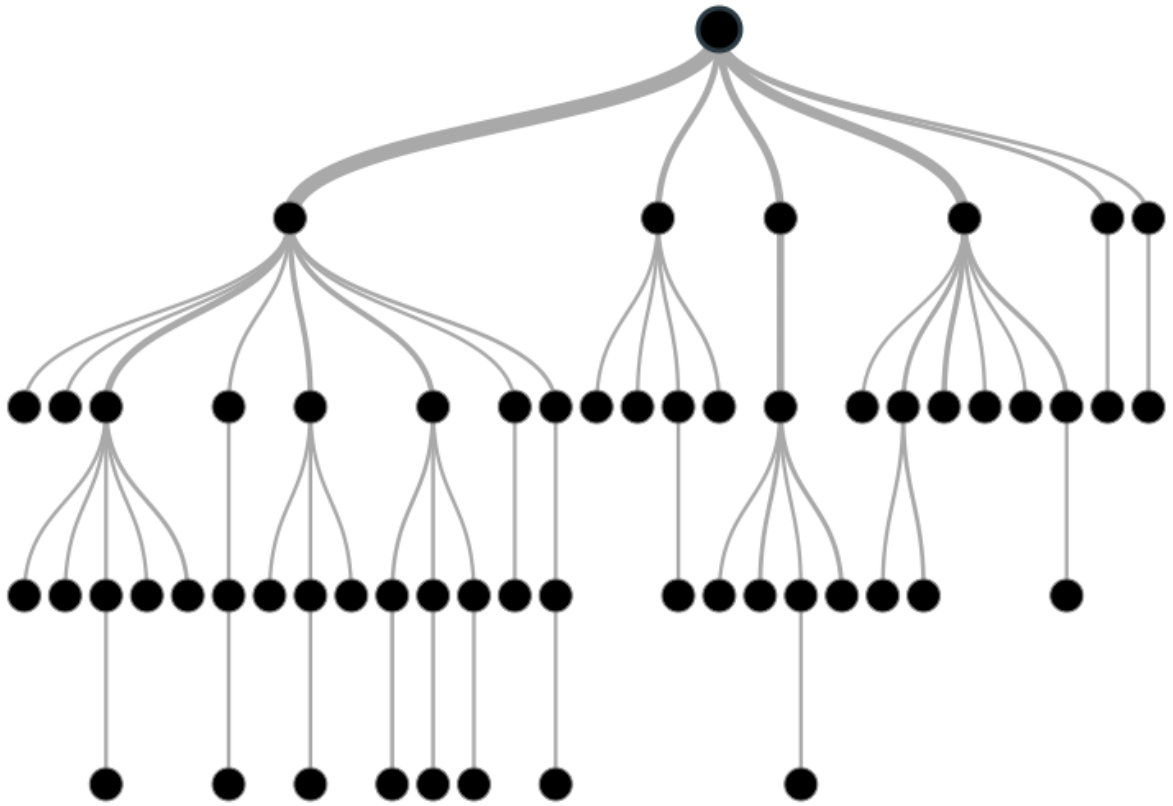
Binary classification of the simulated training and test sets with k=83.

Decision Trees

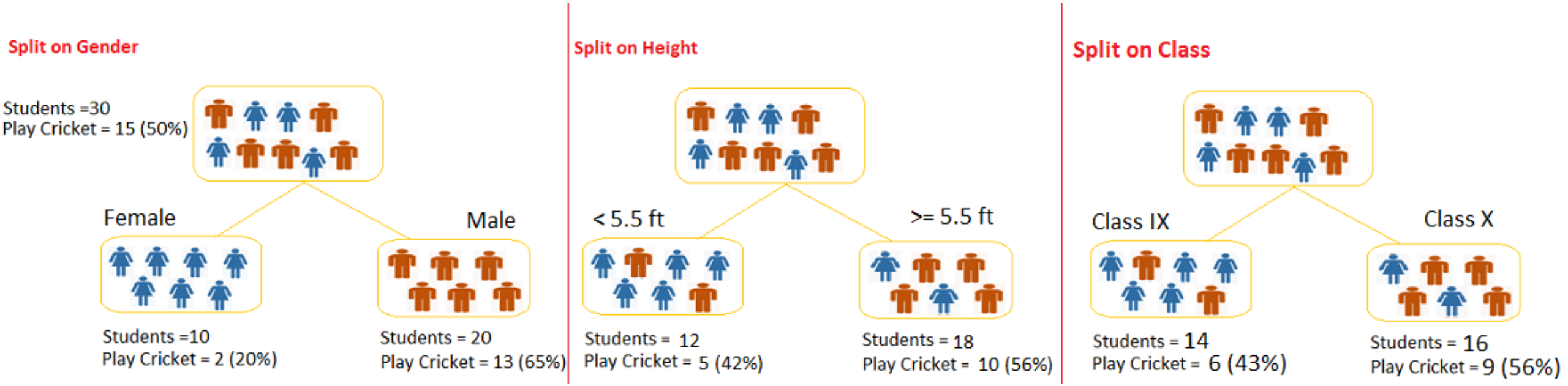
Decision tree or recursive partitioning is a supervised graph based algorithm to represent choices and the results of the choices in the form of a tree.

The nodes in the graph represent an event or choice and it is referred to as a leaf and the set of decisions made at the node is referred to as branches.

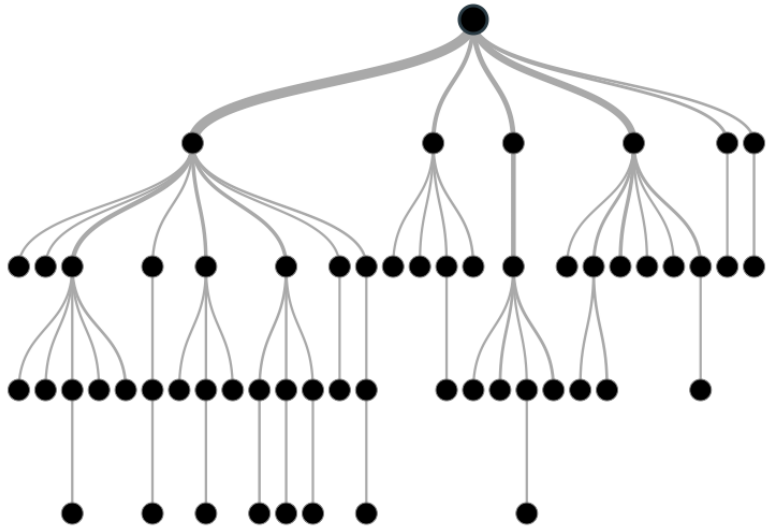
Decision trees map non-linear relationships and the hierarchical leaves and branches makes a Tree.



Decision Trees



Decision Trees



Terminology related to decision trees

Root node: the entire population that can get further divided into homogenous sets

Splitting: process of diving a node into two or more sub-nodes

Decision node: When a sub-node splits into further sub-nodes

Leaf or terminal node: when a node does not split further it is called a terminal node.

Pruning: A loose stopping criteria is used to construct the tree and then the tree is cut back by removing branches that do not contribute to the generalisation accuracy.

Branch: a sub-section of an entire tree

Decision Trees. Splitting decision

Gini Index

If we select two items from a population at random then they must be of same class and the probability for this is 1 if population is pure.

a. It works with categorical target variable “Success” or “Failure”.
b. It performs only Binary splits
c. Higher the value of Gini higher the homogeneity.
d. CART (Classification and Regression Tree) uses Gini method to create binary splits.

Step 1: Calculate Gini for sub-nodes, using formula sum of square of probability for success and failure

$$p^2 + q^2$$

. Step 2: Calculate Gini for split using weighted Gini score of each node of that split.

Decision Trees. Splitting decision

Chi-Square

It is an algorithm to find out the statistical significance between the differences between sub-nodes and parent node. We measure it by sum of squares of standardized differences between observed and expected frequencies of target variable.

a. It works with categorical target variable “Success” or “Failure”.
b. It can perform two or more splits.
c. Higher the value of Chi-Square higher the statistical significance of differences between sub-node and Parent node.
d. Chi-Square of each node is calculated using formula, Chi-square =

$$\sum(Actual - Expected)^2 / Expected$$

Steps to Calculate Chi-square for a split:

1. Calculate Chi-square for individual node by calculating the deviation for Success and Failure both
2. Calculated Chi-square of Split using Sum of all Chi-square of success and Failure of each node of the split

Decision Trees. Splitting decision

Information Gain

The more homogenous something is the less information is needed to describe it and hence it has gained information. Information theory has a measure to define this degree of disorganization in a system and it is known as Entropy. If a sample is completely homogeneous, then the entropy is zero and if it is equally divided (50% – 50%), it has entropy of one.

Entropy can be calculated using formula

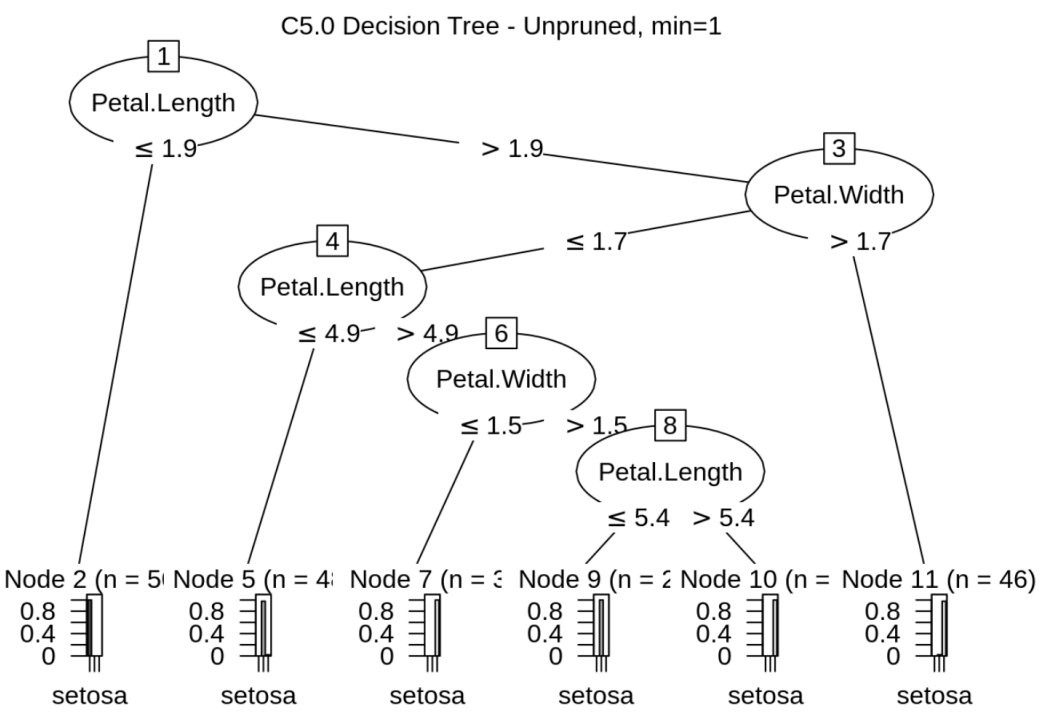
$$Entropy = -plog_2p - qlog_2q$$

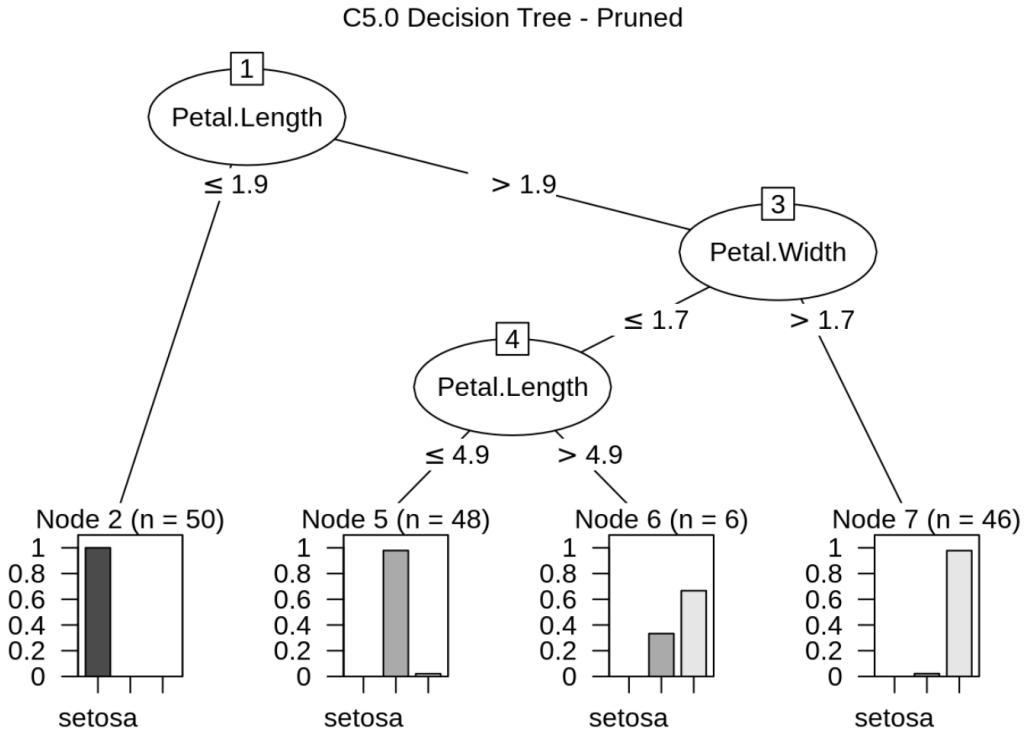
Where p and q are probablity of success and failure

Reduction in Variance

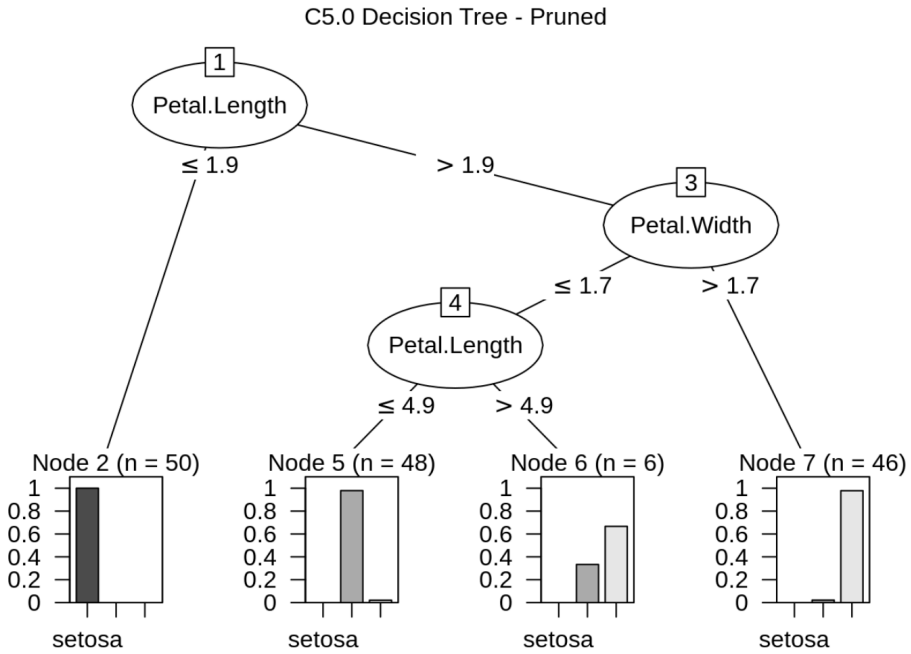
Reduction in variance is an algorithm used for continuous target variables (regression problems). This algorithm uses the standard formula of variance to choose the best split. The split with lower variance is selected as the criteria to split the population:

Decision Trees





Decision Trees



Advantages of decision tree

[a] Simple to understand and use

[b] Algorithms are robust to noisy data

[c] Useful in data exploration

[d] decision tree is 'non parametric' in nature

i.e. does not have any assumptions about the distribution of the variables

Disadvantages of decision tree

[a] Overfitting is the common disadvantage of decision trees.

It is taken care of partially by constraining the model parameter and by pruning.

[b] It is not ideal for continuous variables as in it looses information

Some parameters used to defining a tree and constrain overfitting

Minimum sample for a node split; Minimum sample for a terminal node

Maximum depth of a tree

Maximum number of terminal nodes; Maximum features considered for split

Decision Trees. Random Forests

What is a Random Forest?

It is a kind of ensemble learning method that combines a set of weak models to form a powerful model.
In the process it reduces dimensionality, removes outliers, treats missing values, and more importantly it is both a regression and classification machine learning approach.

How does it work?

In Random Forest, multiple trees are grown as opposed to a single tree in a decision tree model. Assume number of cases in the training set is N. Then, sample of these N cases is taken at random but with replacement. This sample will be the training set for growing the tree. Each tree is grown to the largest extent possible and without pruning.

To classify a new object based on attributes, each tree gives a classification i.e. **“votes” for that class**. The forest chooses the classification having the most votes (over all the trees in the forest) and in case of regression, it takes the average of outputs by different trees.

Random Forests

Key differences between decision trees and random forest

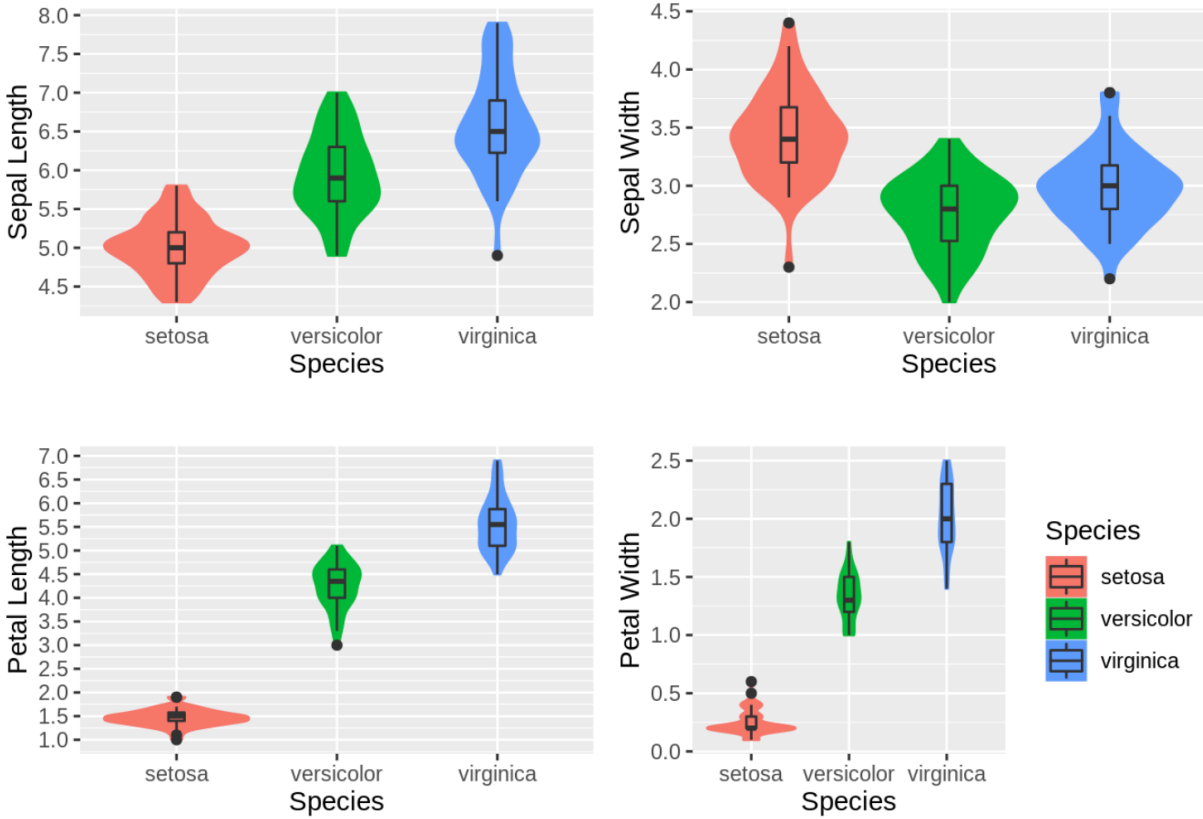
Decision trees proceed by searching for a split on every variable in every node; random forest searches for a split only on one variable in a node - the variable that has the largest association with the target among all other explanatory variables but only on a subset of randomly selected explanatory variables that is tested for that node.

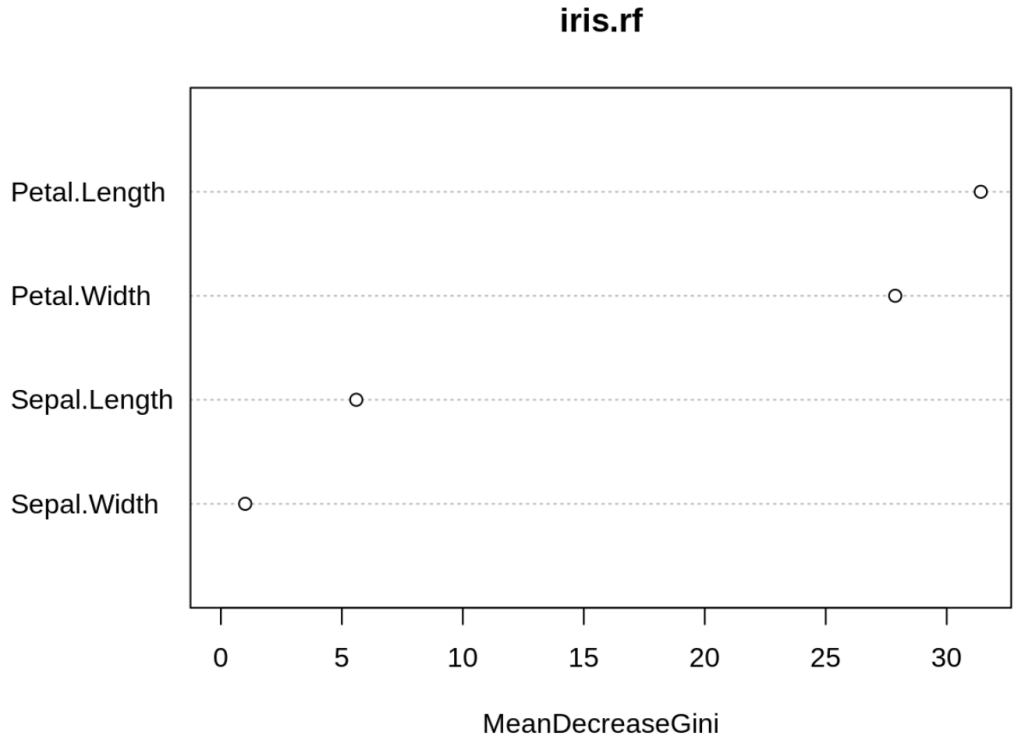
Therefore, eligible variable set will be different from node to node but the important ones will eventually be "voted in" based on their success in predicting the target variable.

This random selection of explanatory variables at each node and which are different at each tree is known as bagging. For each tree the ratio between bagging and out of bagging is 60/40.

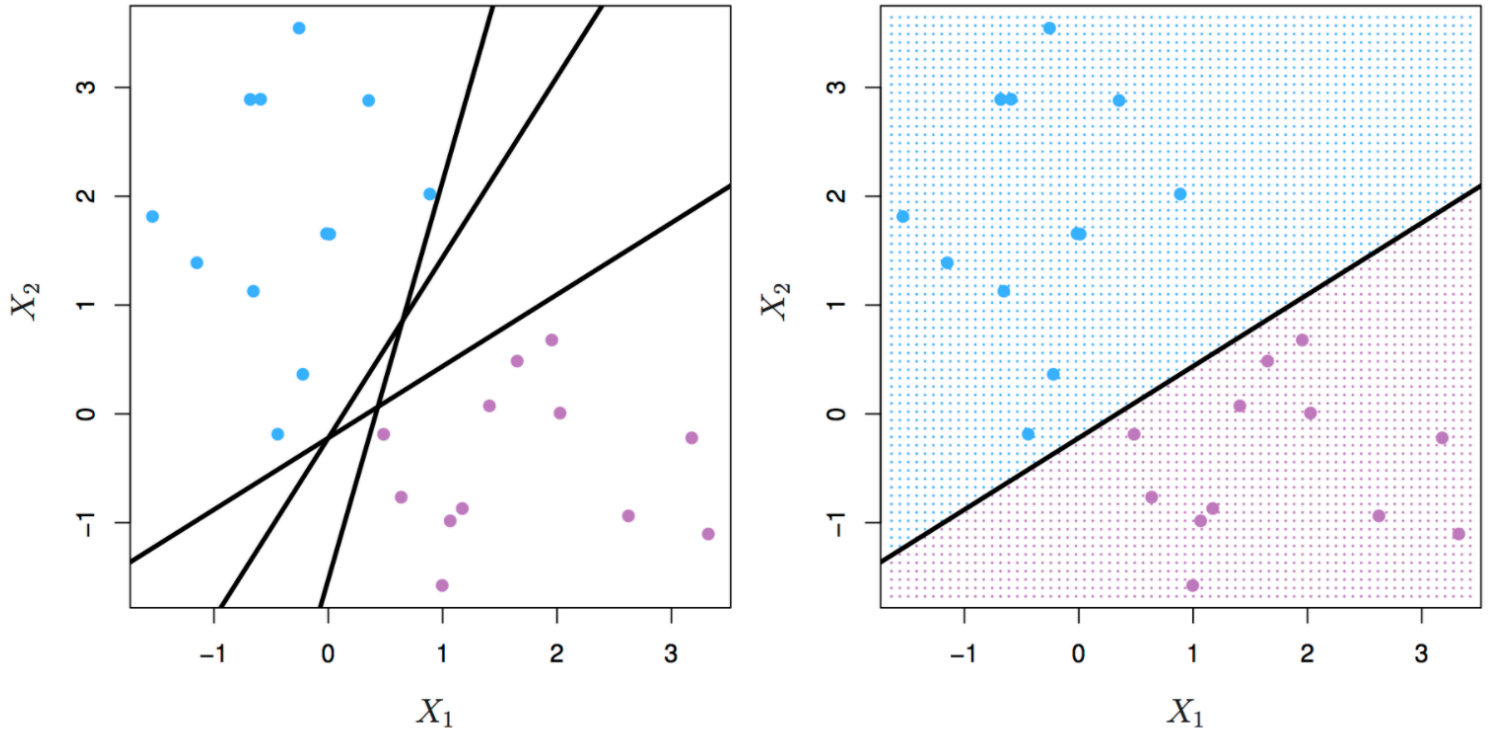
Trees themselves are not interpreted but they are used to collectively rank the importance of each variable.

Random Forests. Variable Importance





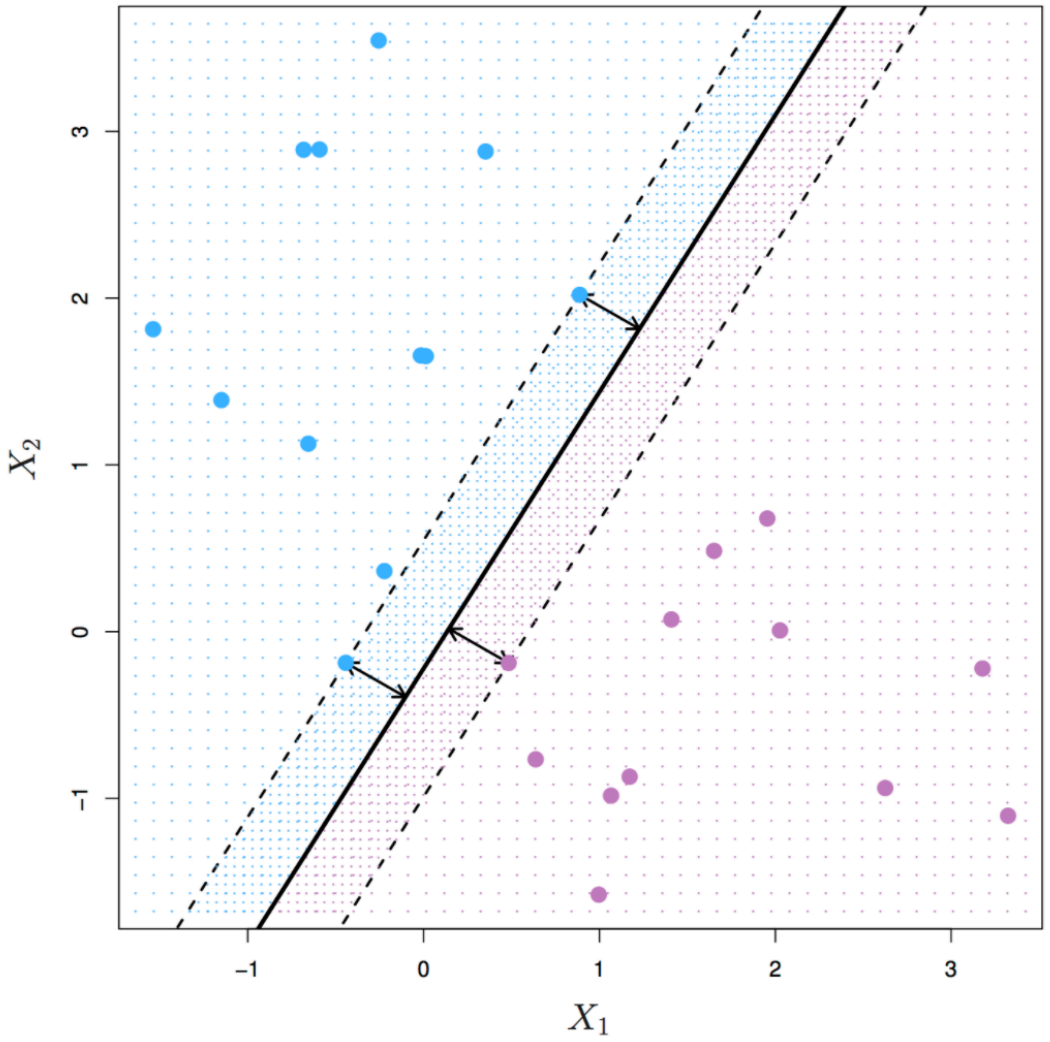
Support Vector Machines (SVMs)

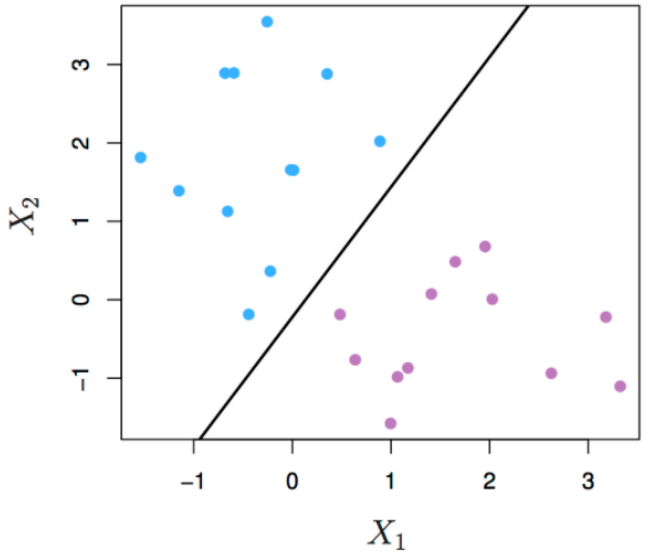


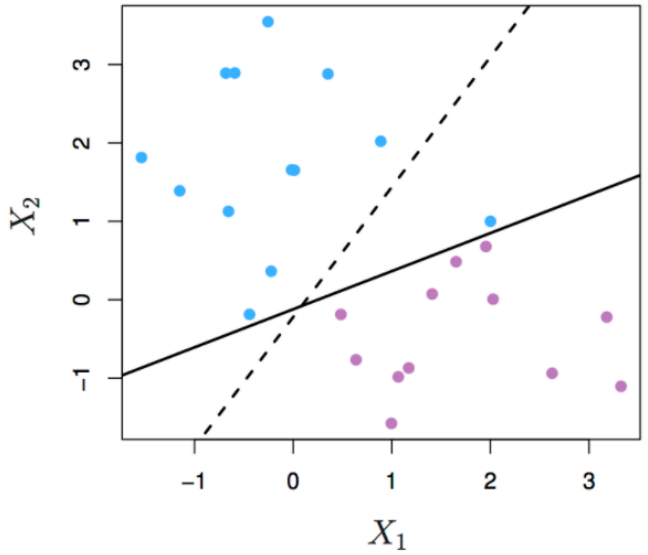
Left: two classes of observations (blue, purple) and three separating hyperplanes. Right: separating hyperplane shown as black line and grid indicates decision rule. Source: http://www-bcf.usc.edu/~gareth/ISL/

The **maximal margin hyperplane** is the separating hyperplane that is farthest from the training observations. The perpendicular distance from a given hyperplane to the nearest training observation is known as the margin. The maximal margin hyperplane is the separating hyperplane for which the margin is largest.

Support Vector Machines (SVMs)

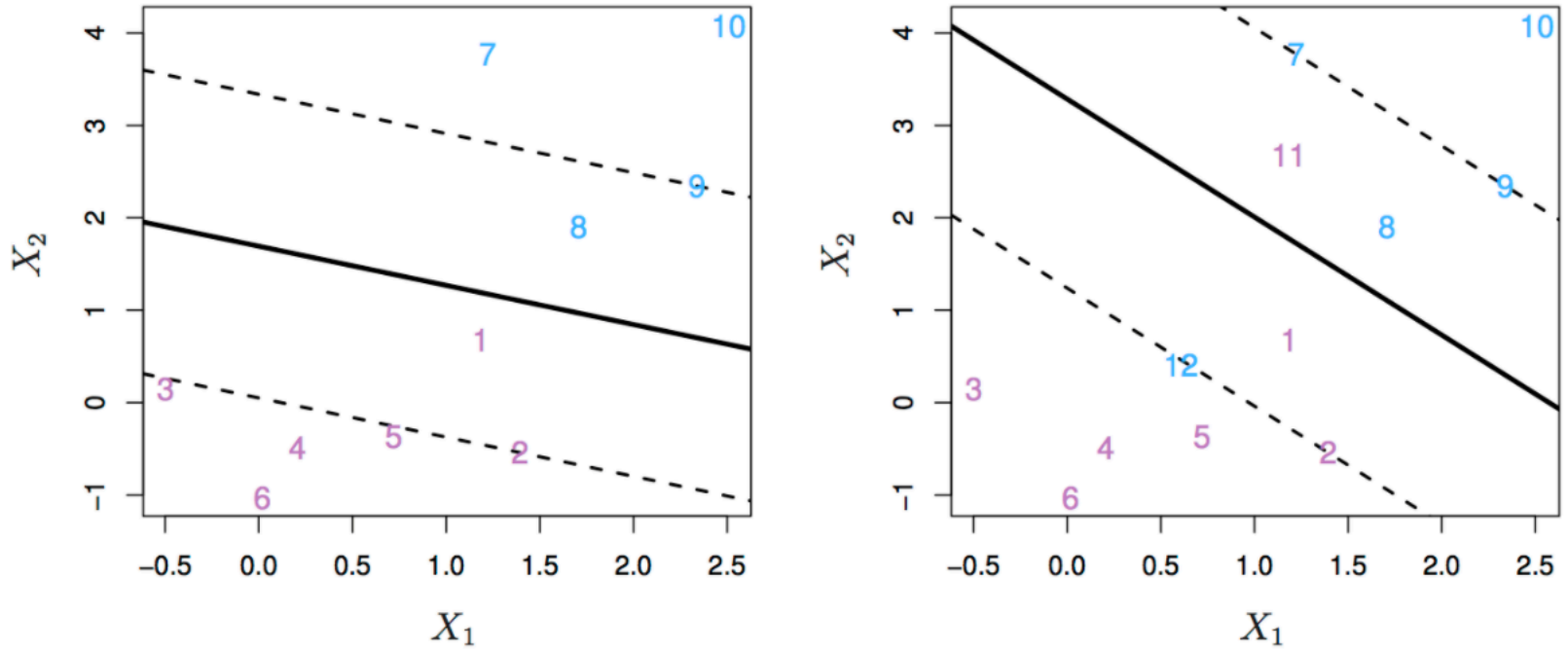






The three training observations that are equidistant from the maximal margin hyperplane and lie on the dashed lines underline the margin. These are the **support vectors**. If these points were moved slightly, the maximal margin hyperplane would also move, hence the term *support*. The maximal margin hyperplane is set by the **support vectors** alone; it is not influenced by any other observations.

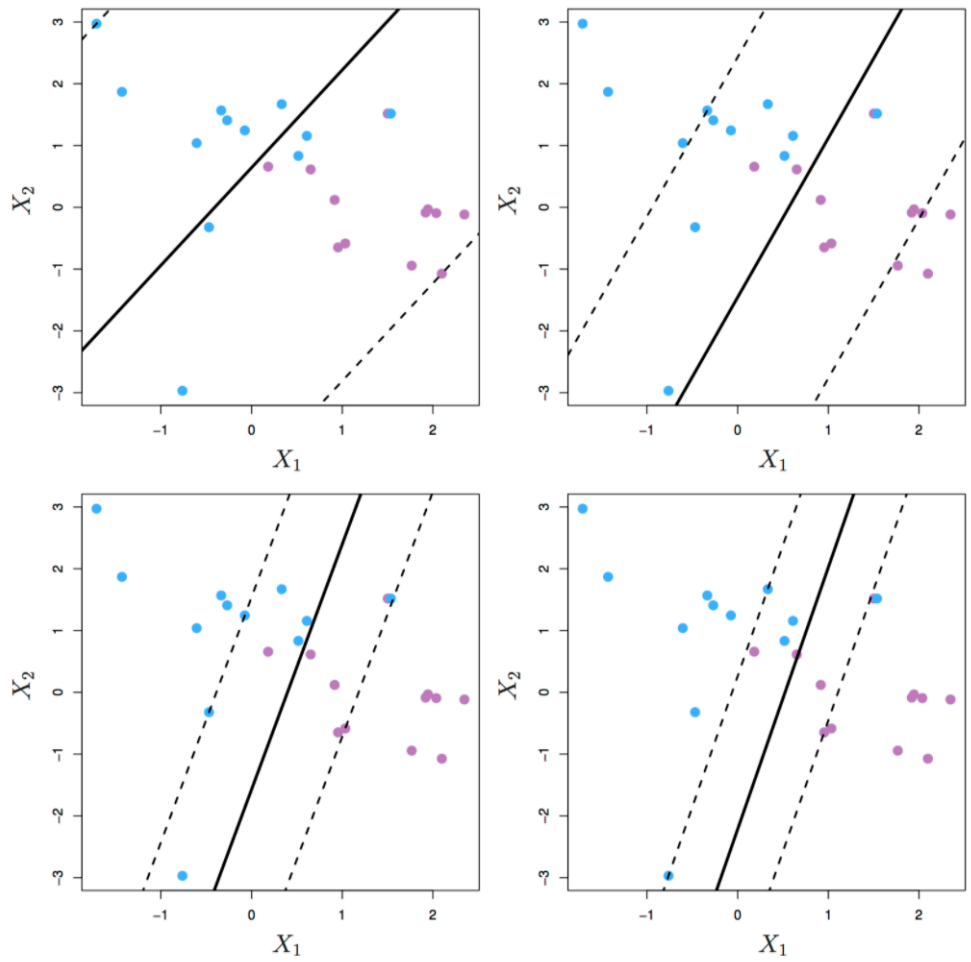
Support Vector Machines (SVMs)



Left: observations on the wrong side of the margin. Right: observations on the wrong side of the margin and observations on the wrong side of the hyperplane. Source: http://www-bcf.usc.edu/~gareth/ISL/

we might tolerate some misclassifications if the prediction of the remaining observations is more reliable. The **support vector classifier** does this by allowing some observations to be on the wrong side of the margin or even on the wrong side of the hyperplane. Observations on the wrong side of the hyperplane are misclassifications.

Support Vector Machines (SVMs)



Margin of a support vector classifier changing with tuning parameter C. Largest value of C was used in the top left panel, and smaller values in the top right, bottom left and bottom right panels. Source: http://www-bcf.usc.edu/~gareth/ISL/

The support vector classifier has a **tuning parameter *C***, that determines the number and severity of the violations to the margin.

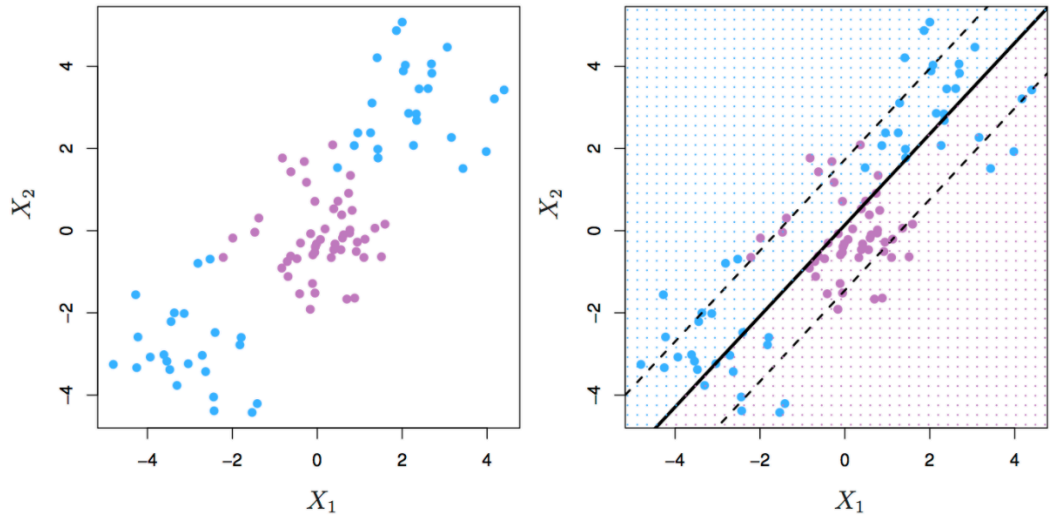
If *C* = 0, then no violations to the margin will be tolerated, which is equivalent to the maximal margin classifier.

As *C* increases, the classifier becomes more tolerant of violations to the margin, and so the margin widens.

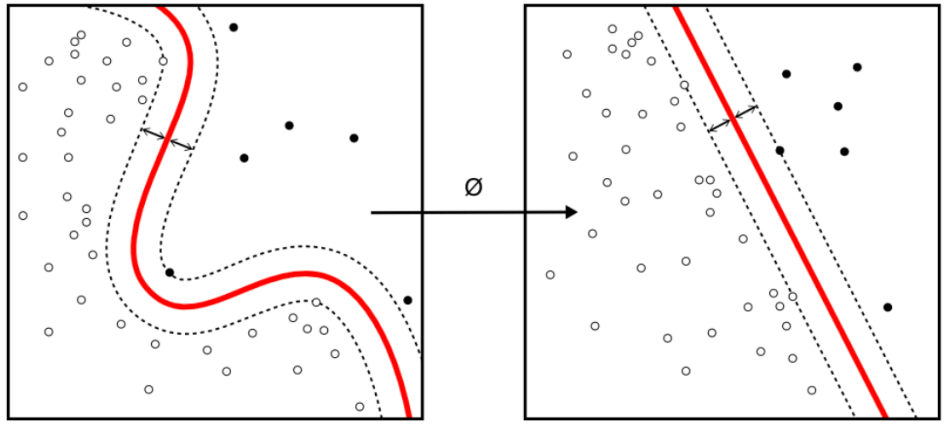
The optimal value of *C* is chosen through **cross-validation**.

C is described as a tuning parameter, because it controls the bias-variance trade-off:
[a] **small *C*** > narrow margins that are rarely violated; the model will have low bias, but high variance.
[b] as *C* increases the margins widen allowing more violations; the bias of the model will increase, but its variance will decrease.

Support Vector Machines (SVMs)

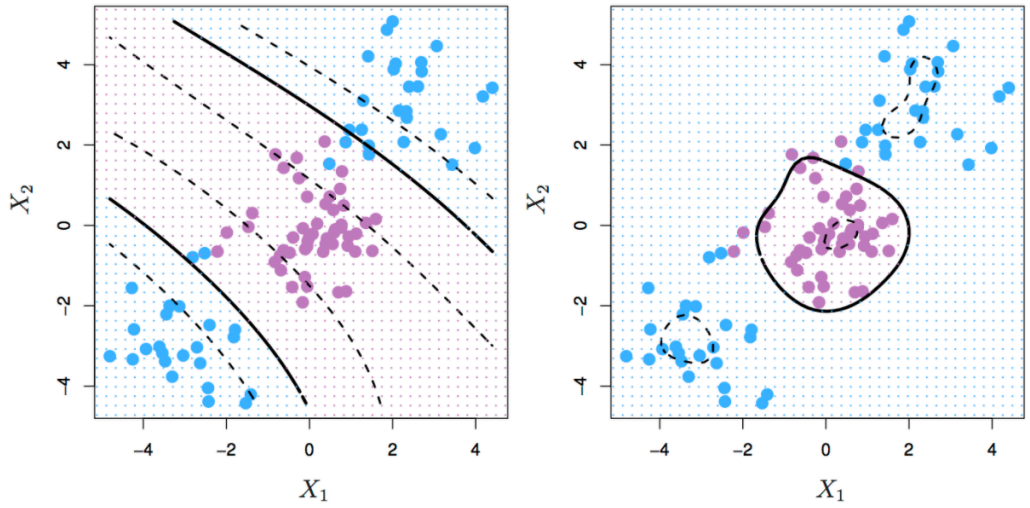


Two classes of observations with a non-linear boundary between them.



Kernel machine. By Alisneaky - Own work, CC0, https://commons.wikimedia.org/w/index.php?curid=14941564

The second parameter of an SVM is the kernel.
For the previous examples we illustrated a linear kernel.
In these examples we are showing an RBF kernel.



Left: SVM with polynomial kernel of degree 3. Right: SVM with radial kernel. Source: http://www-bcf.usc.edu/~gareth/ISL/