CLASSIFIER DECISION BOUNDARY

A view of the decision boundaries of machine learning algorithms

Adriano Henrique Cantão

Master's student in computing applied to complex systems at FFCLRP/USP

cantao@usp.br, adriano.cantao@gmail.com

This presentation and the R script used to generate the data are available at:

https://github.com/ahcantao/classifier_decision_boundary

Datasets and Algorithms

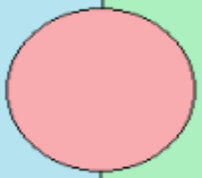
These algorithms are applied to a collection of artificial datasets (machine learning benchmark problems), available on the package 'mlbench':

1. Circle In A Square Problem
2. 2-Dimensional Gaussian Problem
3. Two Spirals Benchmark Problem
4. Continuous XOR Benchmark Problem
5. Corners of d-dimensional Simplex
6. The Smiley
7. Corners of Hypercube
8. Ringnorm Benchmark Problem
9. Twonorm Benchmark Problem
10. Threenorm Benchmark Problem
11. Shapes in 2d
12. Cassini: A 2 Dimensional Problem

1. K Nearest Neighbors (KNN)
   - k = {1, 3, 5, 10, 30}
2. Support Vector Machine (SVM)
   - kernels = {linear, polynomial, radial, sigmoid}
3. Naive Bayes
4. Tree-based:
   - C.50, CART and Random Forest
5. Artificial Neural Network (ANN)
   - neurons = {1, 10, 100, 500, 1000}
6. Artificial Neural Network (ANN)
   - max_iterations = {10, 100, 500, 1000, 10000}

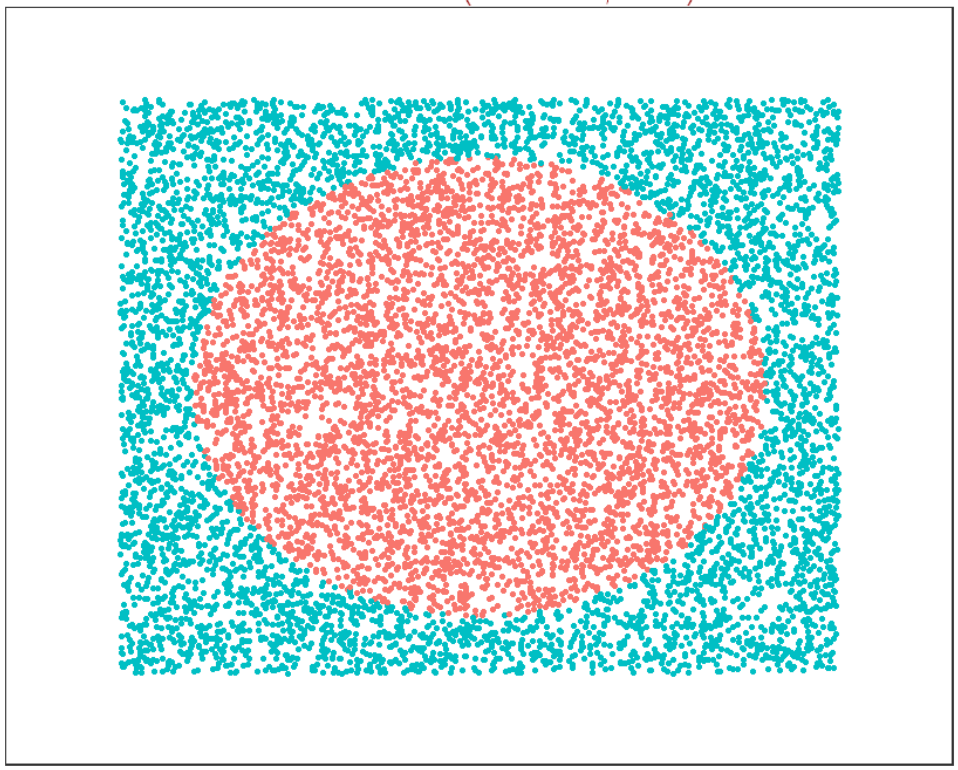
This presentation and the R script used to generate the data are available at:
https://github.com/ahcantao/classifier_decision_boundary

Circle



Dataset

mlbench.circle (n = 10000, d = 2)



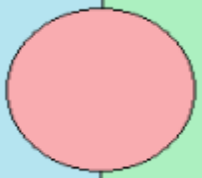
This dataset image contains 10,000 points, so we can se its true form. The next ones will contain a low number in order to have a better view of the decision boundary of each algorithm.

Circle In A Square Problem

The inputs of the circle problem are uniformly distributed on the *d*-dimensional cube with corners + - 1. This is a 2-class problem: The first class is a *d*-dimensional ball in the middle of the cube, the remainder forms the second class. The size of the ball is chosen such that both classes have equal prior probability 0.5

Usage:
mlbench.circle(n, d)

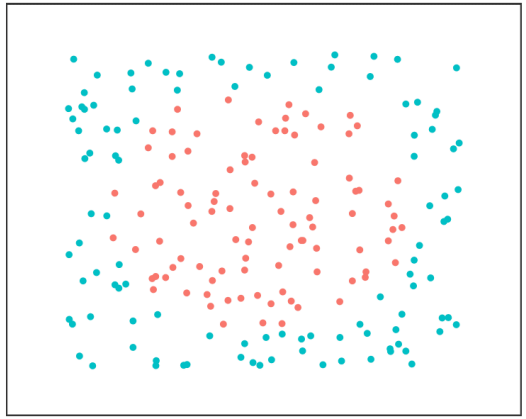
Circle



KNN

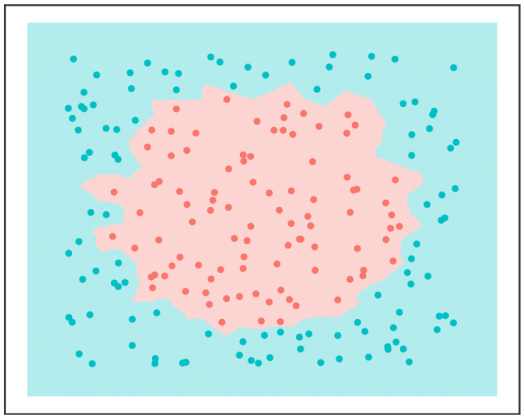
Dataset

mlbench.circle (n=200, d=2)



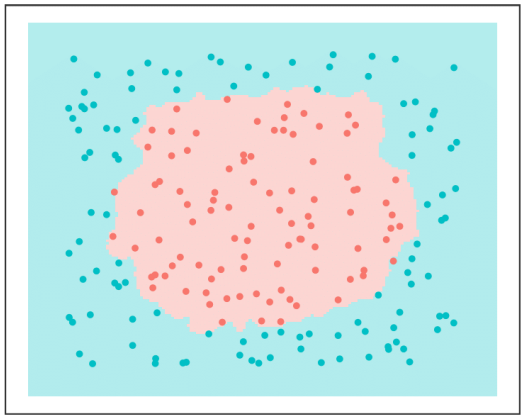
KNN1

training accuracy: 100 % – test accuracy: 94.5 %



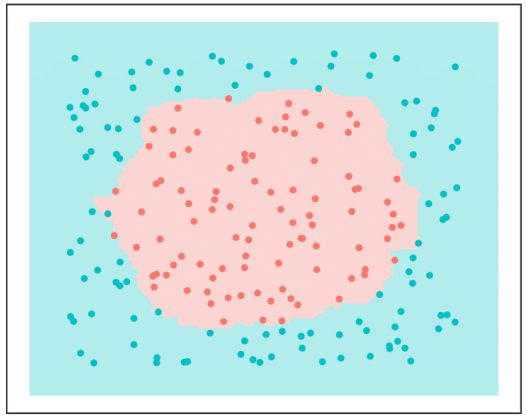
KNN3

training accuracy: 99 % – test accuracy: 95.5 %



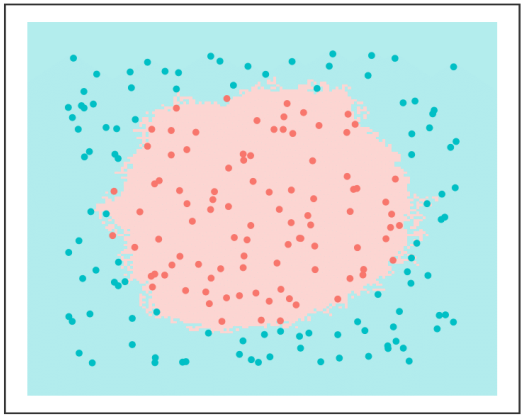
KNN5

training accuracy: 96.5 % – test accuracy: 95.5 %



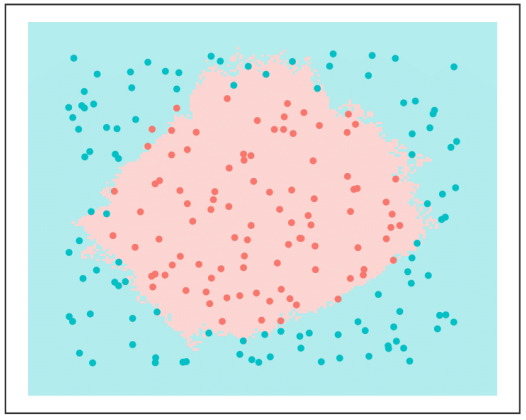
KNN10

training accuracy: 97.5 % – test accuracy: 94 %

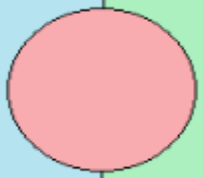


KNN30

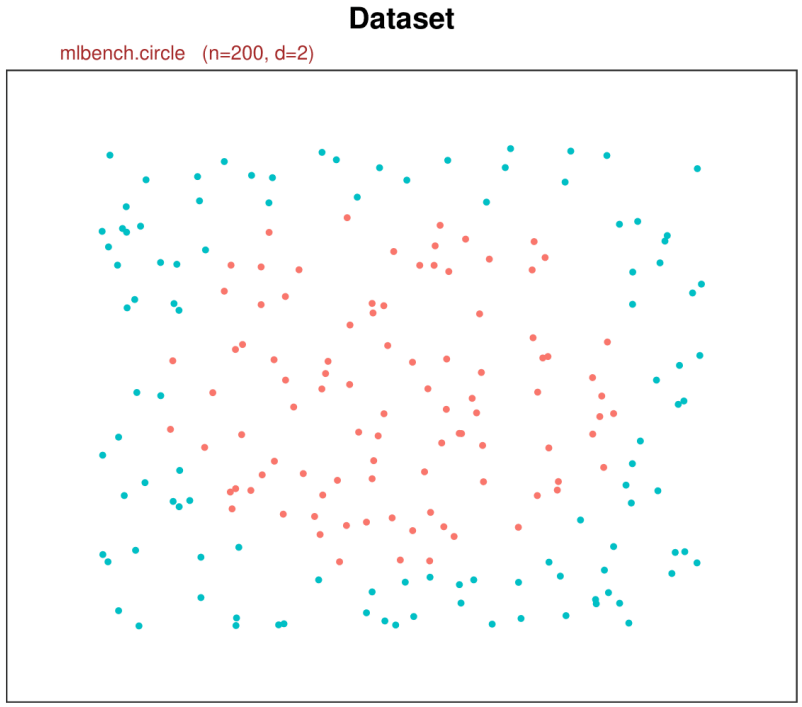
training accuracy: 93.5 % – test accuracy: 93.5 %

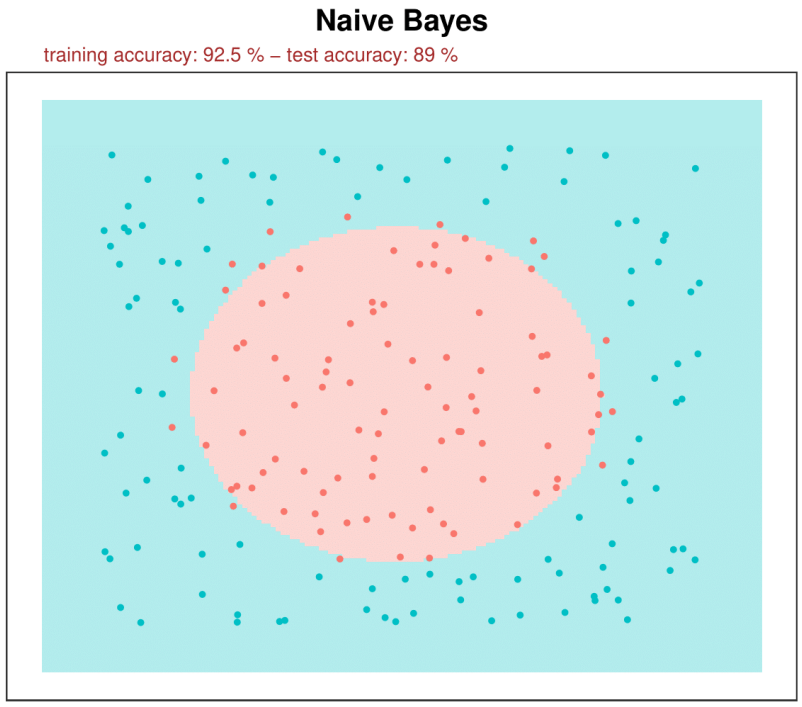


Circle

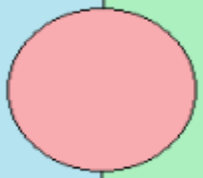


Naïve Bayes





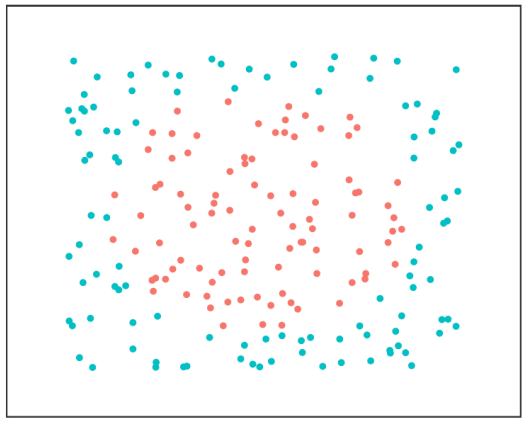
Circle



SVM

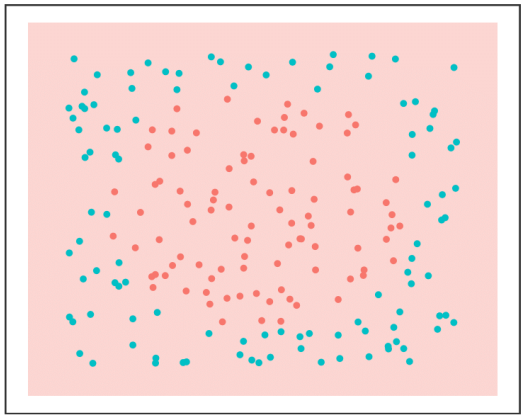
Dataset

mlbench.circle (n=200, d=2)



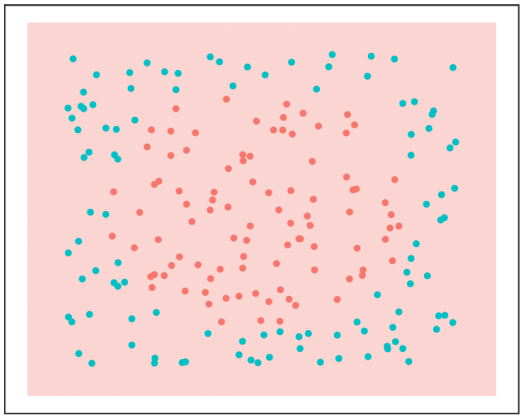
SVMlinear

training accuracy: 52.5 % – test accuracy: 49 %



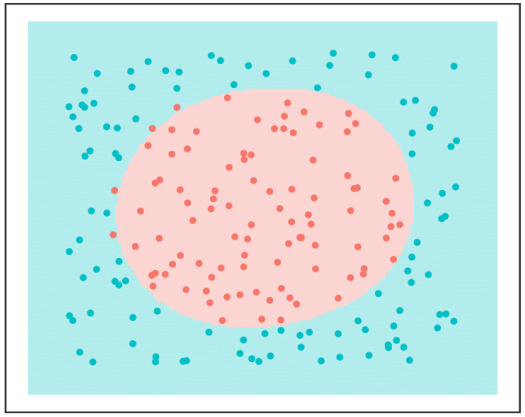
SVMpolynomial

training accuracy: 52.5 % – test accuracy: 49 %



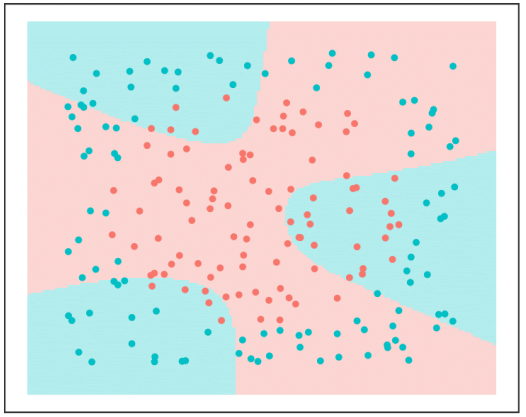
SVMradial

training accuracy: 98 % – test accuracy: 97 %

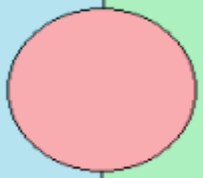


SVMsigmoid

training accuracy: 54 % – test accuracy: 54.5 %



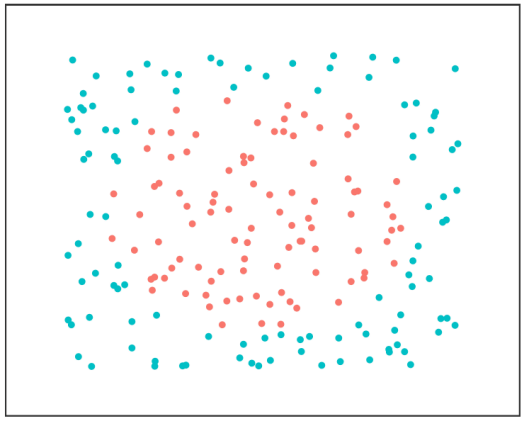
Circle



Trees

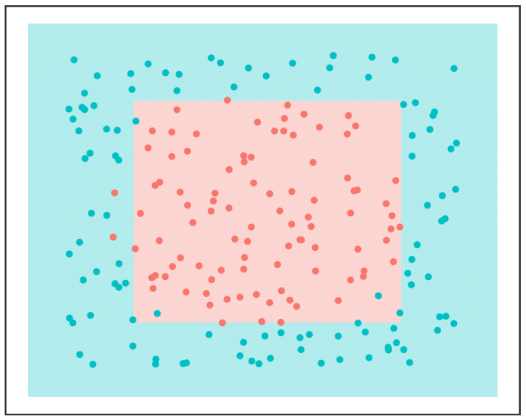
Dataset

mlbench.circle (n=200, d=2)



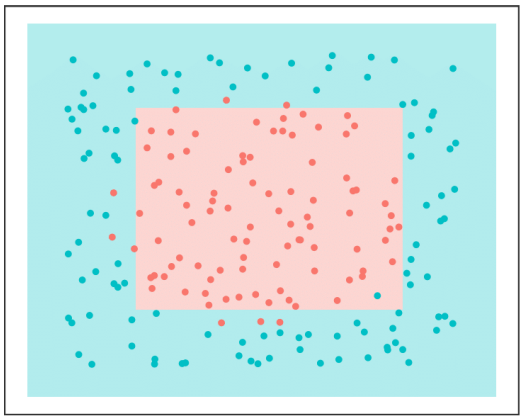
TreesC.50

training accuracy: 97 % – test accuracy: 92 %



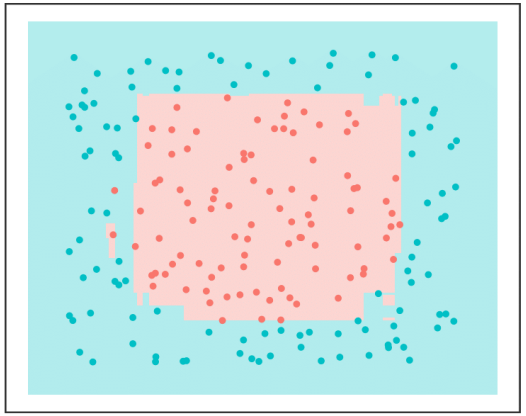
TreesCART

training accuracy: 95.5 % – test accuracy: 90 %



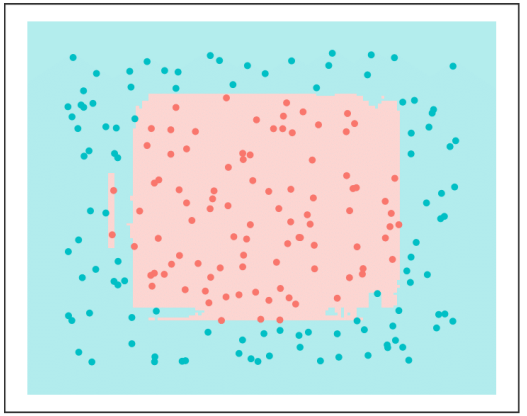
TreesRandom Forest 3 trees

training accuracy: 99 % – test accuracy: 91.5 %

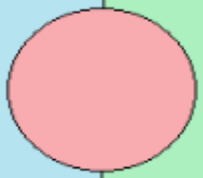


TreesRandom Forest 128 trees

training accuracy: 100 % – test accuracy: 92.5 %



Circle

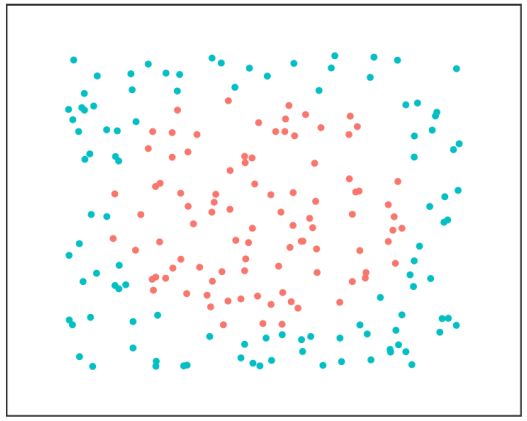


ANN

fixed iterations

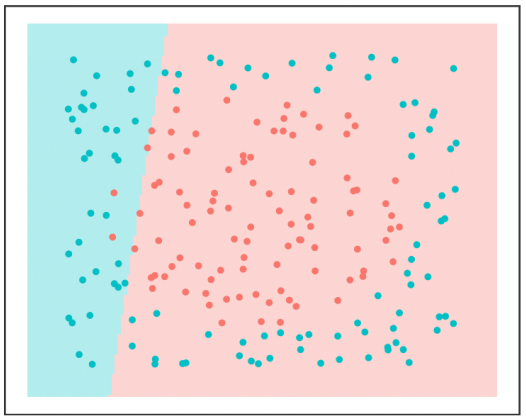
Dataset

mlbench.circle(n=200, d=2)



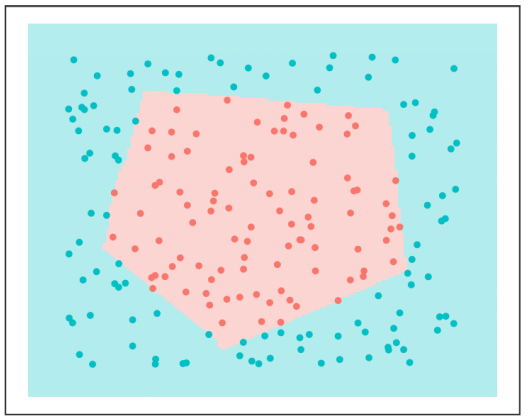
ANN; iters=100; neurons=1

training accuracy: 63 % – test accuracy: 65.5 %



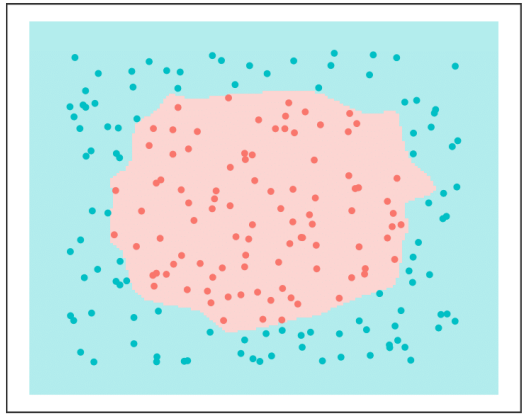
ANN; iters=100; neurons=10

training accuracy: 100 % – test accuracy: 96 %



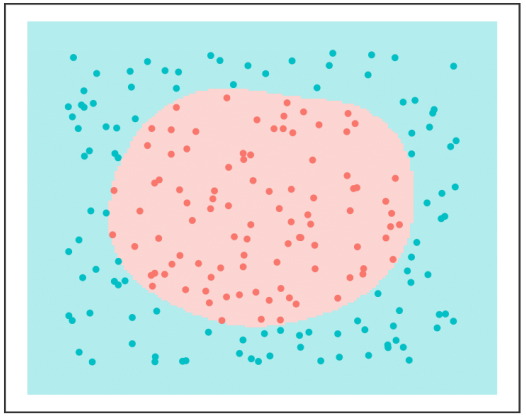
ANN; iters=100; neurons=100

training accuracy: 100 % – test accuracy: 96.5 %



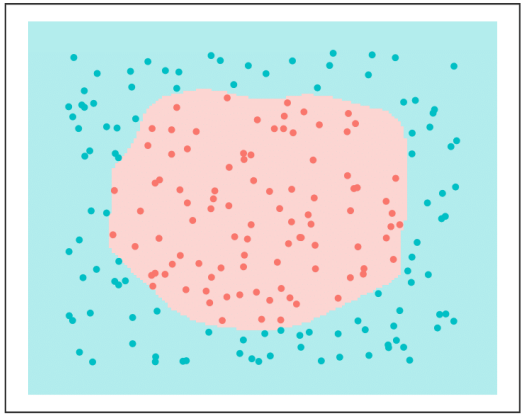
ANN; iters=100; neurons=500

training accuracy: 100 % – test accuracy: 97.5 %

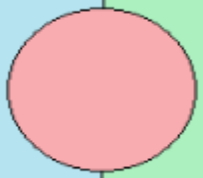


ANN; iters=100; neurons=1000

training accuracy: 100 % – test accuracy: 96.5 %



Circle

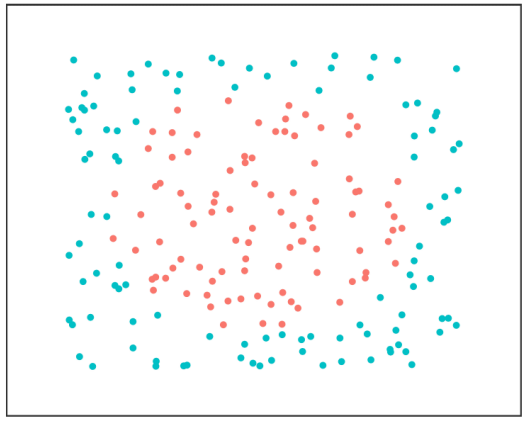


ANN

fixed neurons

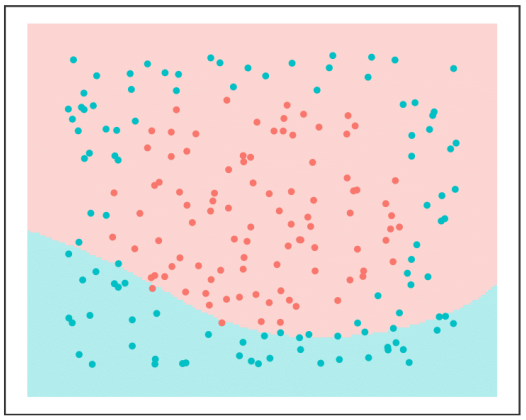
Dataset

mlbench.circle(n=200, d=2)



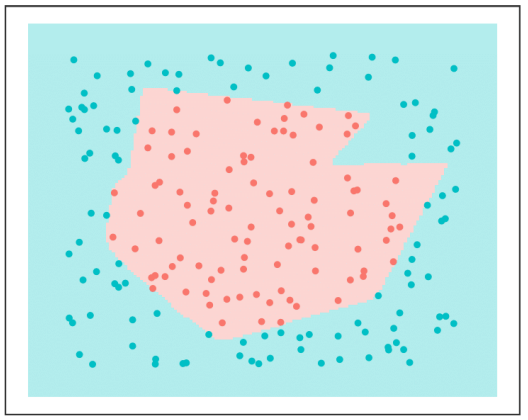
ANN; neurons=10; iters=10

training accuracy: 66 % – test accuracy: 61.5 %



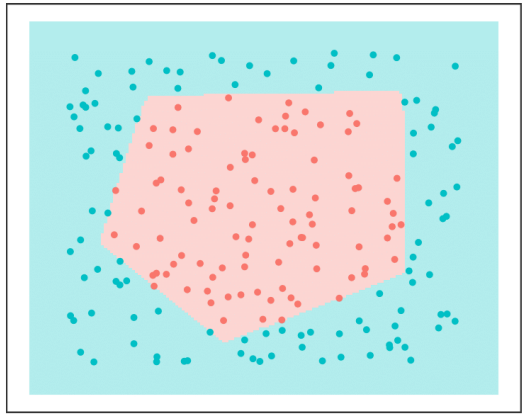
ANN; neurons=10; iters=100

training accuracy: 100 % – test accuracy: 97.5 %



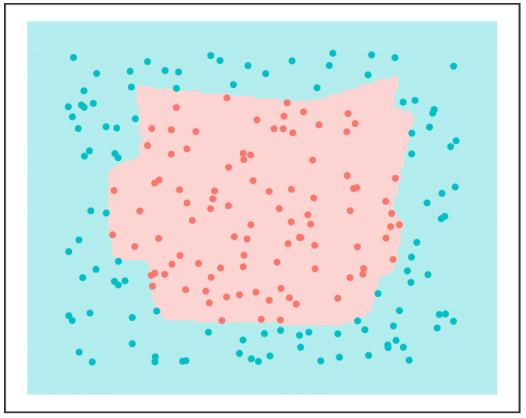
ANN; neurons=10; iters=500

training accuracy: 100 % – test accuracy: 95.5 %



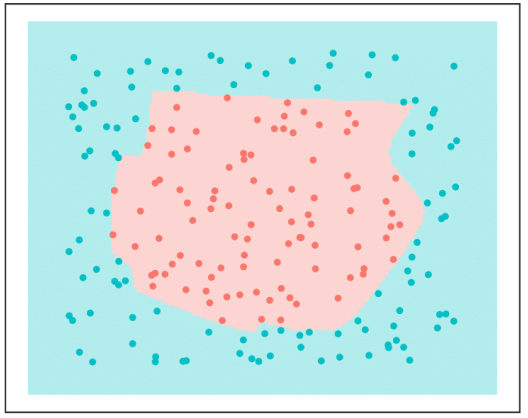
ANN; neurons=10; iters=1000

training accuracy: 100 % – test accuracy: 92.5 %

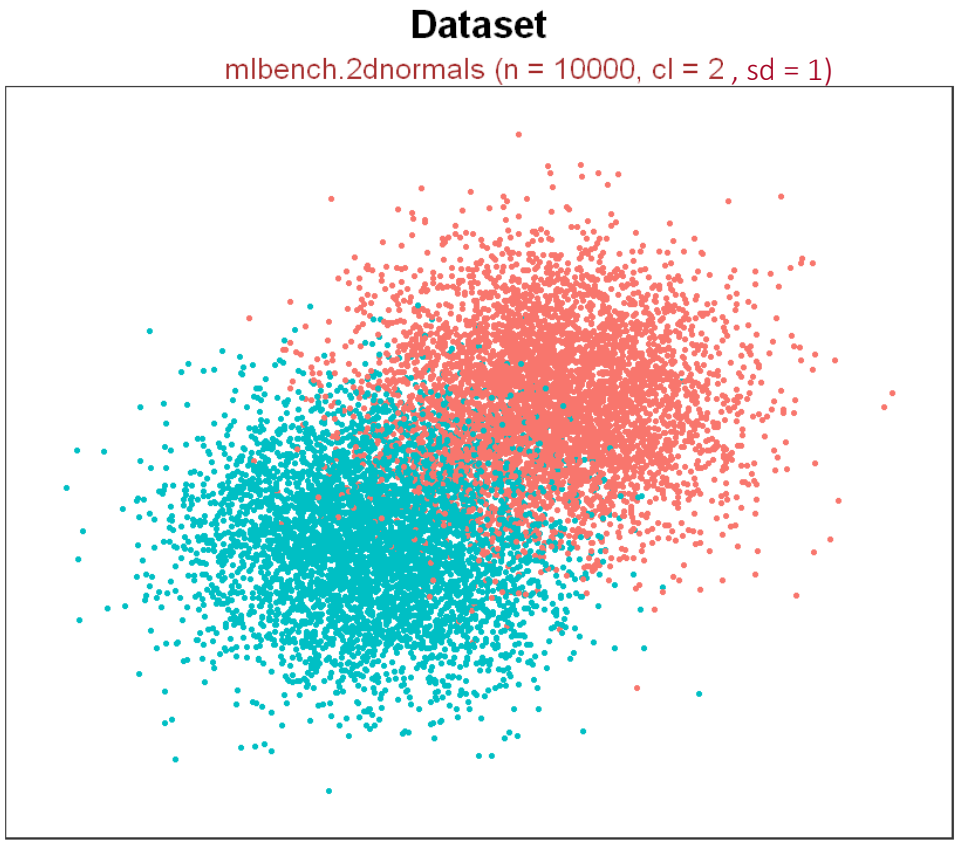


ANN; neurons=10; iters=10000

training accuracy: 100 % – test accuracy: 94 %



2D Normals



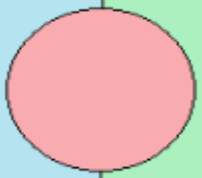
This dataset image contains 10,000 points, so we can se its true form. The next ones will contain a low number in order to have a better view of the decision boundary of each algorithm.

2-Dimensional Gaussian Problem

Each of the *cl* classes consists of a 2-dimensional Gaussian. The centers are equally spaced on a circle around the origin with radius *r*.

Usage:
mlbench.2dnormals(n, cl, r, sd)

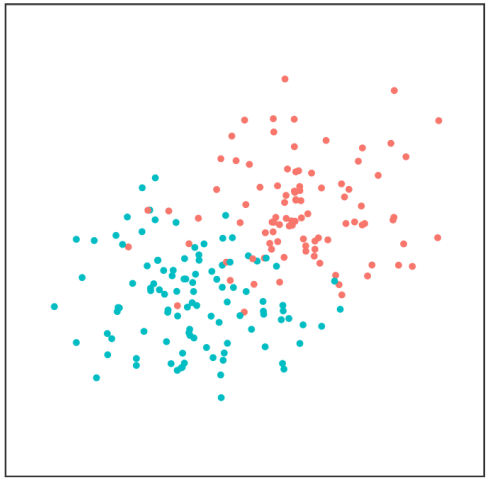
2D Normals



KNN

Dataset

mlbench.2dnormals(n=200, cl=2)

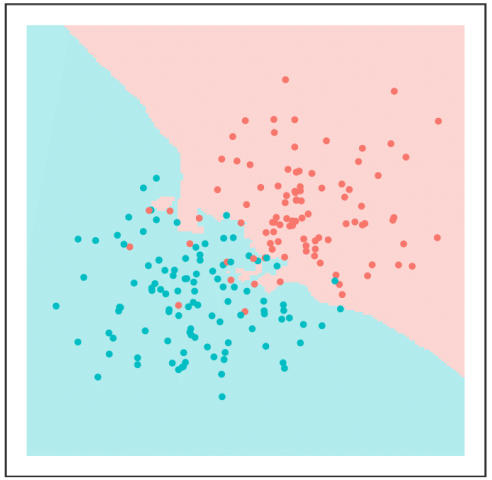


KNN1

training accuracy: 100 % – test accuracy: 88 %

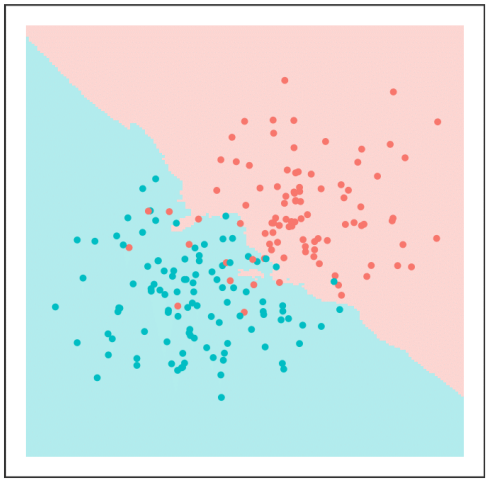
KNN3

training accuracy: 92 % – test accuracy: 91 %



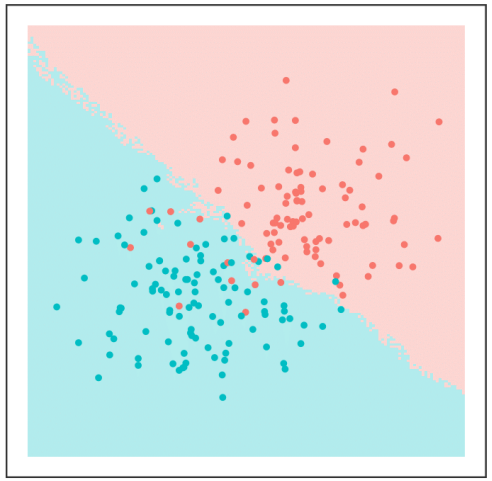
KNN5

training accuracy: 91 % – test accuracy: 91 %



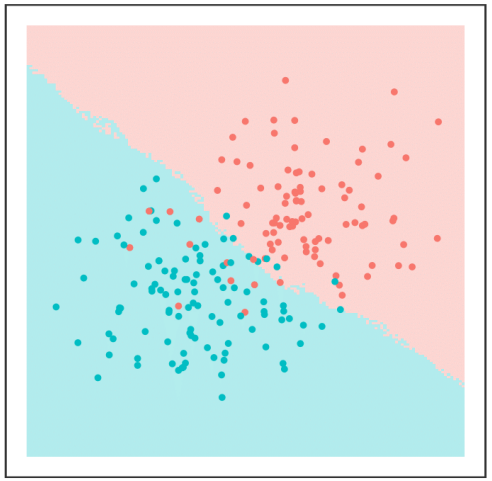
KNN10

training accuracy: 92 % – test accuracy: 89 %

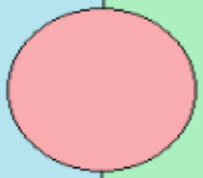


KNN30

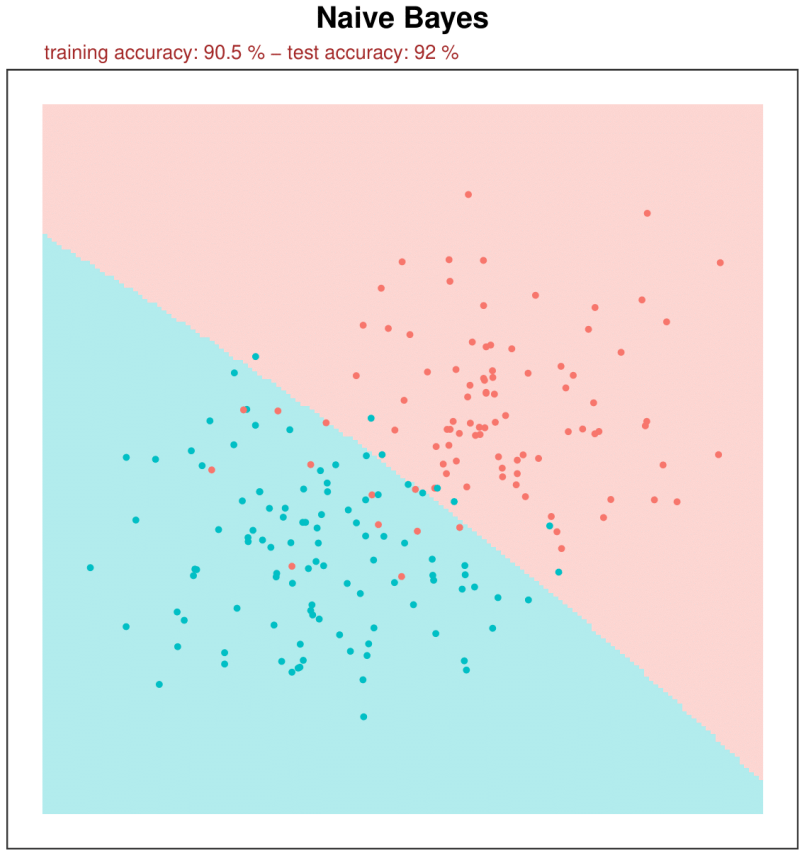
training accuracy: 91 % – test accuracy: 90 %



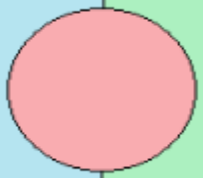
2D Normals



Naïve Bayes



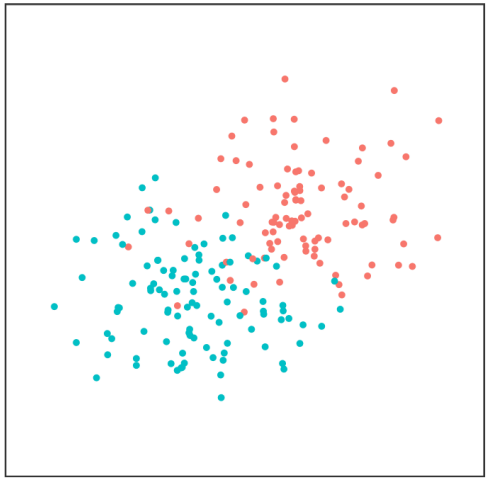
2D Normals



SVM

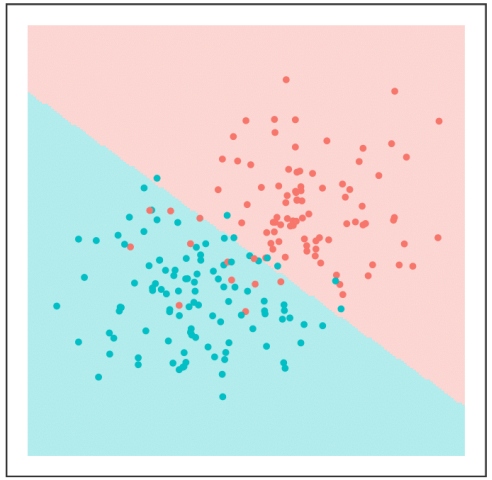
Dataset

mlbench.2dnormals(n=200, cl=2)



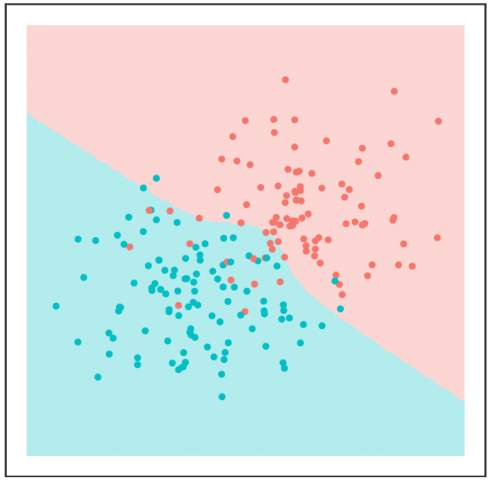
SVMlinear

training accuracy: 90.5 % – test accuracy: 92.5 %



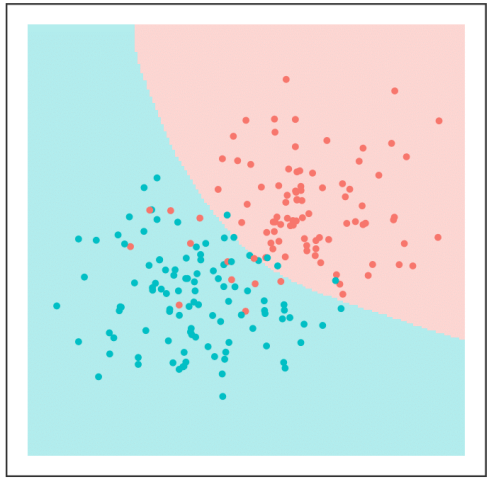
SVMpolynomial

training accuracy: 89.5 % – test accuracy: 90 %



SVMradial

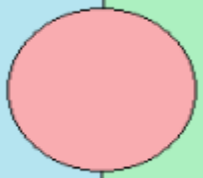
training accuracy: 92 % – test accuracy: 91 %



SVMsigmoid

training accuracy: 91.5 % – test accuracy: 90 %

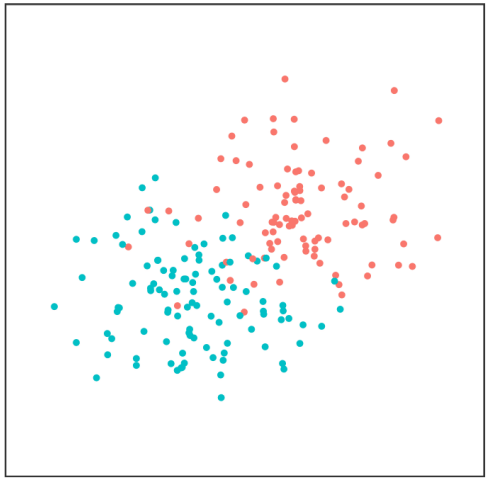
2D Normals



Trees

Dataset

mlbench.2dnormals(n=200, cl=2)



TreesC.50

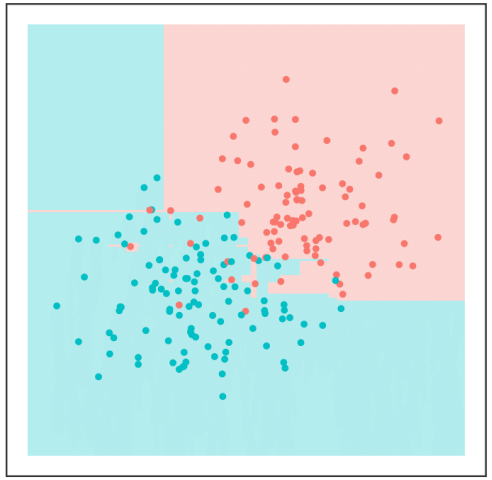
training accuracy: 91.5 % – test accuracy: 87.5 %

TreesCART

training accuracy: 91.5 % – test accuracy: 86.5 %

TreesRandom Forest 3 trees

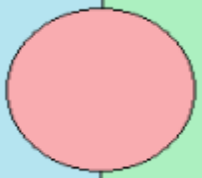
training accuracy: 98.5 % – test accuracy: 91.5 %



TreesRandom Forest 128 trees

training accuracy: 100 % – test accuracy: 89 %

2D Normals

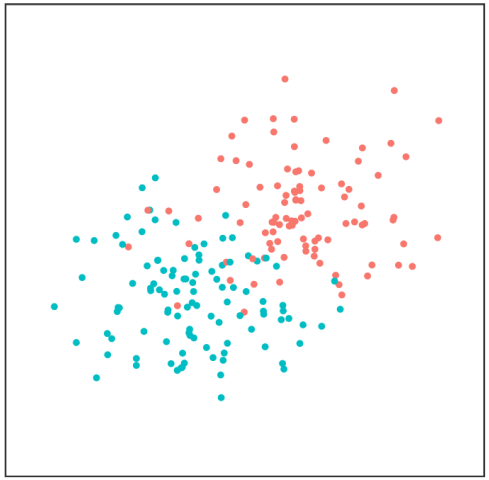


ANN

fixed iterations

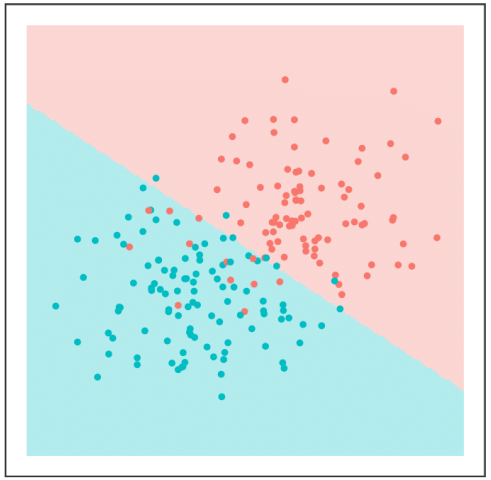
Dataset

mlbench.2dnormals(n=200, cl=2)



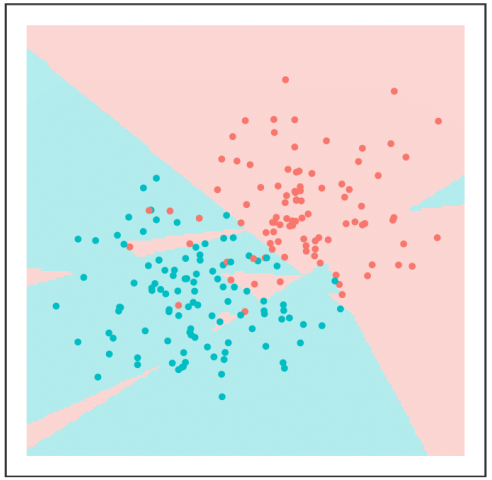
ANN; iters=100; neurons=1

training accuracy: 90.5 % – test accuracy: 92.5 %



ANN; iters=100; neurons=10

training accuracy: 95.5 % – test accuracy: 89 %



ANN; iters=100; neurons=100

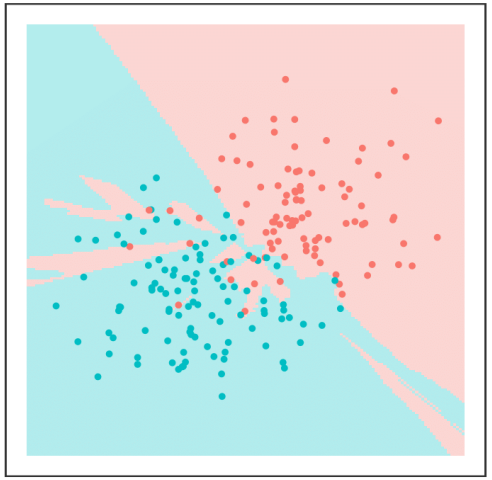
training accuracy: 100 % – test accuracy: 86 %

ANN; iters=100; neurons=500

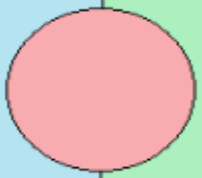
training accuracy: 100 % – test accuracy: 89.5 %

ANN; iters=100; neurons=1000

training accuracy: 100 % – test accuracy: 87.5 %



2D Normals

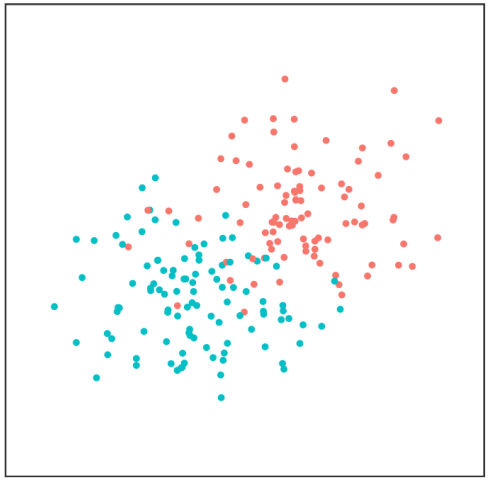


ANN

fixed neurons

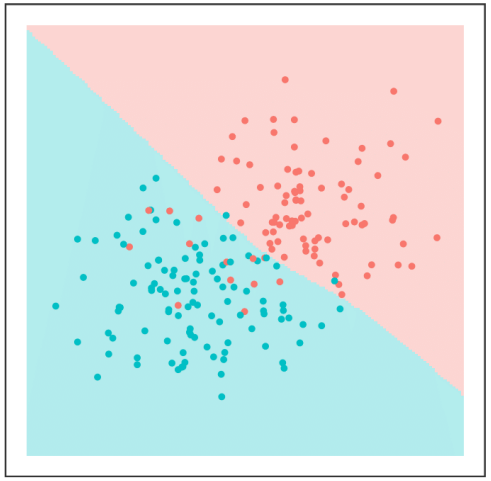
Dataset

mlbench.2dnormals(n=200, cl=2)



ANN; neurons=10; iters=10

training accuracy: 92.5 % – test accuracy: 91 %

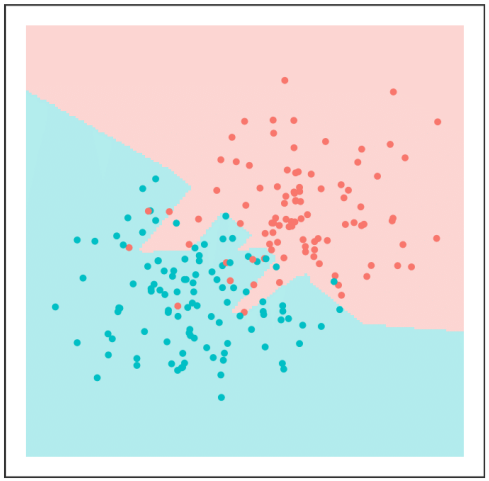


ANN; neurons=10; iters=100

training accuracy: 94.5 % – test accuracy: 89.5 %

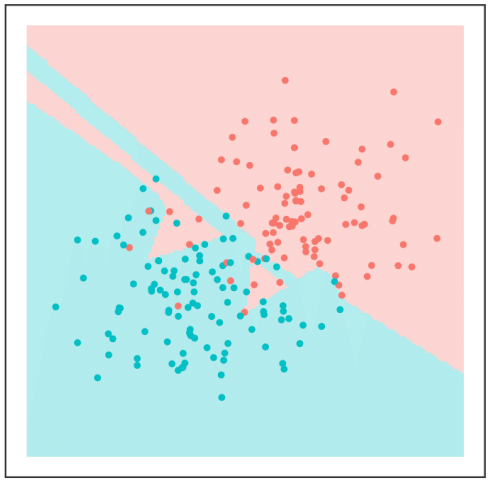
ANN; neurons=10; iters=500

training accuracy: 94 % – test accuracy: 87 %



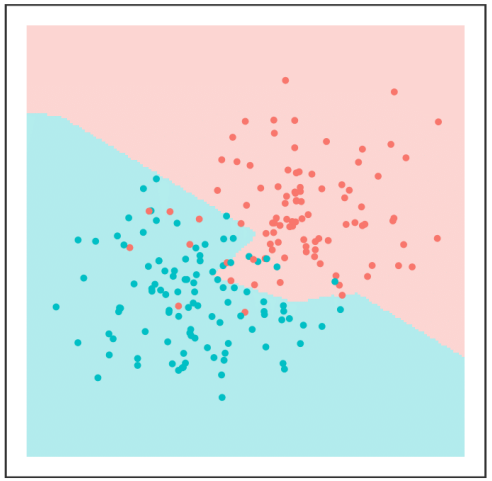
ANN; neurons=10; iters=1000

training accuracy: 96.5 % – test accuracy: 87.5 %



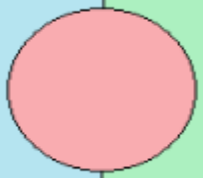
ANN; neurons=10; iters=10000

training accuracy: 93.5 % – test accuracy: 89 %



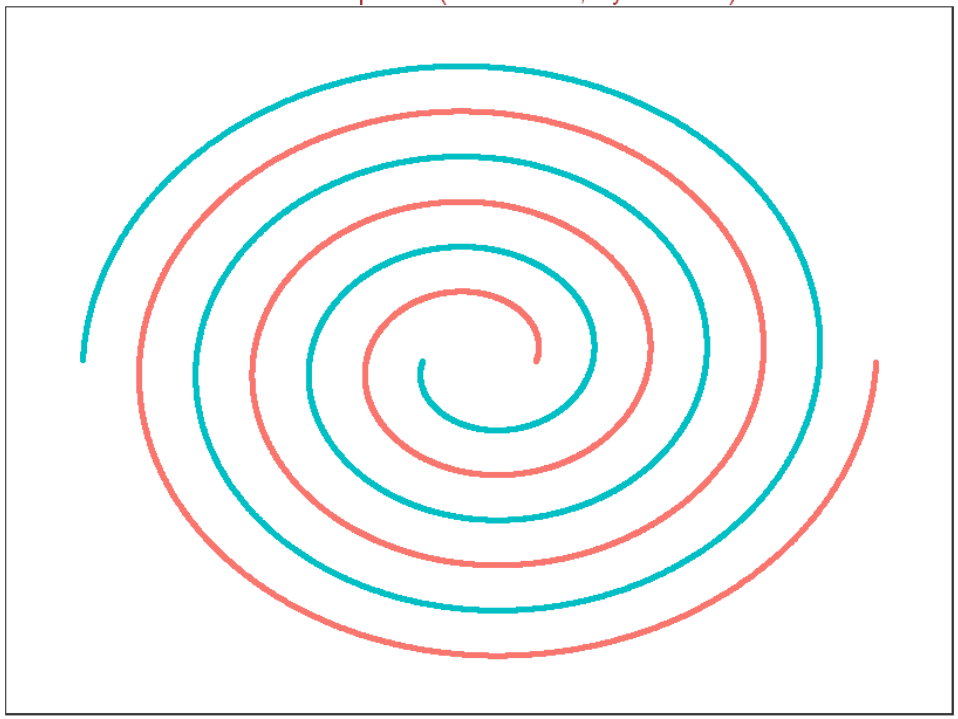
Spirals

3 cycles



Dataset

mlbench.spirals (n = 10000, cycles = 3)



This dataset image contains 10,000 points, so we can se its true form. The next ones will contain a lower number in order to have a better view of the decision boundary of each algorithm.

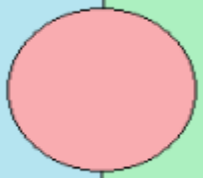
Two Spirals Benchmark Problem

The inputs of the spirals problem are points on two entangled spirals. If *sd* > 0, then Gaussian noise is added to each data point.

Usage:
mlbench.spirals(n, cycles, sd)

Spirals

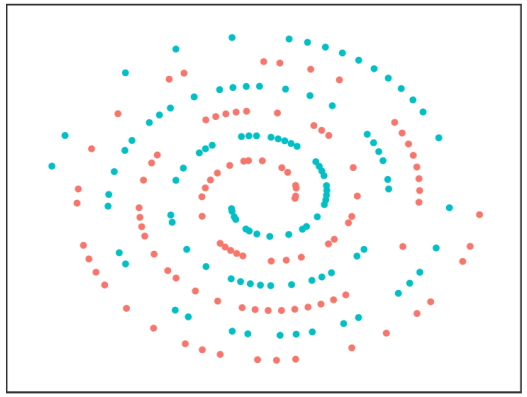
3 cycles



KNN

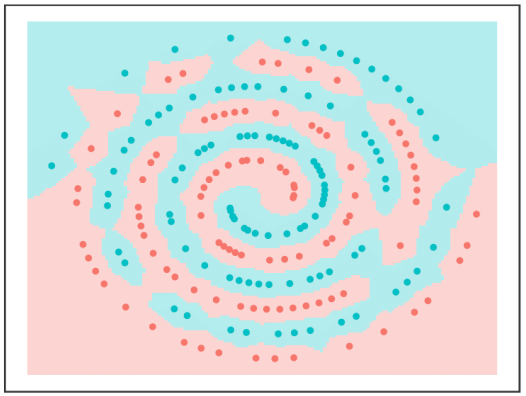
Dataset

mlbench.spirals (n=200, cycles=3, sd=0.0)



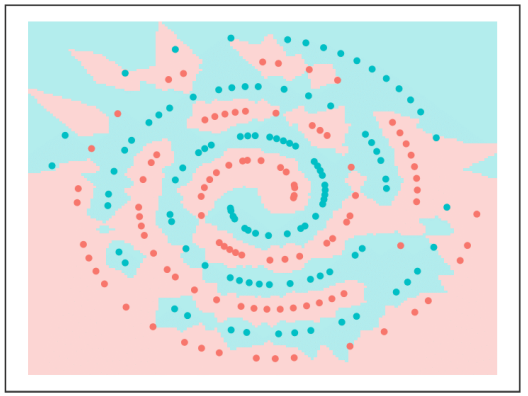
KNN1

training accuracy: 100 % – test accuracy: 94.5 %



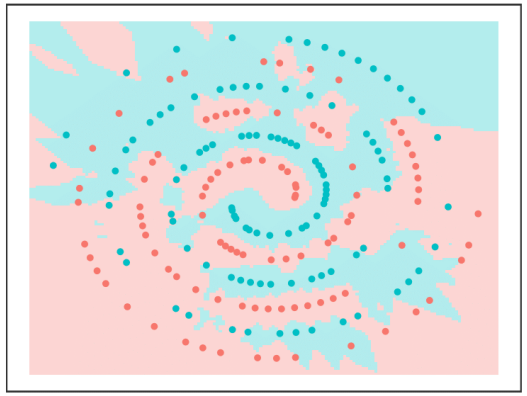
KNN3

training accuracy: 95 % – test accuracy: 79.5 %



KNN5

training accuracy: 81.5 % – test accuracy: 68.5 %

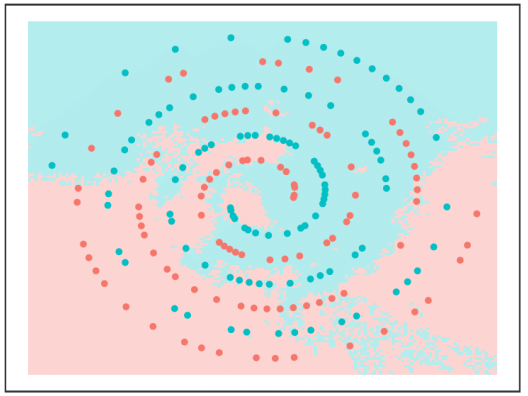


KNN10

training accuracy: 56.5 % – test accuracy: 43.5 %

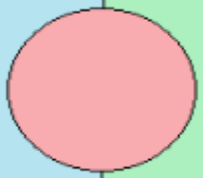
KNN30

training accuracy: 55 % – test accuracy: 53.5 %

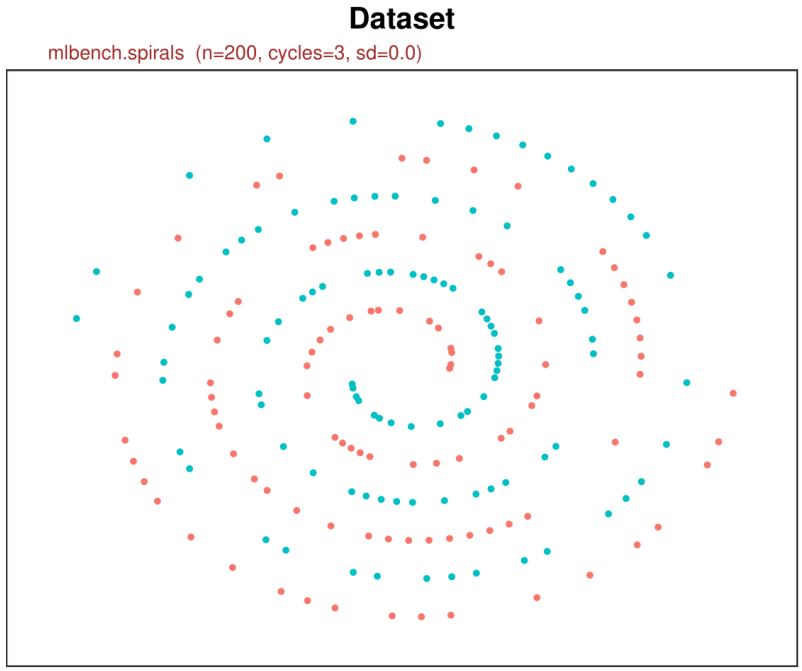


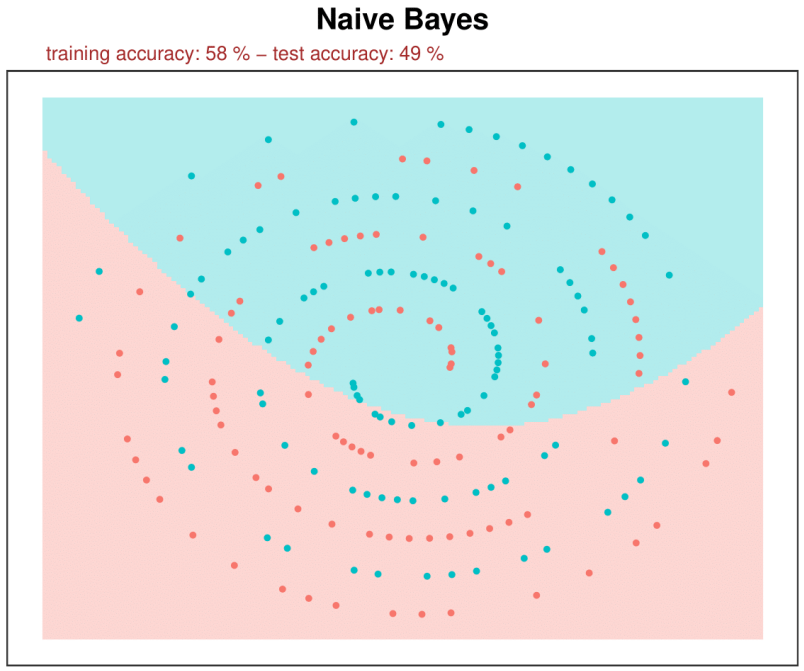
Spirals

3 cycles



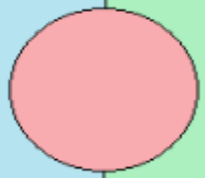
Naive bayes





Spirals

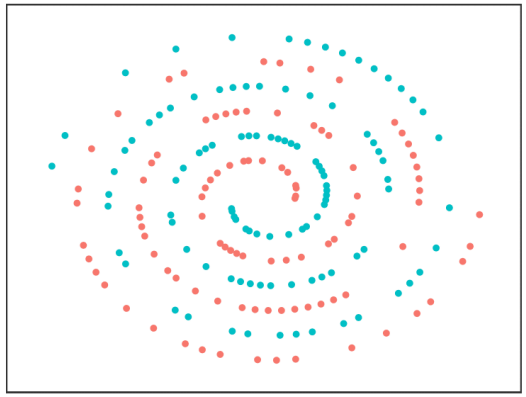
3 cycles



SVM

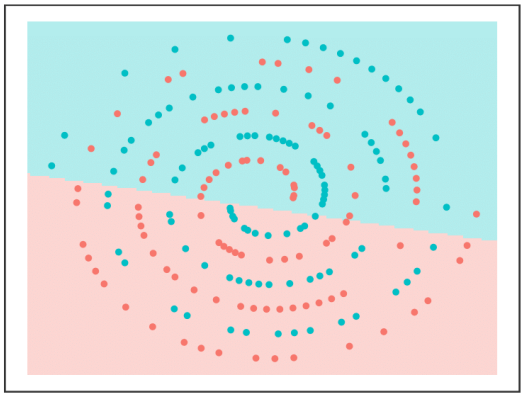
Dataset

mlbench.spirals (n=200, cycles=3, sd=0.0)



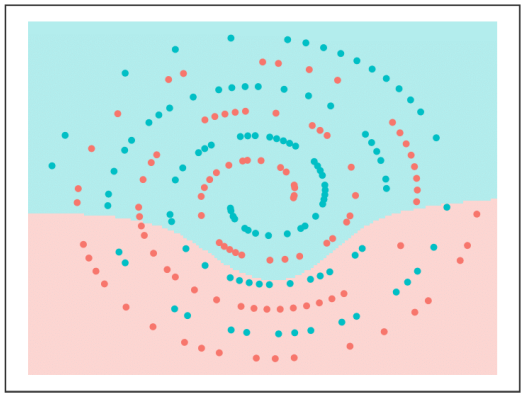
SVMlinear

training accuracy: 55 % – test accuracy: 46 %



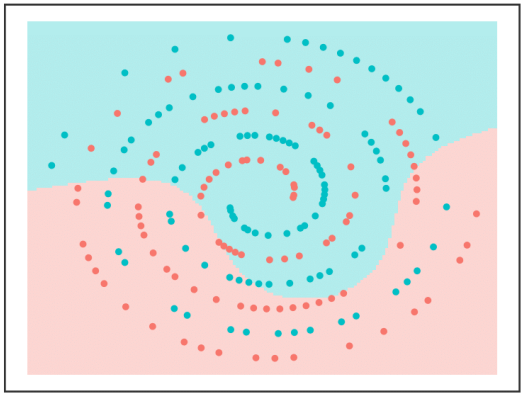
SVMpolynomial

training accuracy: 55 % – test accuracy: 47 %



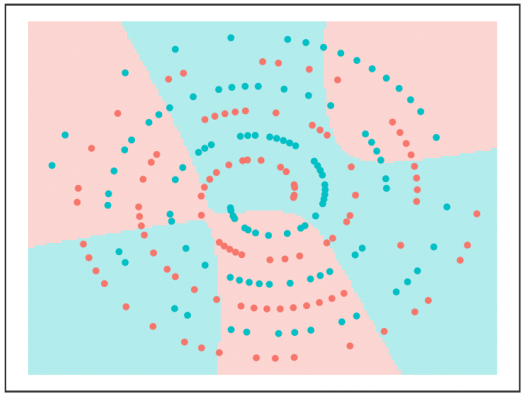
SVMradial

training accuracy: 62 % – test accuracy: 49.5 %



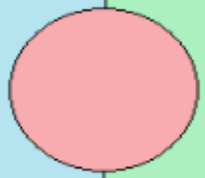
SVMsigmoid

training accuracy: 45.5 % – test accuracy: 46.5 %



Spirals

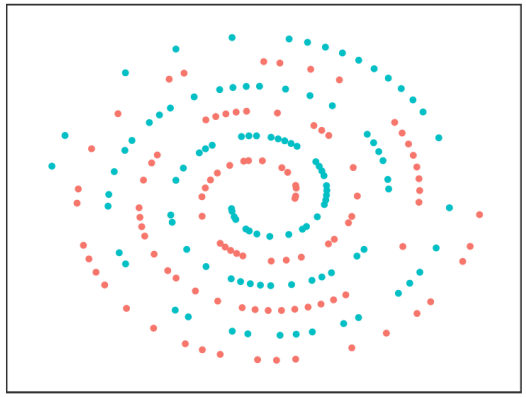
3 cycles



TREES

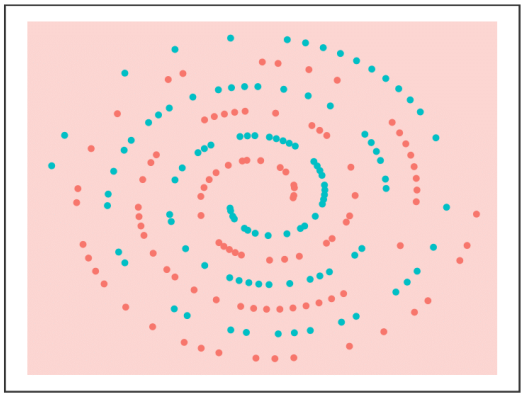
Dataset

mlbench.spirals (n=200, cycles=3, sd=0.0)



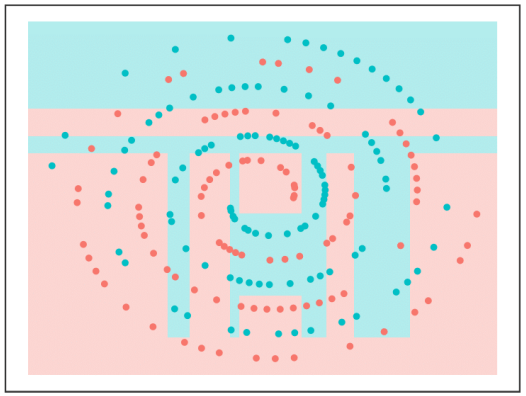
TreesC.50

training accuracy: 51.5 % – test accuracy: 48.5 %



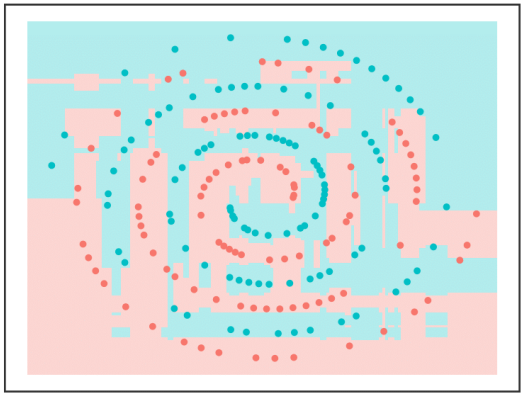
TreesCART

training accuracy: 80.5 % – test accuracy: 63.5 %



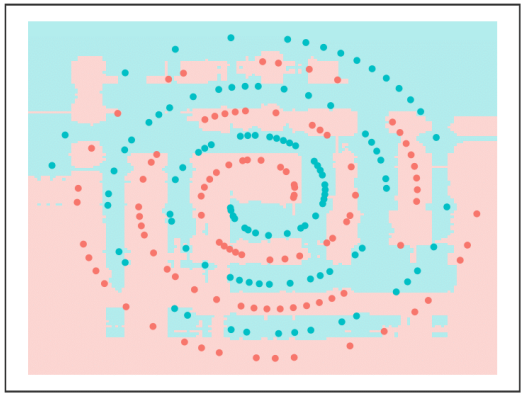
TreesRandom Forest 3 trees

training accuracy: 97 % – test accuracy: 75.5 %



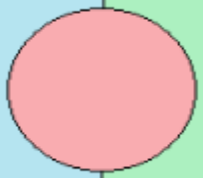
TreesRandom Forest 128 trees

training accuracy: 100 % – test accuracy: 75 %



Spirals

3 cycles

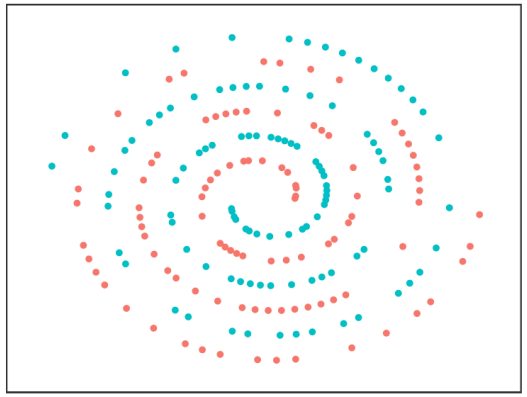


ANN

fixed iterations

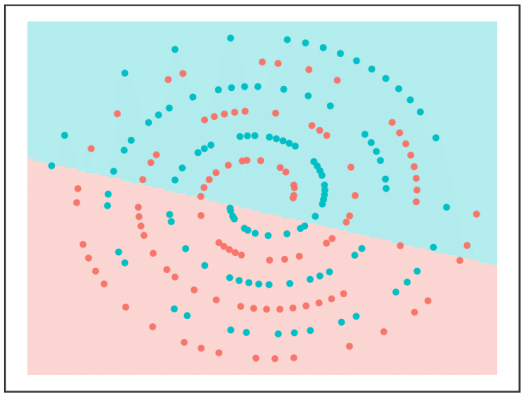
Dataset

mlbench.spirals(n=200, cycles=3, sd=0.0)



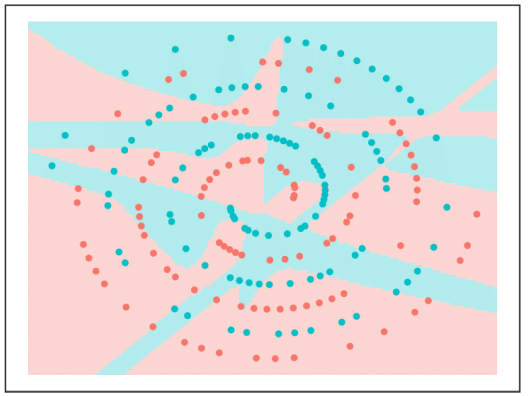
ANN; iters=100; neurons=1

training accuracy: 54.5 % – test accuracy: 47 %



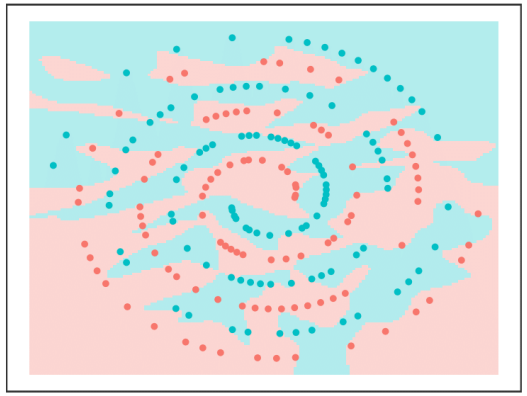
ANN; iters=100; neurons=10

training accuracy: 73 % – test accuracy: 45.5 %



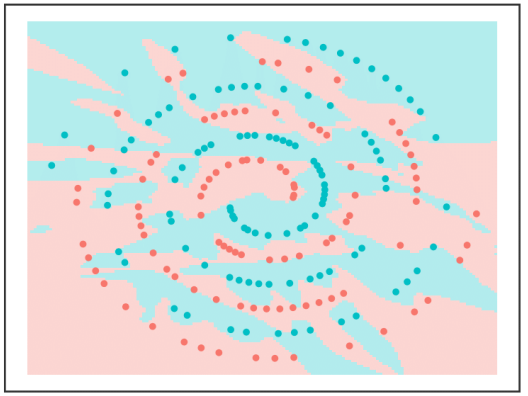
ANN; iters=100; neurons=100

training accuracy: 93.5 % – test accuracy: 60 %



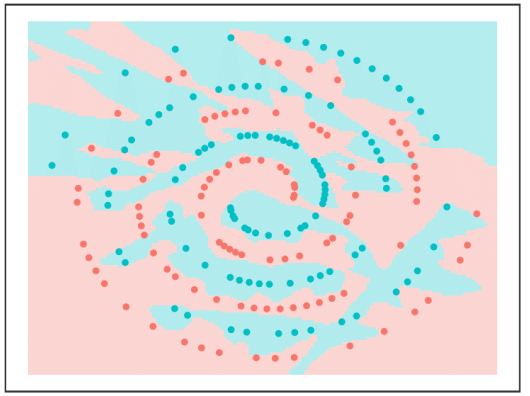
ANN; iters=100; neurons=500

training accuracy: 100 % – test accuracy: 58 %



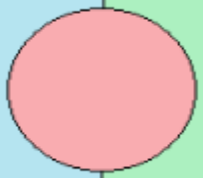
ANN; iters=100; neurons=1000

training accuracy: 100 % – test accuracy: 59.5 %



Spirals

3 cycles

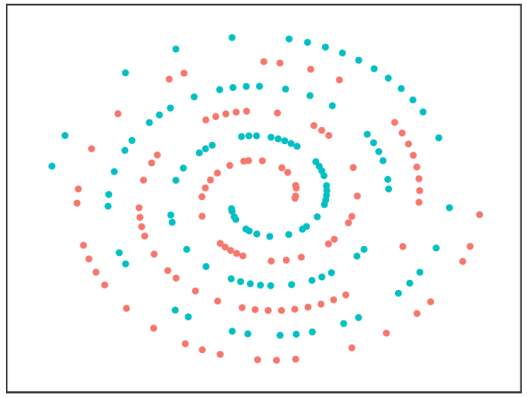


ANN

fixed neurons

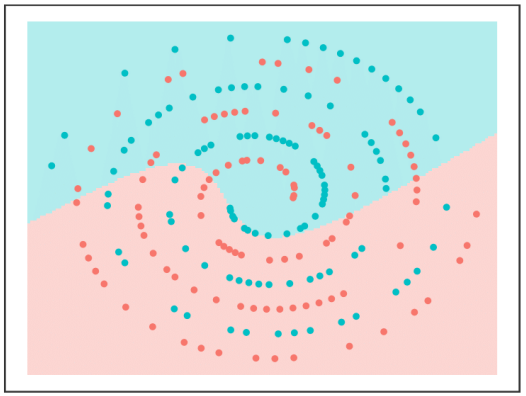
Dataset

mlbench.spirals(n=200, cycles=3, sd=0.0)



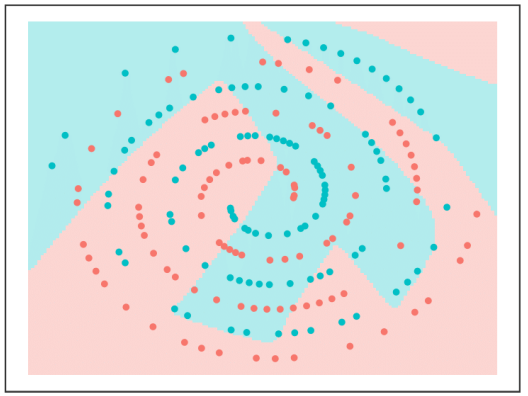
ANN; neurons=10; iters=10

training accuracy: 59.5 % – test accuracy: 50 %



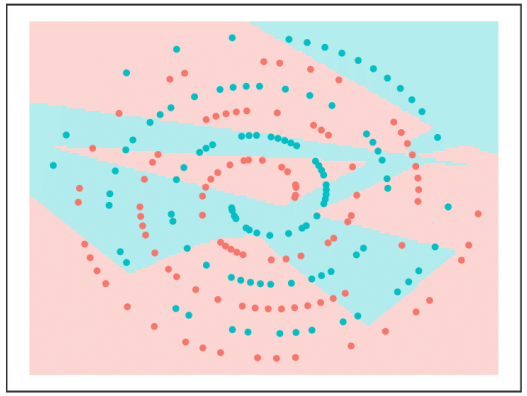
ANN; neurons=10; iters=100

training accuracy: 64.5 % – test accuracy: 51.5 %



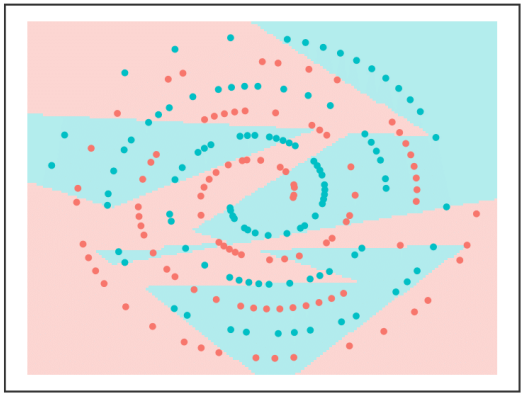
ANN; neurons=10; iters=500

training accuracy: 70.5 % – test accuracy: 57 %



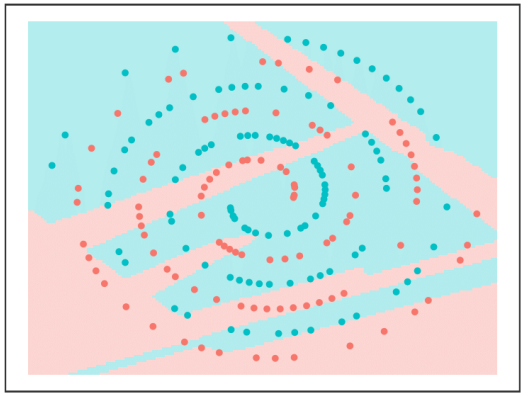
ANN; neurons=10; iters=1000

training accuracy: 70.5 % – test accuracy: 57 %

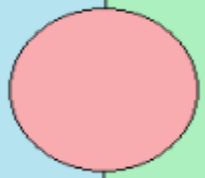


ANN; neurons=10; iters=10000

training accuracy: 75 % – test accuracy: 54.5 %

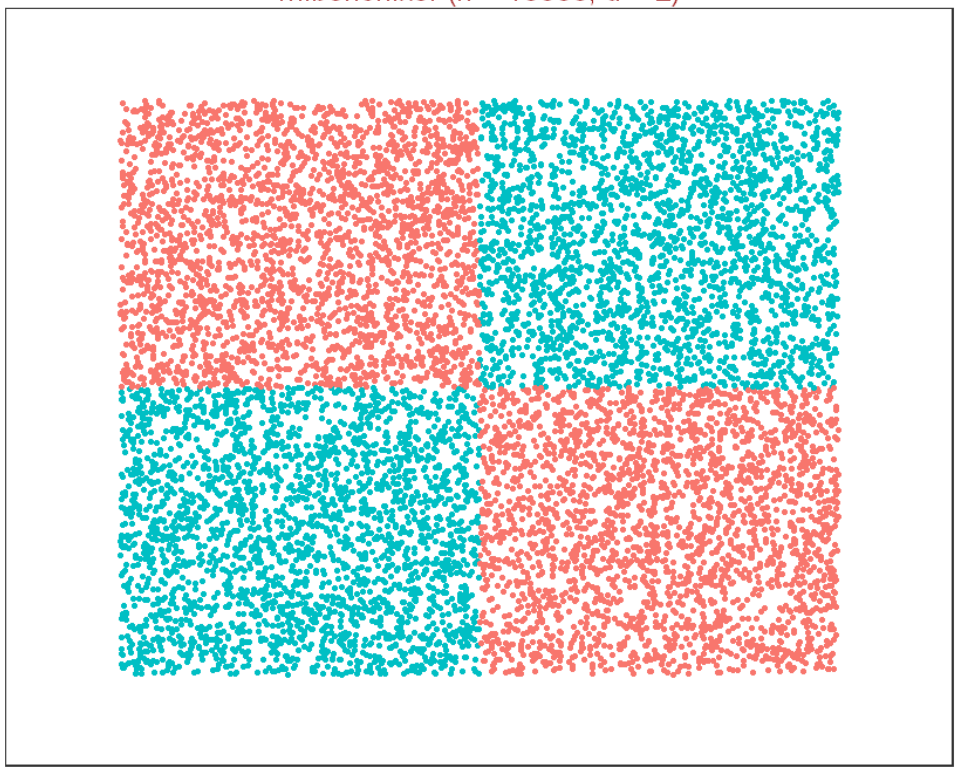


XOR



Dataset

mlbench.xor (n = 10000, d = 2)



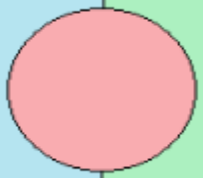
This dataset image contains 10,000 points, so we can se its true form. The next ones will contain a low number in order to have a better view of the decision boundary of each algorithm.

Continuous XOR Benchmark Problem

The inputs of the XOR problem are uniformly distributed on the *d*-dimensional cube with corners +-1. Each pair of opposite corners form one class, hence the total number of classes is 2^{d-1}

Usage:
mlbench.xor(n, d)

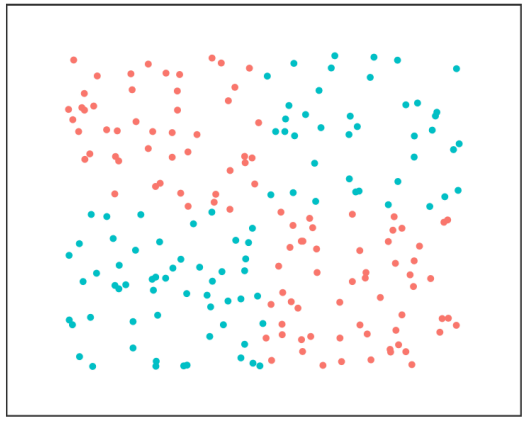
XOR



KNN

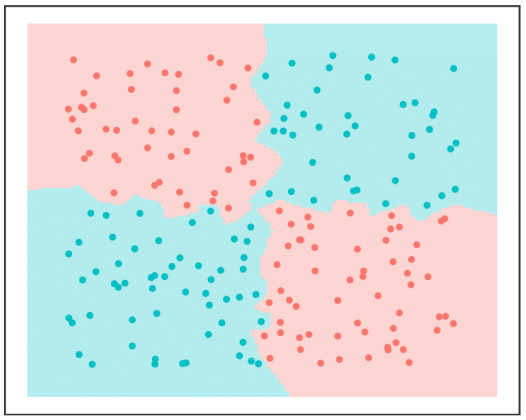
Dataset

mlbench.xor (n=200, d=2)



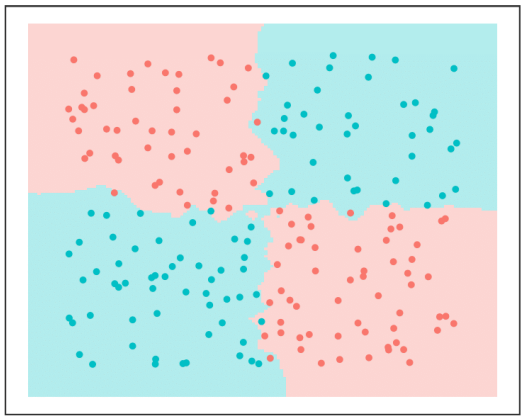
KNN1

training accuracy: 100 % – test accuracy: 94.5 %



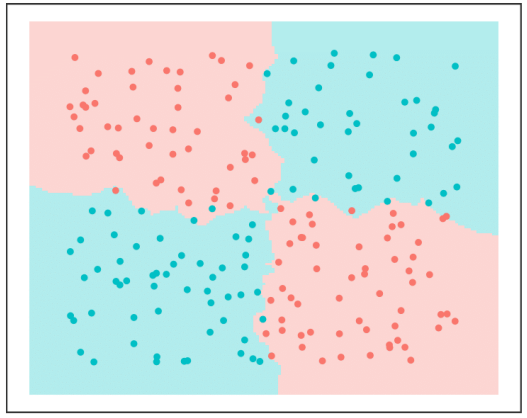
KNN3

training accuracy: 97 % – test accuracy: 93 %



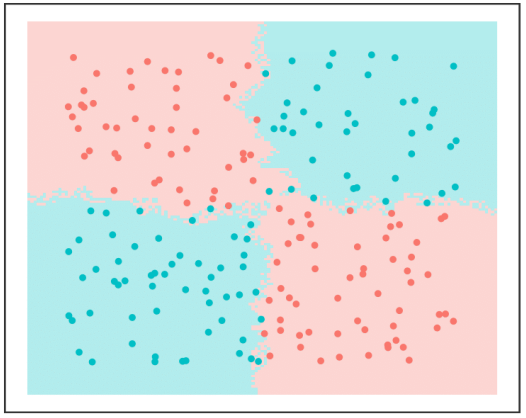
KNN5

training accuracy: 94.5 % – test accuracy: 92.5 %



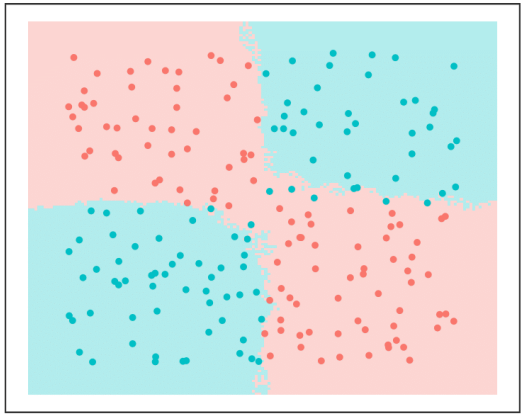
KNN10

training accuracy: 95.5 % – test accuracy: 93.5 %

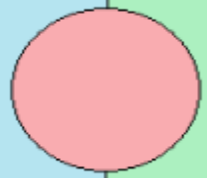


KNN30

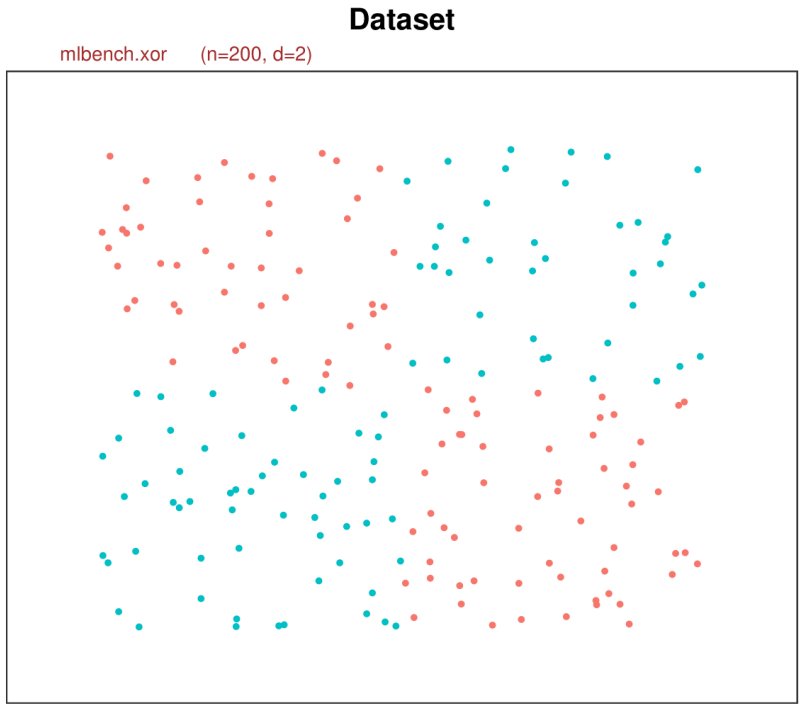
training accuracy: 95 % – test accuracy: 92 %

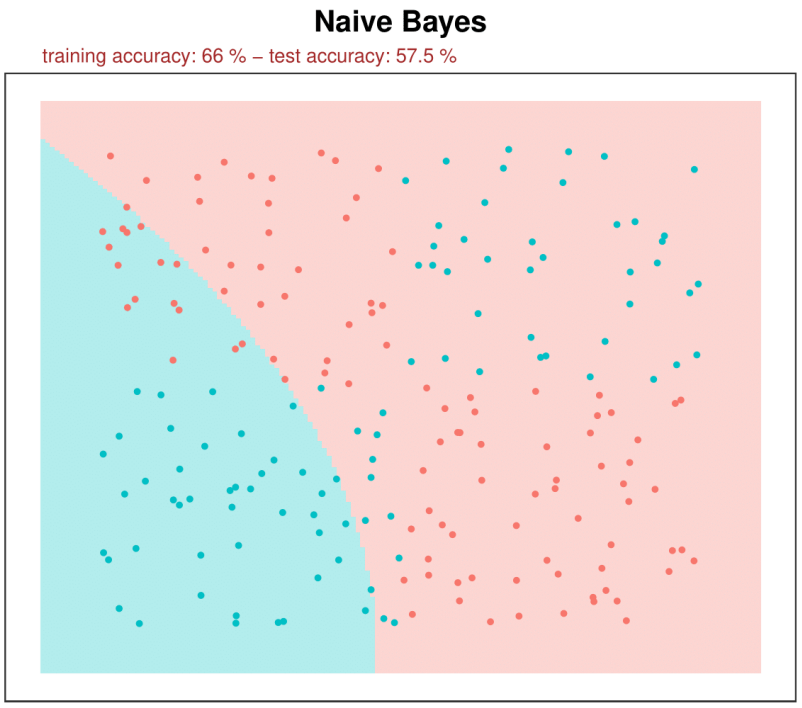


XOR

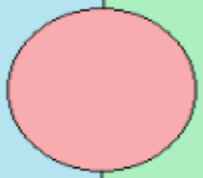


Naïve Bayes





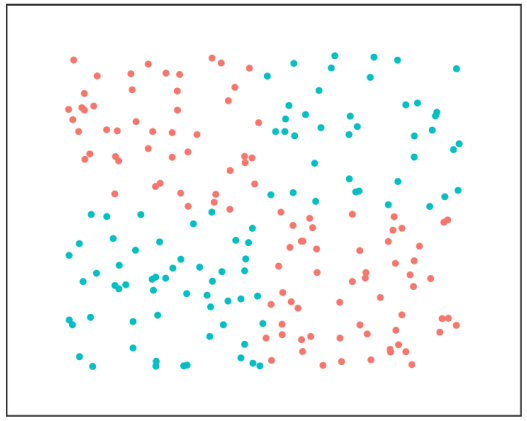
XOR



SVM

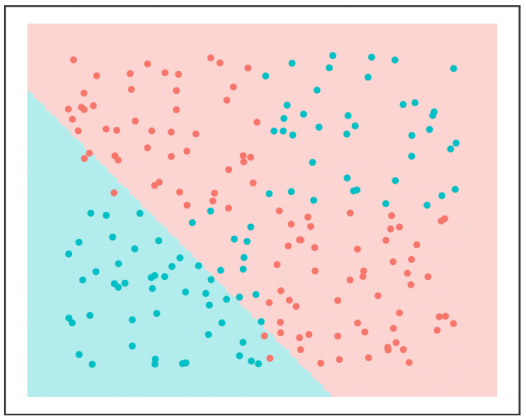
Dataset

mlbench.xor (n=200, d=2)



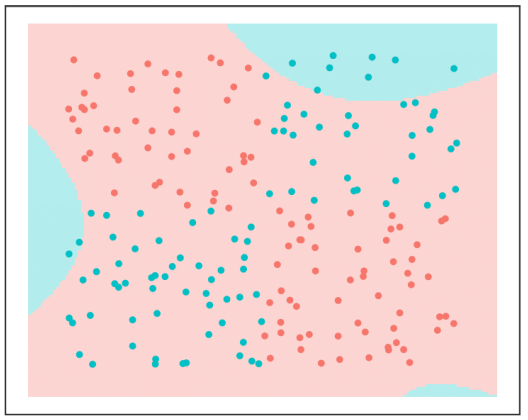
SVMlinear

training accuracy: 72 % – test accuracy: 60.5 %



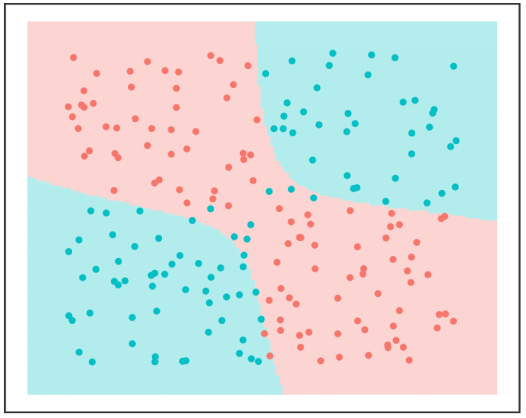
SVMpolynomial

training accuracy: 57.5 % – test accuracy: 59 %



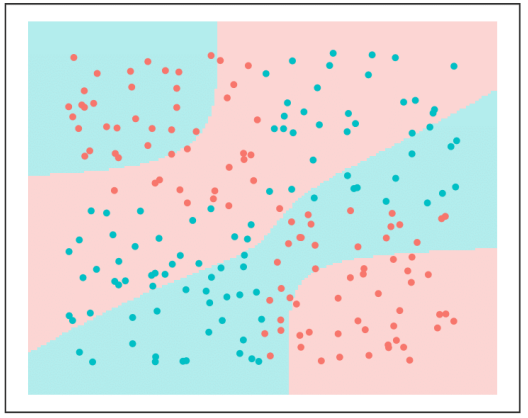
SVMradial

training accuracy: 95 % – test accuracy: 91.5 %

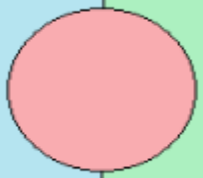


SVMsigmoid

training accuracy: 45 % – test accuracy: 41 %



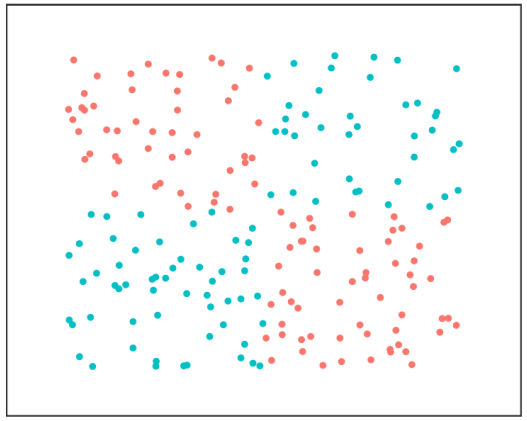
XOR



TREES

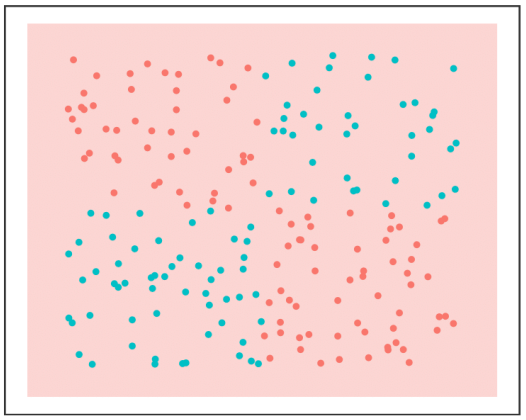
Dataset

mlbench.xor (n=200, d=2)



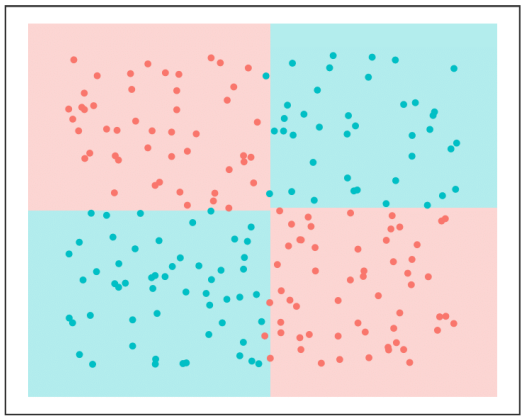
TreesC.50

training accuracy: 52.5 % – test accuracy: 52.5 %



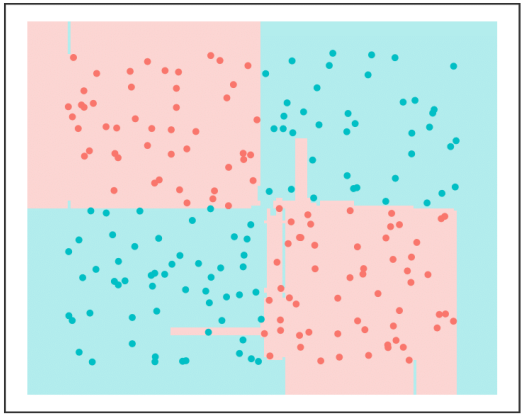
TreesCART

training accuracy: 98.5 % – test accuracy: 96.5 %



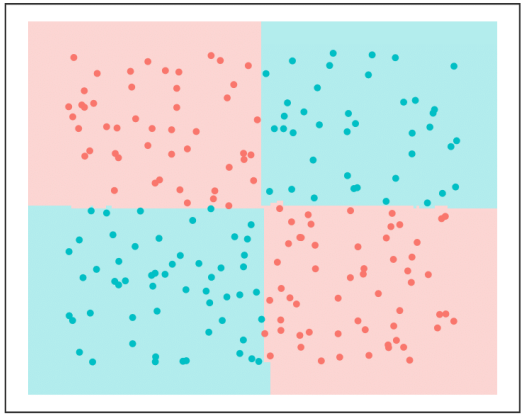
TreesRandom Forest 3 trees

training accuracy: 99.5 % – test accuracy: 95.5 %

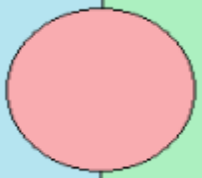


TreesRandom Forest 128 trees

training accuracy: 100 % – test accuracy: 98.5 %



XOR

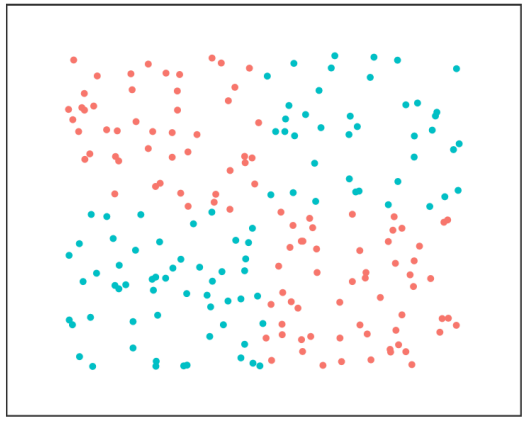


ANN

fixed iterations

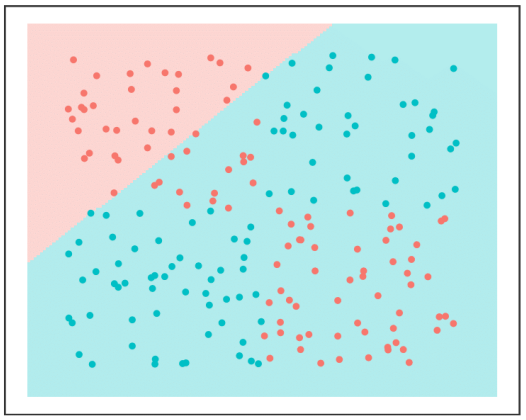
Dataset

mlbench.xor(n=200, d=2)



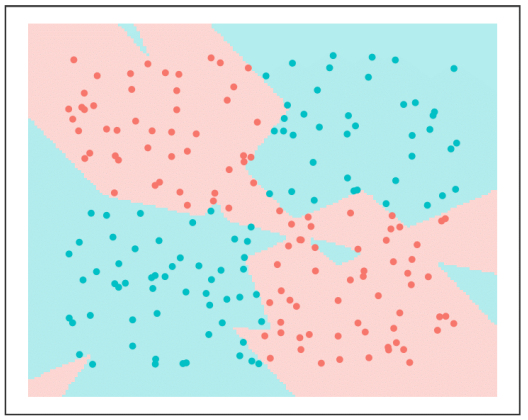
ANN; iters=100; neurons=1

training accuracy: 62.5 % – test accuracy: 67.5 %



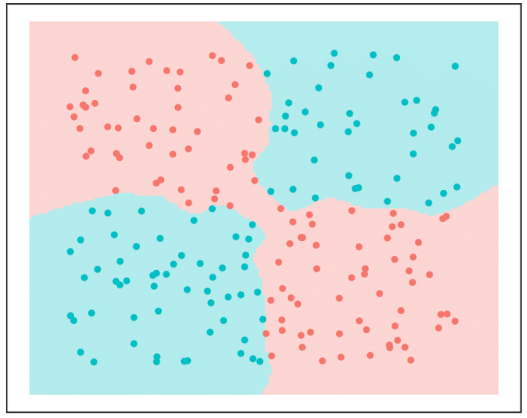
ANN; iters=100; neurons=10

training accuracy: 100 % – test accuracy: 91 %



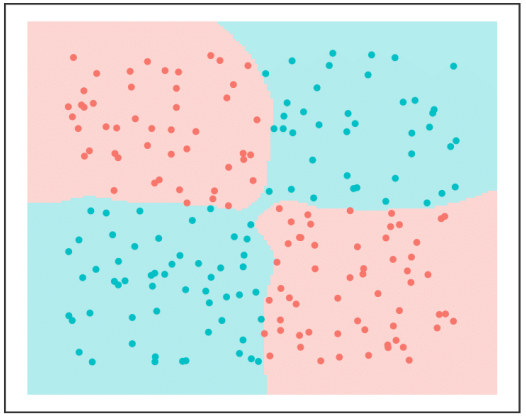
ANN; iters=100; neurons=100

training accuracy: 100 % – test accuracy: 97.5 %



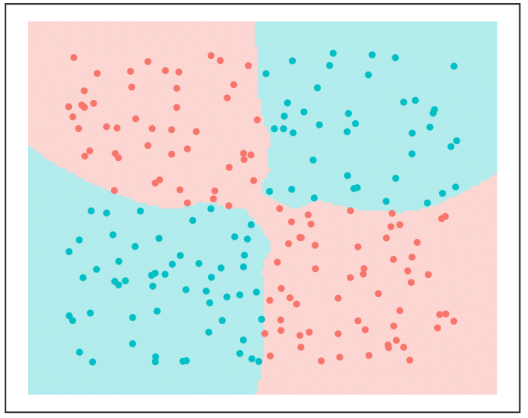
ANN; iters=100; neurons=500

training accuracy: 100 % – test accuracy: 95 %

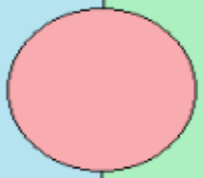


ANN; iters=100; neurons=1000

training accuracy: 100 % – test accuracy: 96 %



XOR

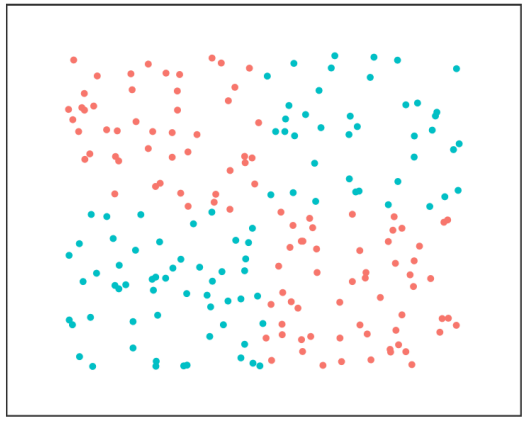


ANN

fixed neurons

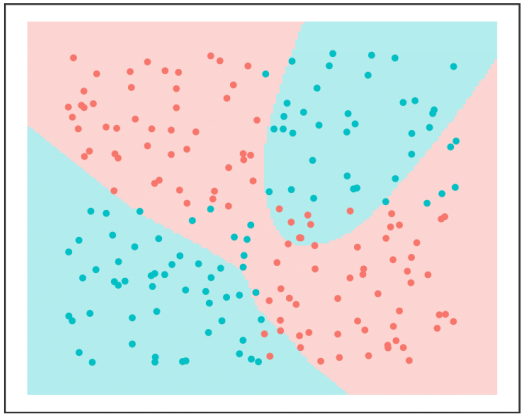
Dataset

mlbench.xor(n=200, d=2)



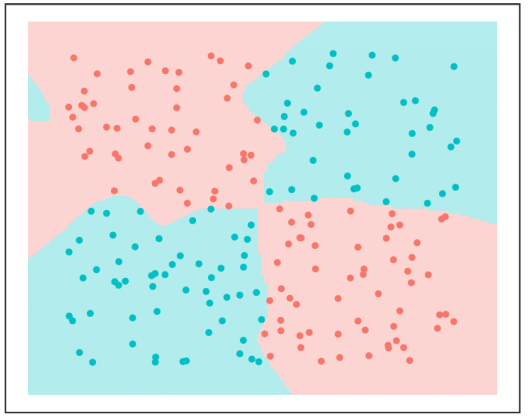
ANN; neurons=10; iters=10

training accuracy: 87 % – test accuracy: 81.5 %



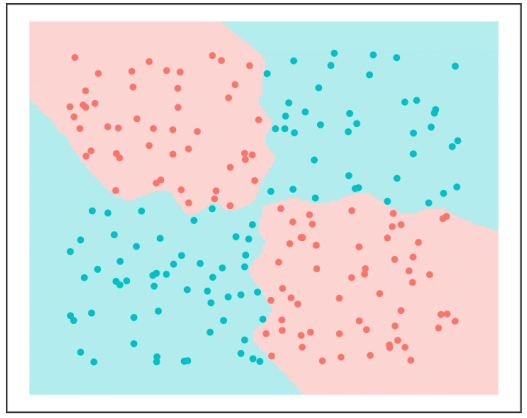
ANN; neurons=10; iters=100

training accuracy: 100 % – test accuracy: 96 %



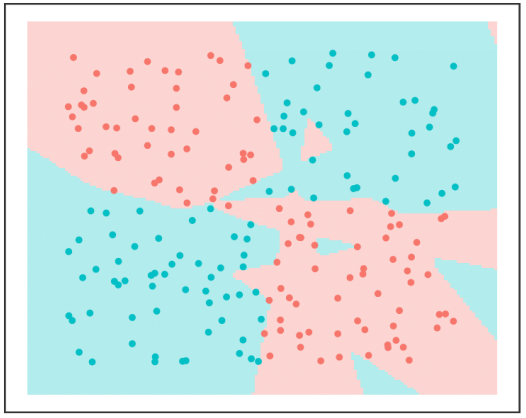
ANN; neurons=10; iters=500

training accuracy: 100 % – test accuracy: 93 %



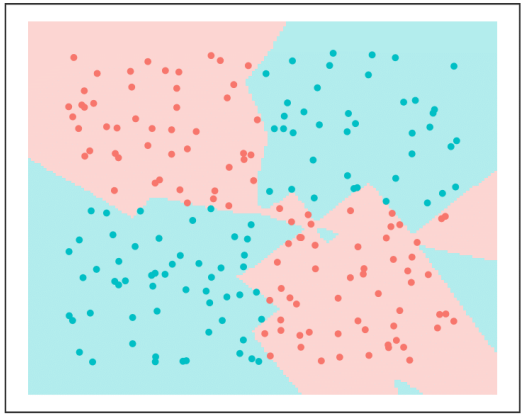
ANN; neurons=10; iters=1000

training accuracy: 100 % – test accuracy: 92.5 %

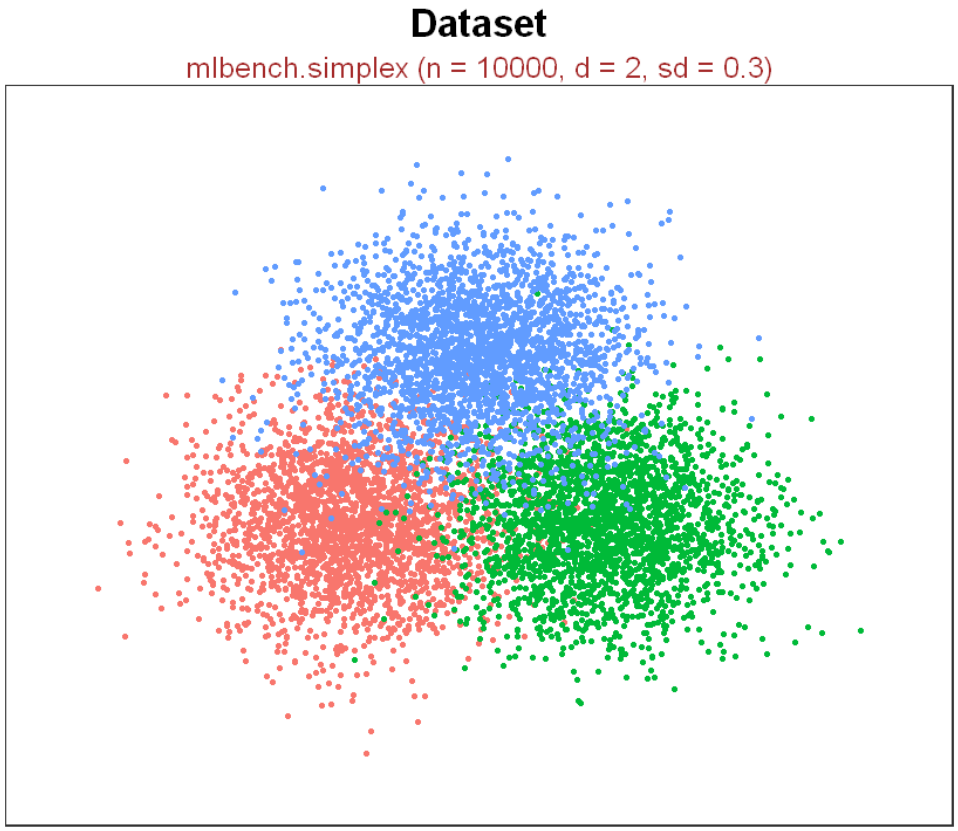


ANN; neurons=10; iters=10000

training accuracy: 100 % – test accuracy: 90 %



Simplex



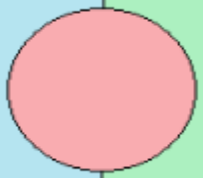
This dataset image contains 10,000 points, so we can se its true form. The next ones will contain a low number in order to have a better view of the decision boundary of each algorithm.

Corners of d-dimensional Simplex

The created data are *d*-dimensional spherical Gaussians with standard deviation *sd* and means at the corners of a *d*-dimensional simplex. The number of classes is *d* + 1.

Usage:
mlbench.simplex(n, d , sides, sd, center)

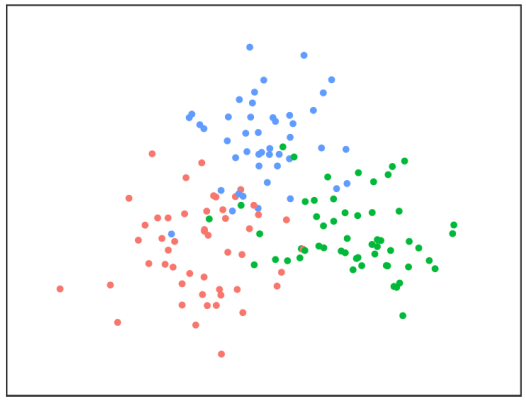
Simplex



KNN

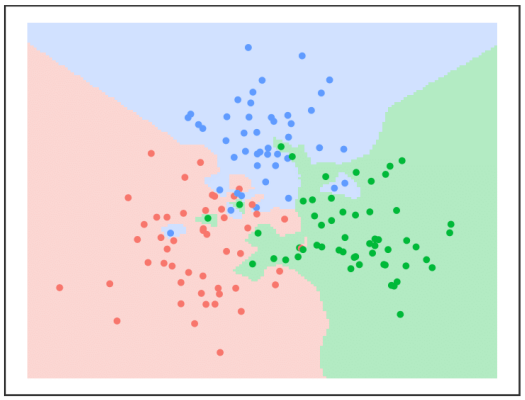
Dataset

mlbench.simplex (n=200, d=2, sd = 0.3)



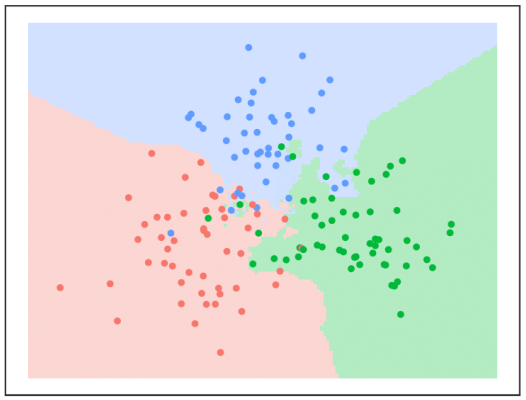
KNN1

training accuracy: 100 % – test accuracy: 88 %



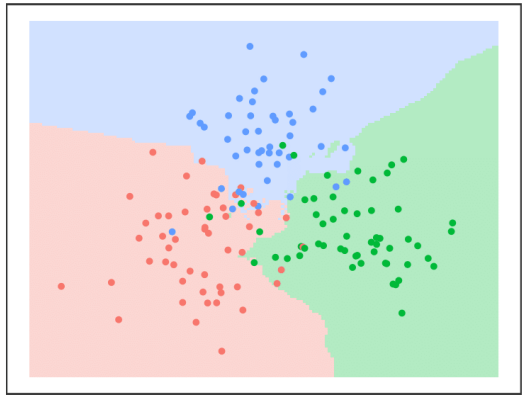
KNN3

training accuracy: 90.67 % – test accuracy: 89.33 %



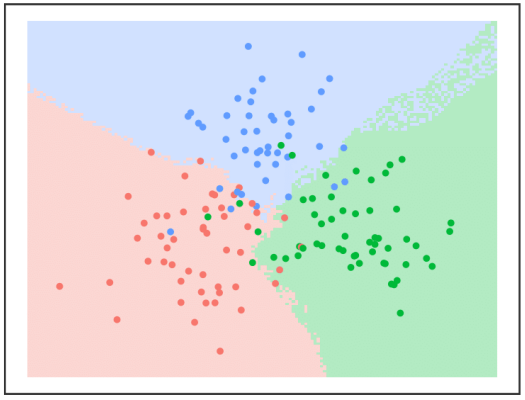
KNN5

training accuracy: 86.67 % – test accuracy: 90 %



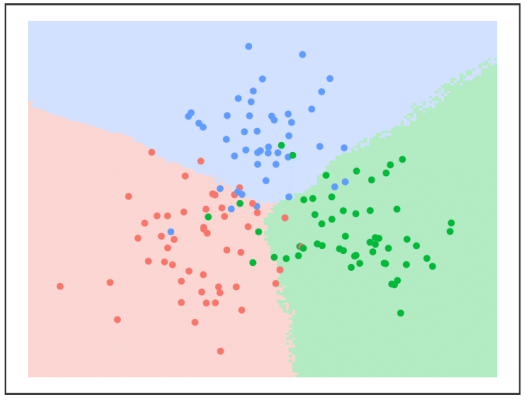
KNN10

training accuracy: 86.67 % – test accuracy: 88.67 %

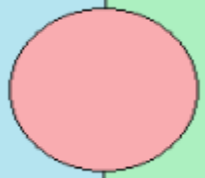


KNN30

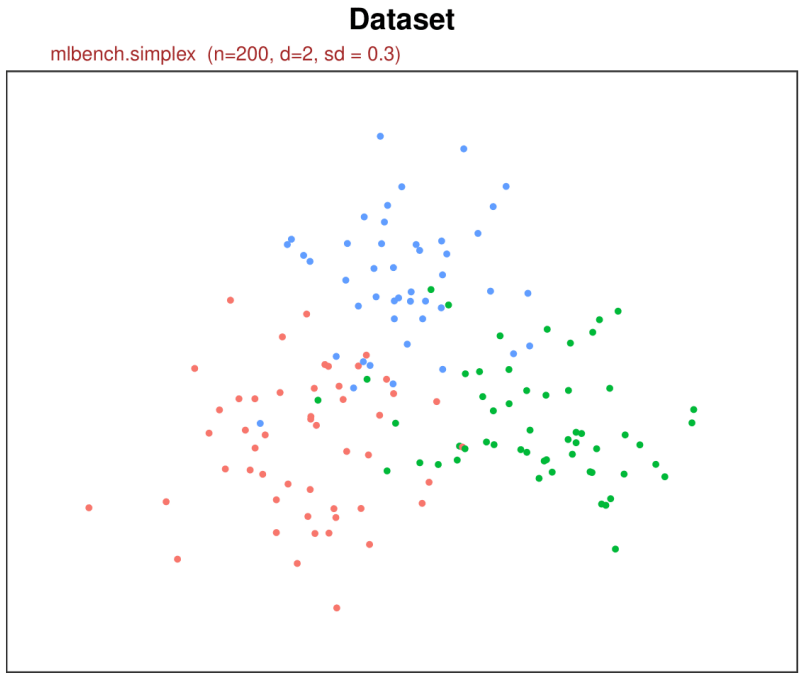
training accuracy: 87.33 % – test accuracy: 88.67 %

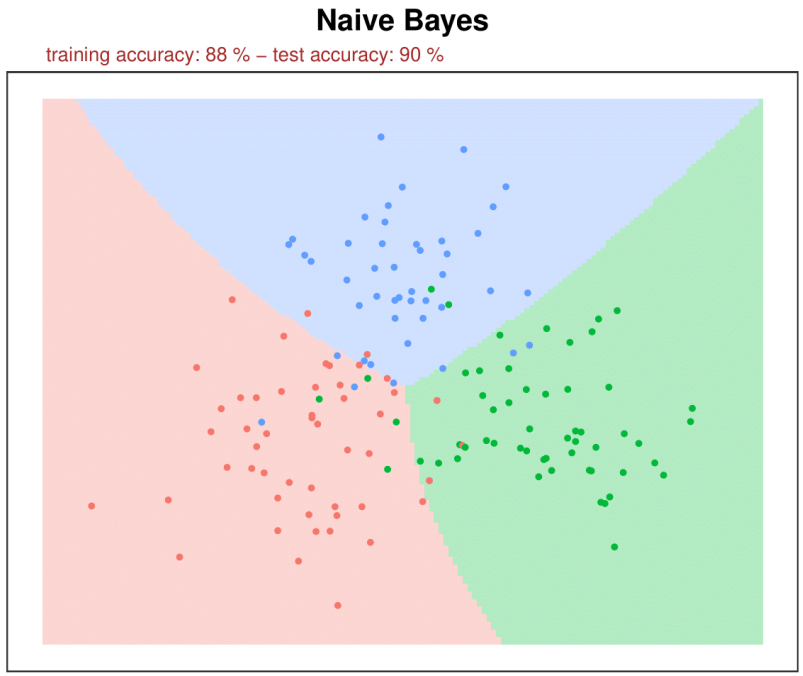


Simplex

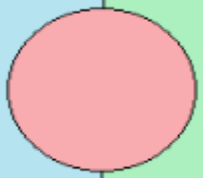


Naïve Bayes





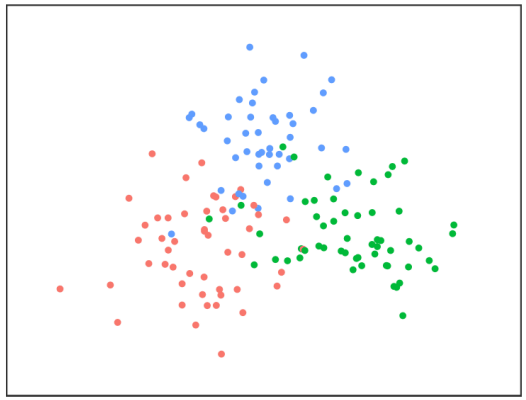
Simplex



SVM

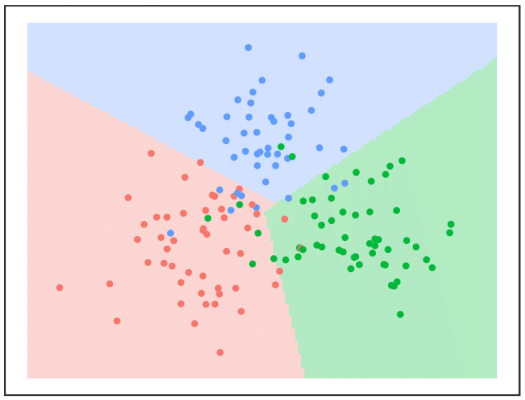
Dataset

mlbench.simplex (n=200, d=2, sd = 0.3)



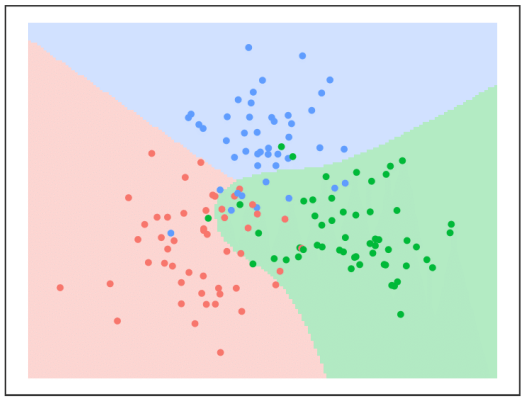
SVMlinear

training accuracy: 86.67 % – test accuracy: 90.67 %



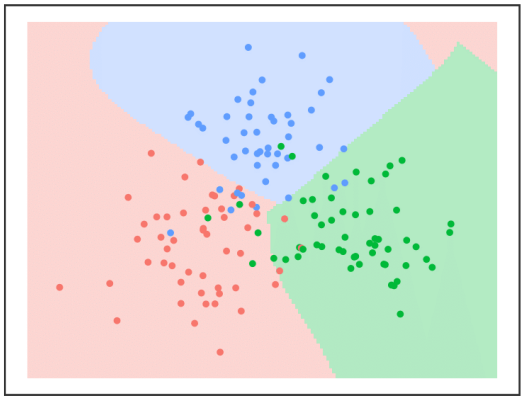
SVMpolynomial

training accuracy: 83.33 % – test accuracy: 85.33 %



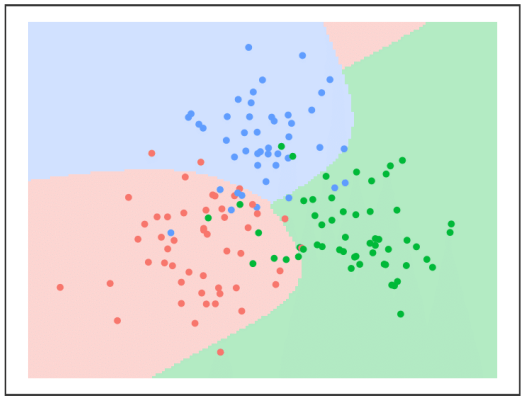
SVMradial

training accuracy: 88 % – test accuracy: 91.33 %

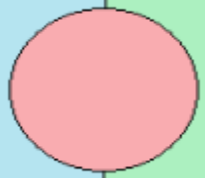


SVMsigmoid

training accuracy: 84.67 % – test accuracy: 88.67 %



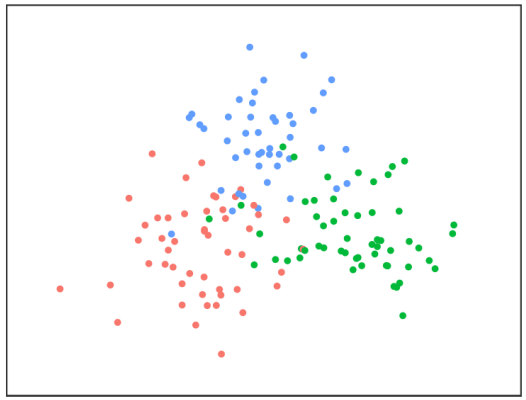
Simplex



Trees

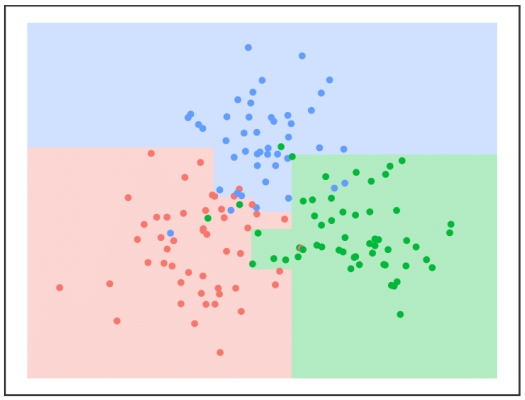
Dataset

mlbench.simplex (n=200, d=2, sd = 0.3)



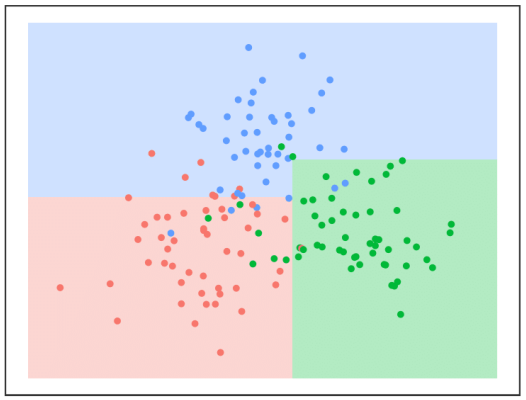
TreesC.50

training accuracy: 94 % – test accuracy: 88 %



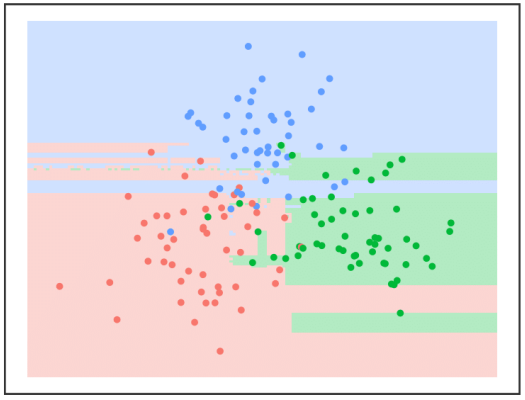
TreesCART

training accuracy: 86.67 % – test accuracy: 87.33 %



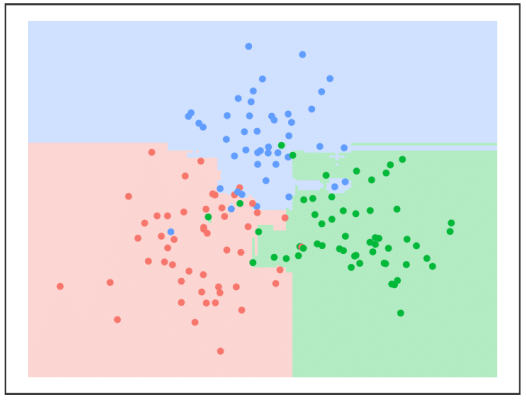
TreesRandom Forest 3 trees

training accuracy: 97.33 % – test accuracy: 84.67 %

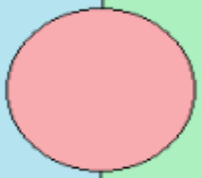


TreesRandom Forest 128 trees

training accuracy: 100 % – test accuracy: 87.33 %



Simplex

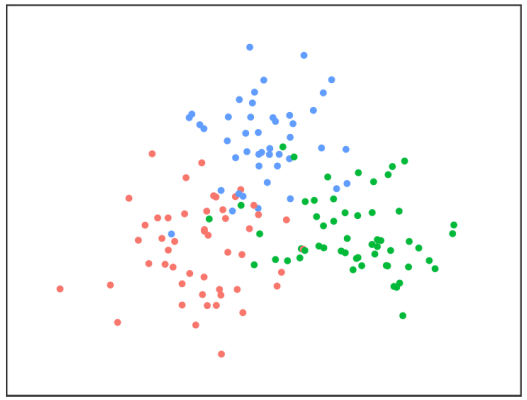


ANN

fixed iterations

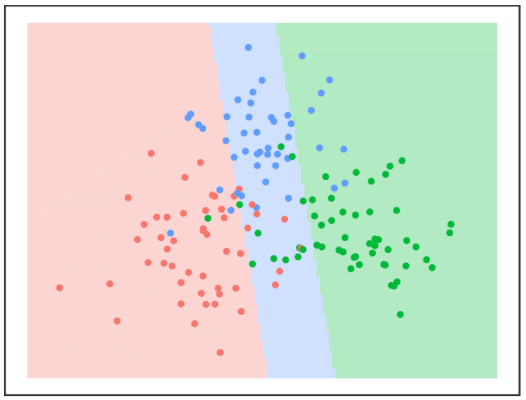
Dataset

mlbench.simplex(n=200, d=2, sd = 0.3)



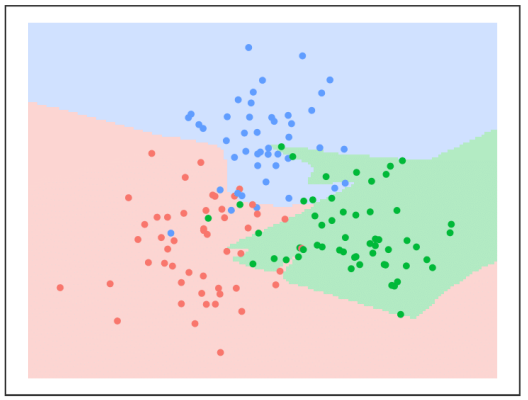
ANN; iters=100; neurons=1

training accuracy: 75.33 % – test accuracy: 70 %



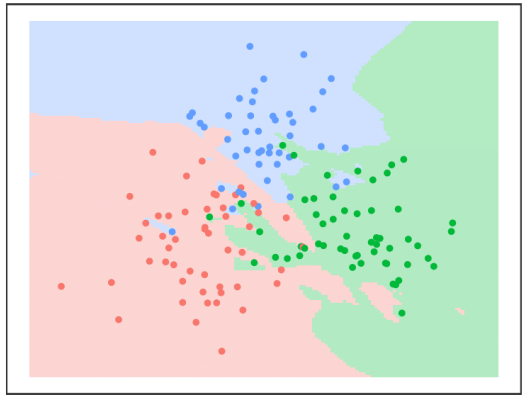
ANN; iters=100; neurons=10

training accuracy: 92.67 % – test accuracy: 85.33 %



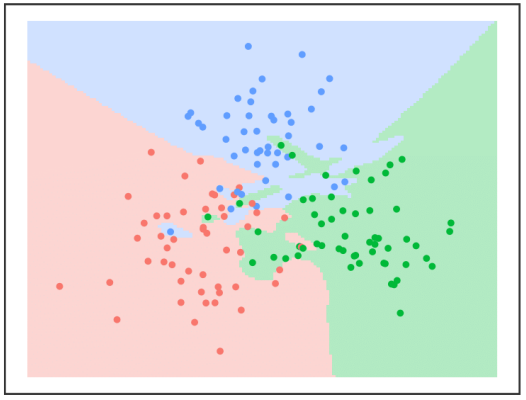
ANN; iters=100; neurons=100

training accuracy: 100 % – test accuracy: 81.33 %



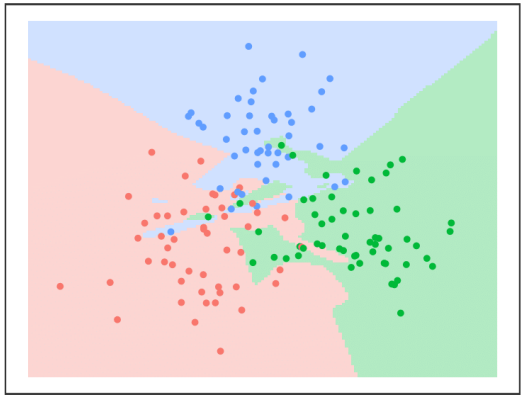
ANN; iters=100; neurons=500

training accuracy: 98 % – test accuracy: 83.33 %

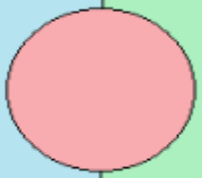


ANN; iters=100; neurons=1000

training accuracy: 98 % – test accuracy: 80.67 %



Simplex

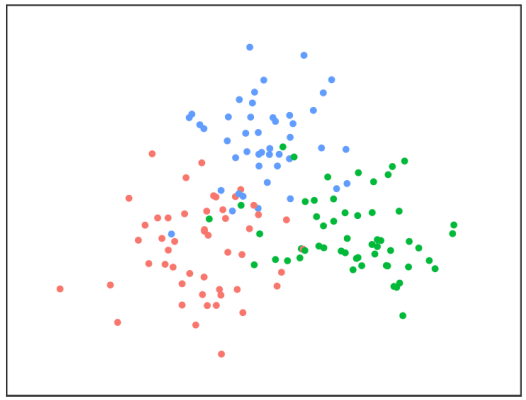


ANN

fixed neurons

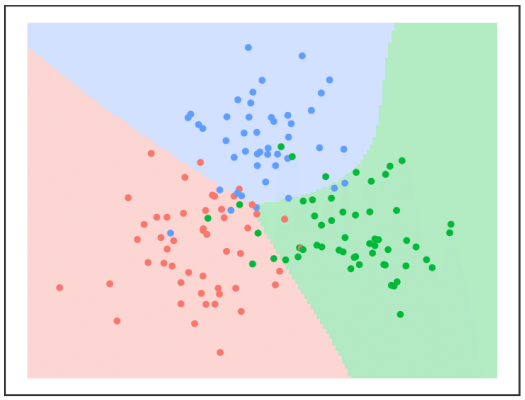
Dataset

mlbench.simplex(n=200, d=2, sd = 0.3)



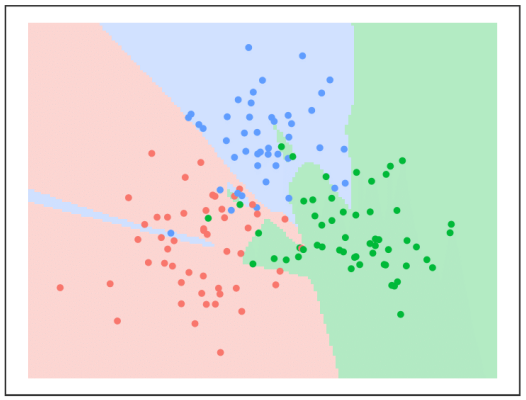
ANN; neurons=10; iters=10

training accuracy: 86.67 % – test accuracy: 88.67 %



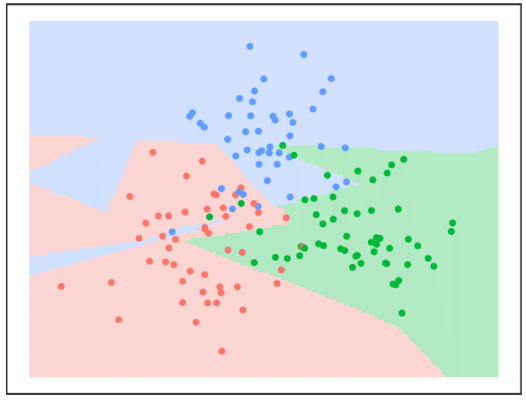
ANN; neurons=10; iters=100

training accuracy: 94 % – test accuracy: 85.33 %



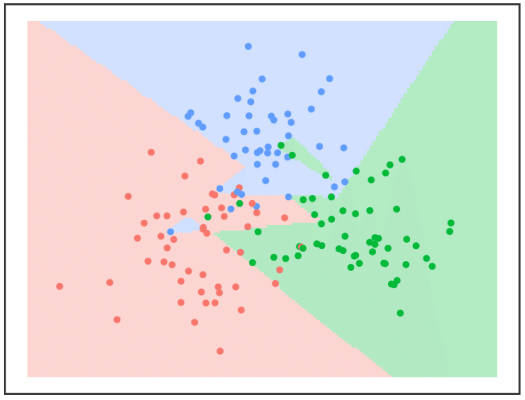
ANN; neurons=10; iters=500

training accuracy: 92 % – test accuracy: 82.67 %



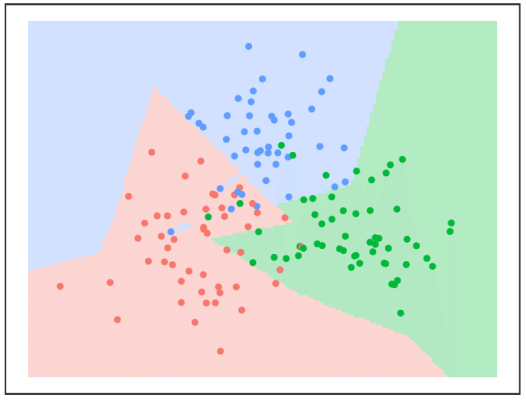
ANN; neurons=10; iters=1000

training accuracy: 94.67 % – test accuracy: 85.33 %

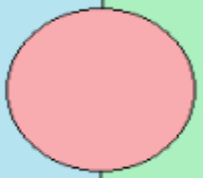


ANN; neurons=10; iters=10000

training accuracy: 92 % – test accuracy: 86 %

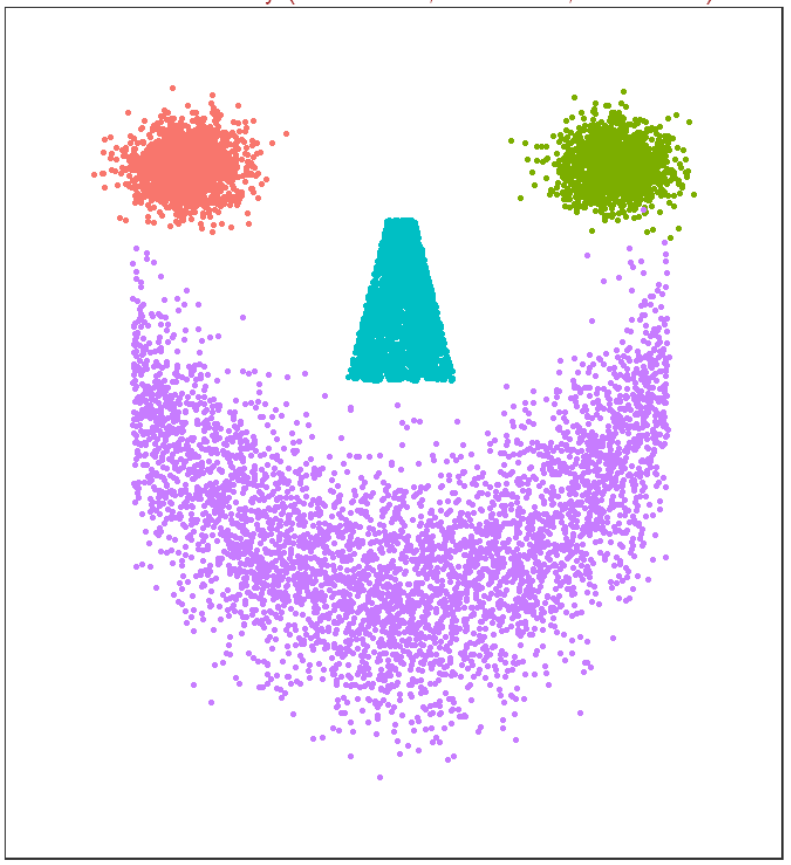


Smiley



Dataset

mlbench.smiley (n = 10000, sd1 = 0.1, sd2 = 0.3)



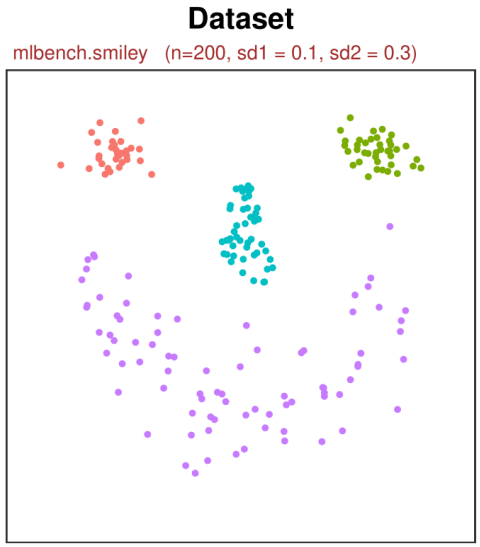
This dataset image contains 10,000 points, so we can se its true form. The next ones will contain a low number in order to have a better view of the decision boundary of each algorithm.

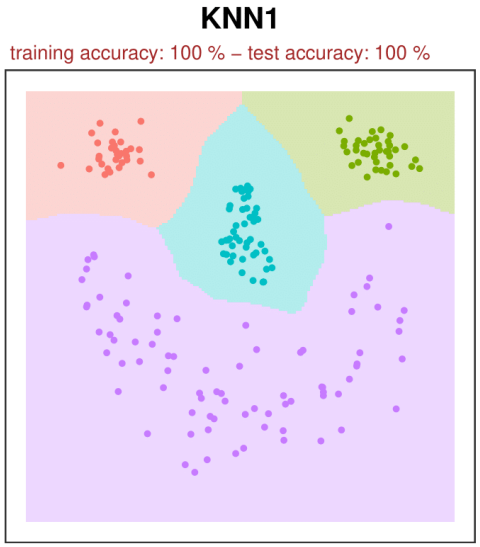
The Smiley

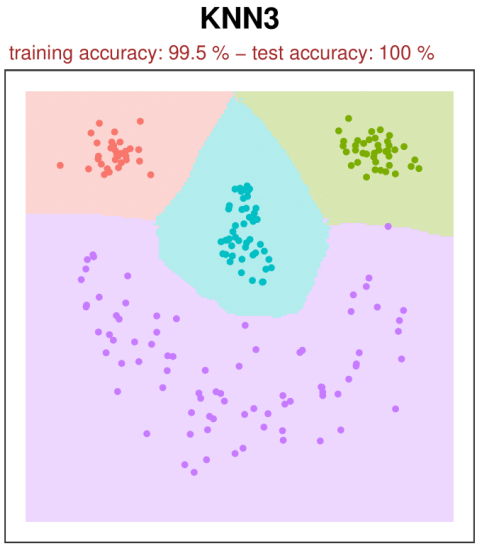
The smiley consists of 2 Gaussian eyes, a trapezoid nose and a parabula mouth (with vertical Gaussian noise).

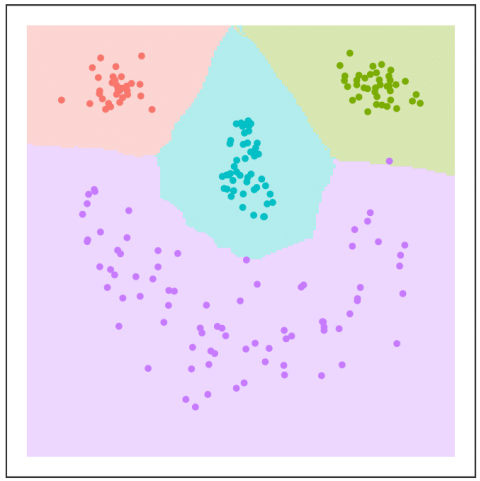
Usage:
mlbench.smiley(n, sd1, sd2)

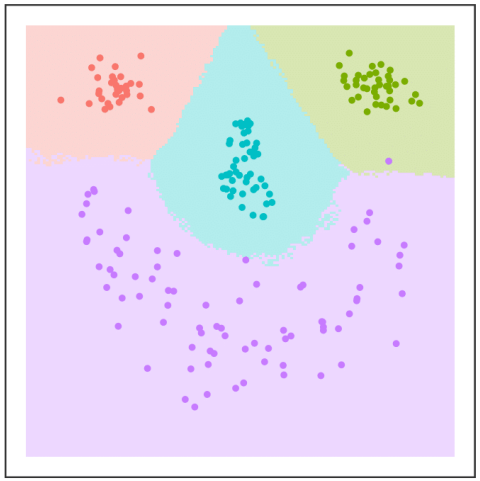
Smiley KNN

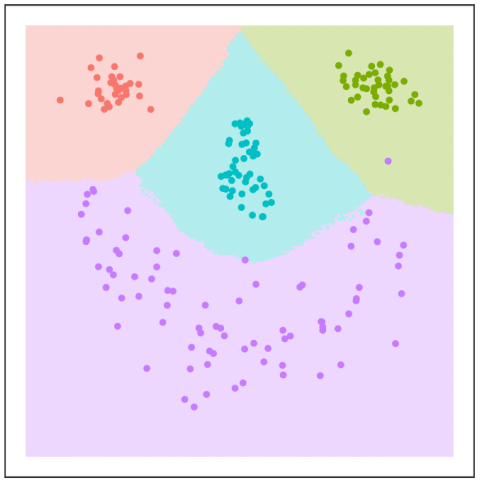

Dataset
mlbench.smiley (n=200, sd1 = 0.1, sd2 = 0.3)


KNN1
training accuracy: 100 % – test accuracy: 100 %

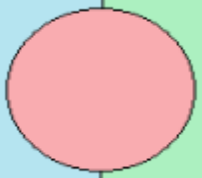

KNN3
training accuracy: 99.5 % – test accuracy: 100 %


KNN5
training accuracy: 99.5 % – test accuracy: 99.5 %

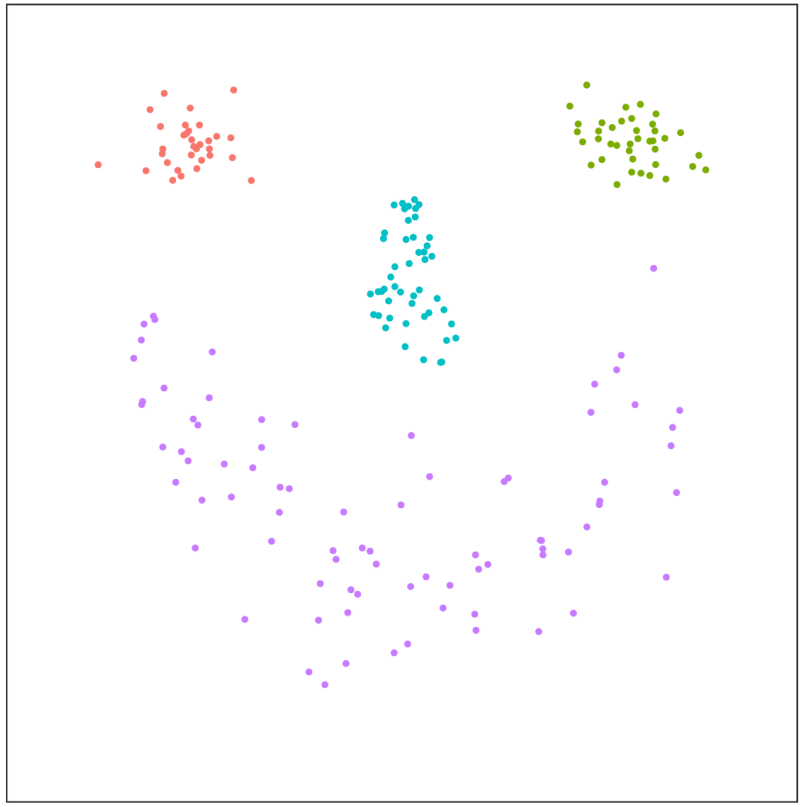

KNN10
training accuracy: 99.5 % – test accuracy: 99.5 %

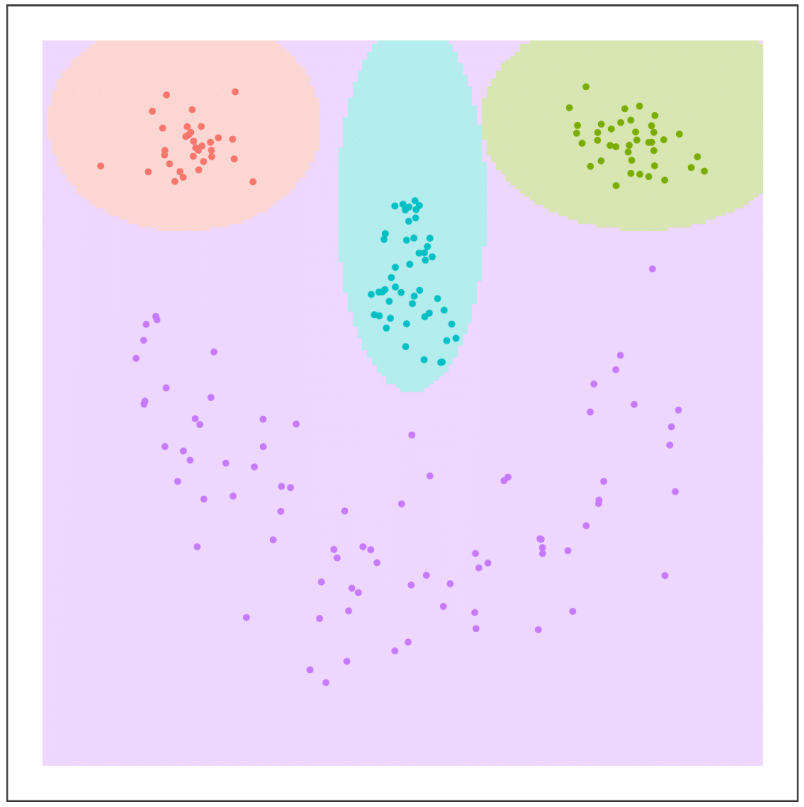

KNN30
training accuracy: 99 % – test accuracy: 99 %

Smiley

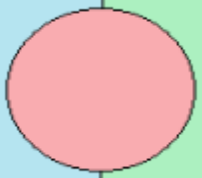


Naïve Bayes



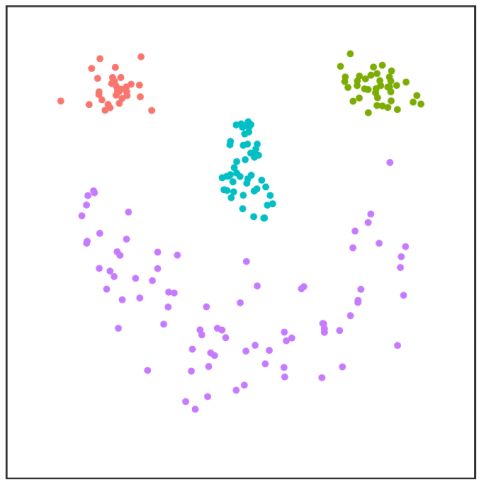


Smiley SVM



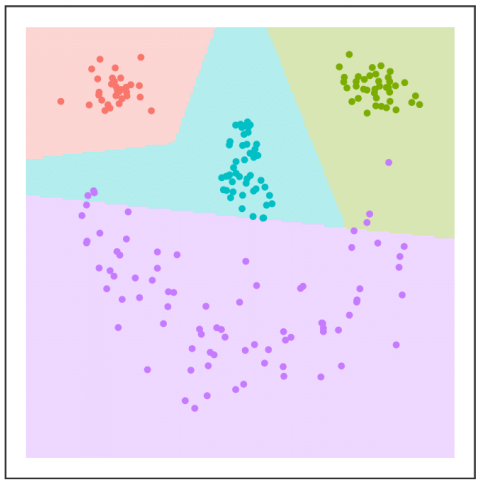
Dataset

mlbench.smiley (n=200, sd1 = 0.1, sd2 = 0.3)



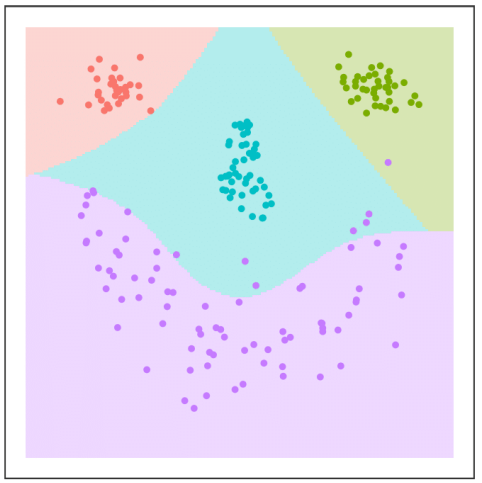
SVMlinear

training accuracy: 97 % – test accuracy: 97.5 %



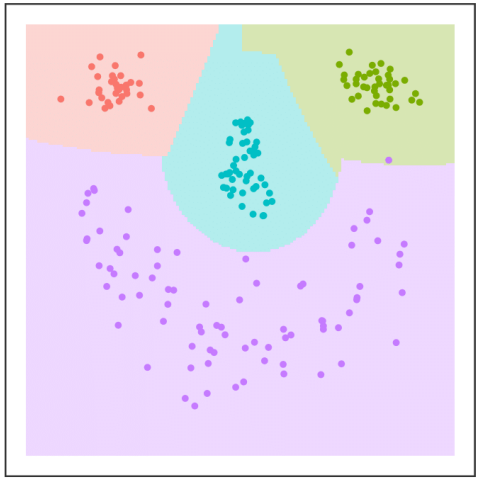
SVMpolynomial

training accuracy: 96 % – test accuracy: 97 %



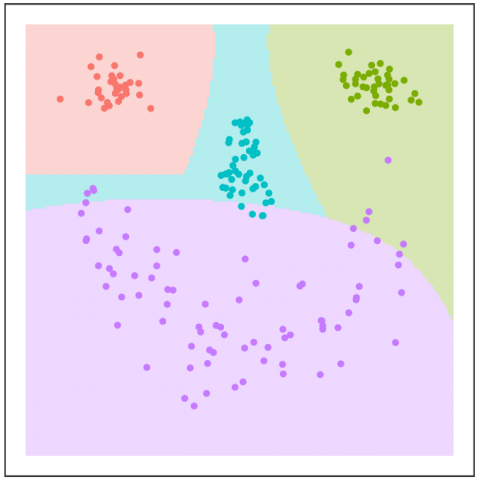
SVMradial

training accuracy: 99.5 % – test accuracy: 100 %

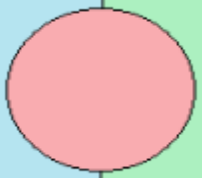


SVMsigmoid

training accuracy: 94.5 % – test accuracy: 93.5 %



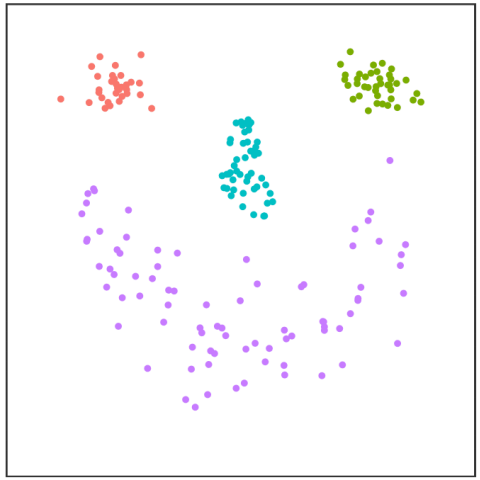
Smiley



Trees

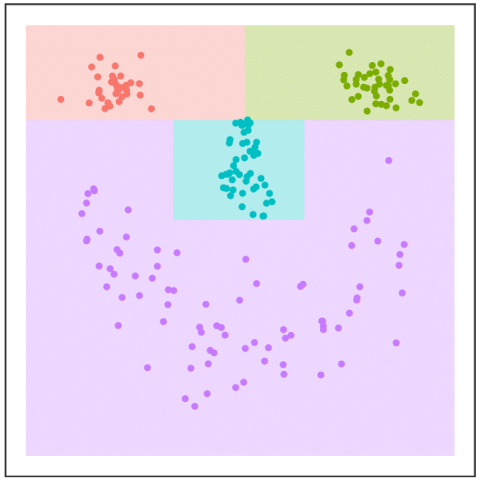
Dataset

mlbench.smiley (n=200, sd1 = 0.1, sd2 = 0.3)



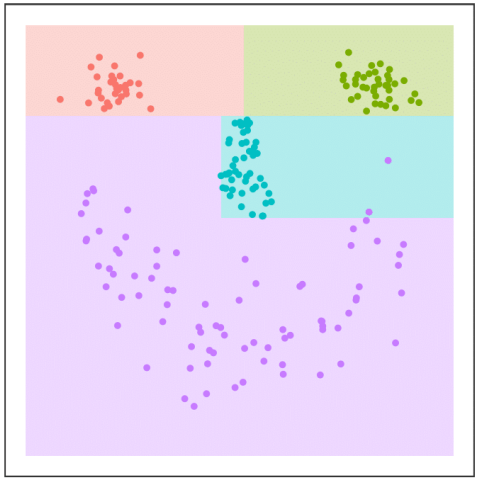
TreesC.50

training accuracy: 100 % – test accuracy: 99 %



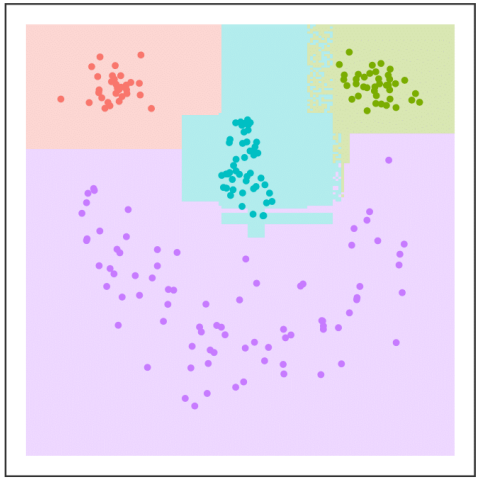
TreesCART

training accuracy: 98.5 % – test accuracy: 97 %



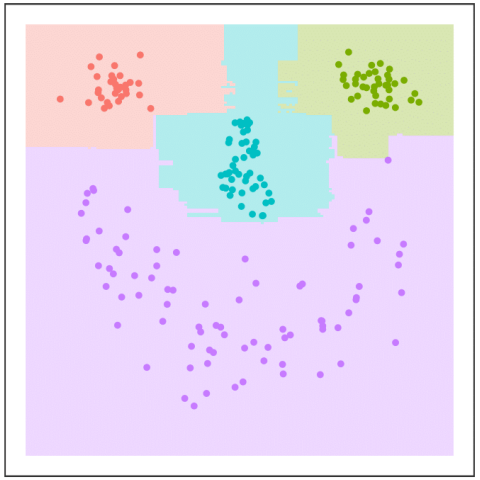
TreesRandom Forest 3 trees

training accuracy: 100 % – test accuracy: 98.5 %

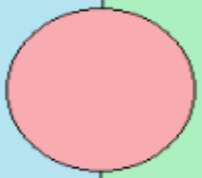


TreesRandom Forest 128 trees

training accuracy: 100 % – test accuracy: 99.5 %



Smiley

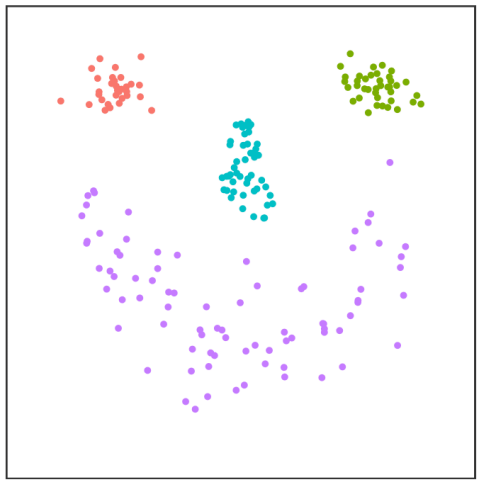


ANN

fixed iterations

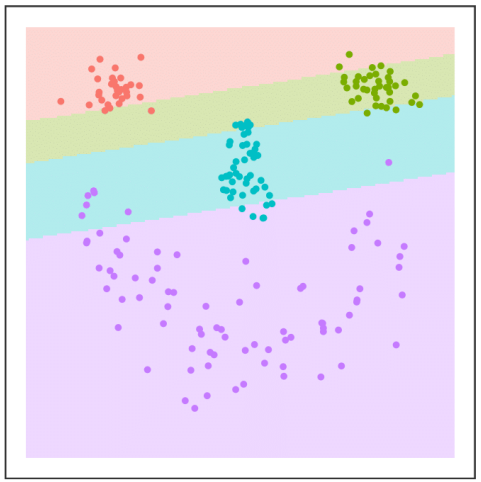
Dataset

mlbench.smiley(n=200, sd1 = 0.1, sd2 = 0.3)



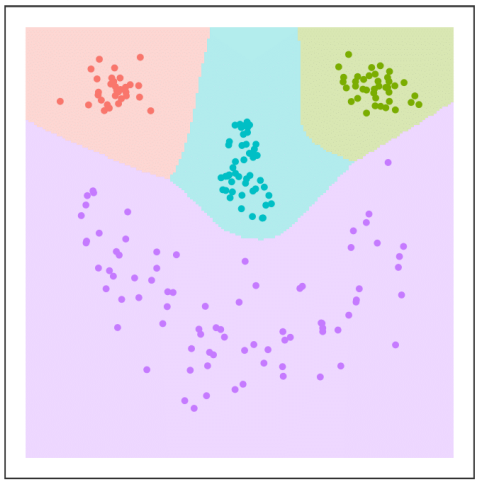
ANN; iters=100; neurons=1

training accuracy: 86 % – test accuracy: 89 %



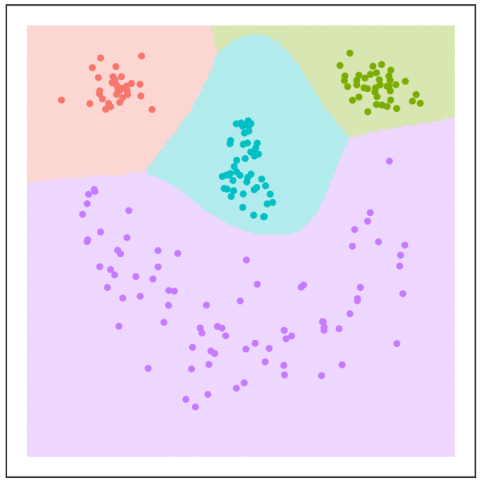
ANN; iters=100; neurons=10

training accuracy: 100 % – test accuracy: 100 %



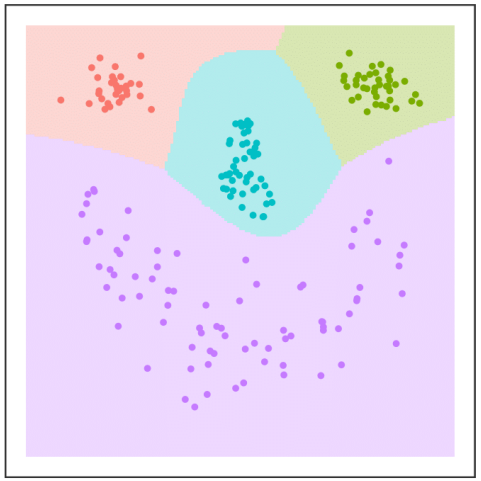
ANN; iters=100; neurons=100

training accuracy: 100 % – test accuracy: 100 %



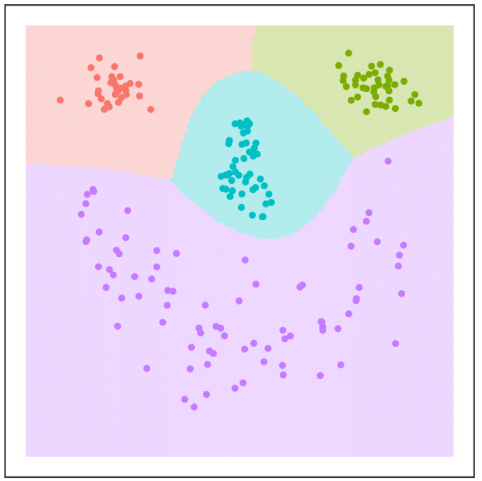
ANN; iters=100; neurons=500

training accuracy: 100 % – test accuracy: 100 %

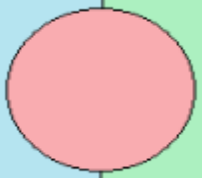


ANN; iters=100; neurons=1000

training accuracy: 100 % – test accuracy: 100 %



Smiley

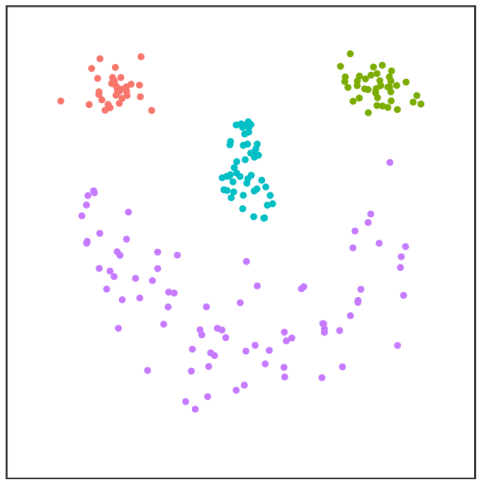


ANN

fixed neurons

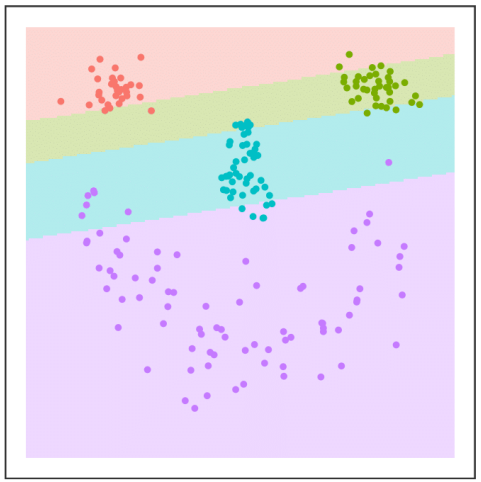
Dataset

mlbench.smiley(n=200, sd1 = 0.1, sd2 = 0.3)



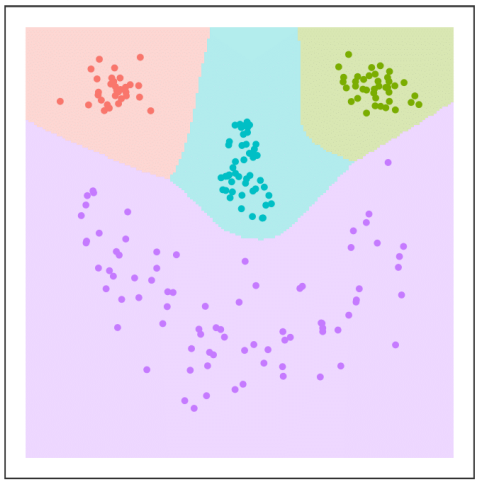
ANN; iters=100; neurons=1

training accuracy: 86 % – test accuracy: 89 %



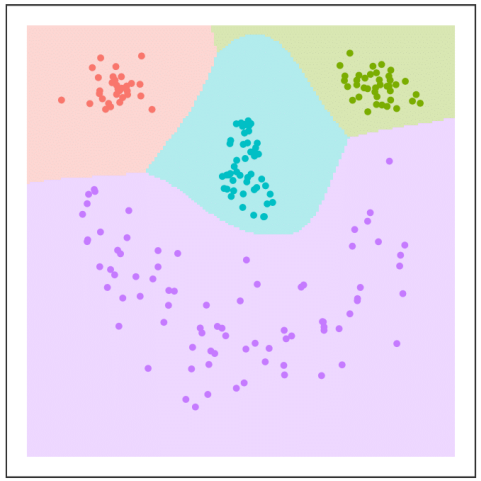
ANN; iters=100; neurons=10

training accuracy: 100 % – test accuracy: 100 %



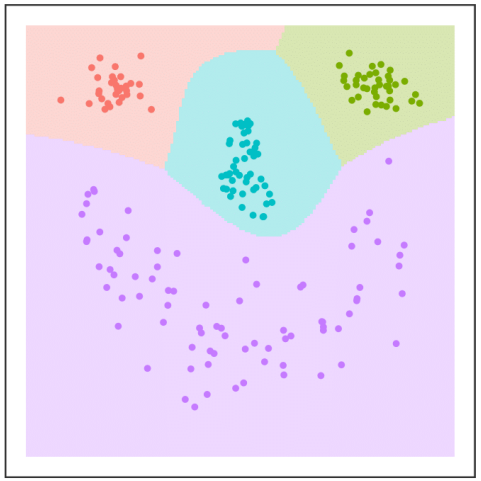
ANN; iters=100; neurons=100

training accuracy: 100 % – test accuracy: 100 %



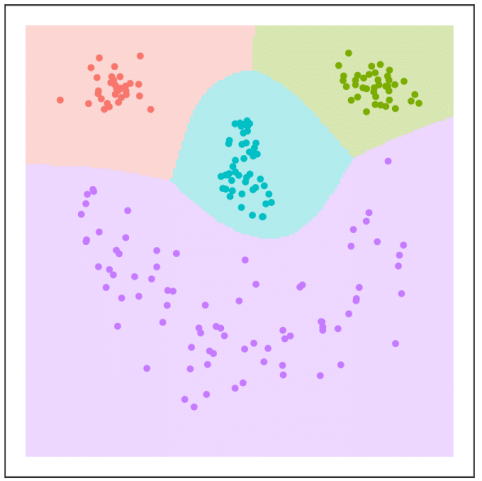
ANN; iters=100; neurons=500

training accuracy: 100 % – test accuracy: 100 %

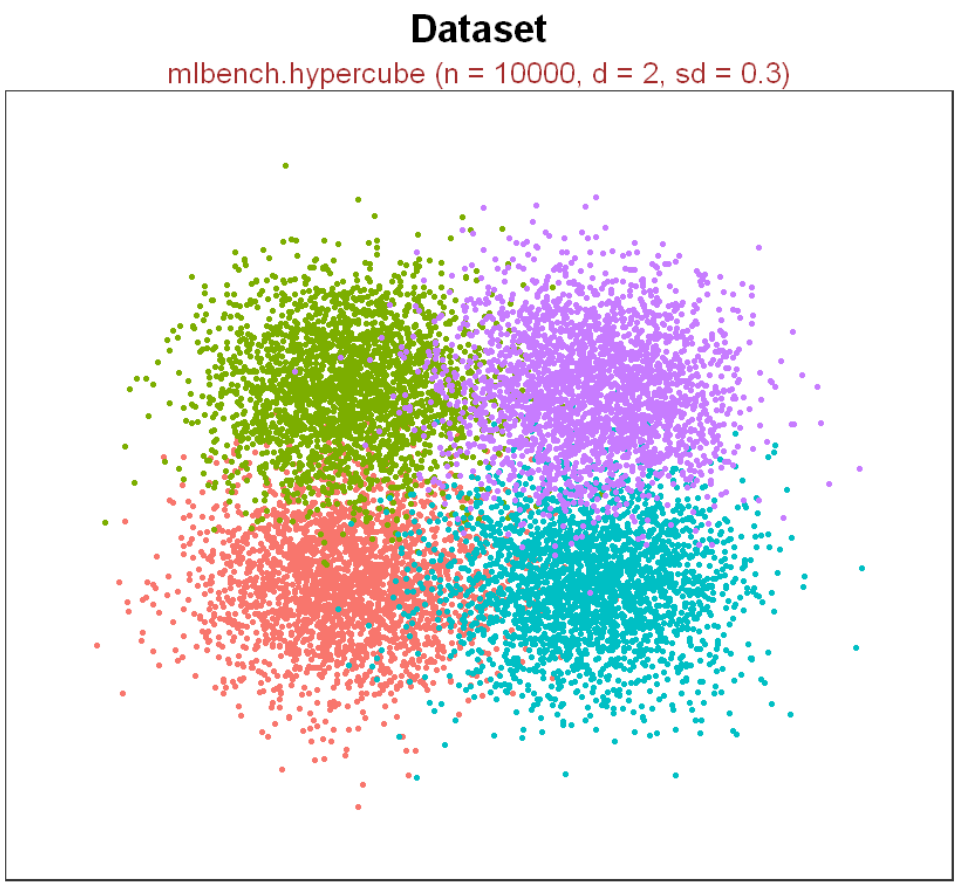


ANN; iters=100; neurons=1000

training accuracy: 100 % – test accuracy: 100 %



Hypercube



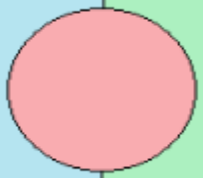
This dataset image contains 10,000 points, so we can se its true form. The next ones will contain a low number in order to have a better view of the decision boundary of each algorithm.

Corners of Hypercube

The created data are *d*-dimensional spherical Gaussians with standard deviation *sd* and means at the corners of a *d*-dimensional simplex. The number of classes is *d* + 1.

Usage:
mlbench.hypercube(n, d, sides, sd)

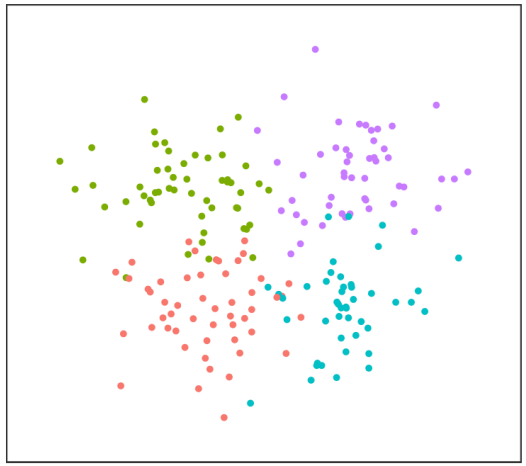
Hypercube



KNN

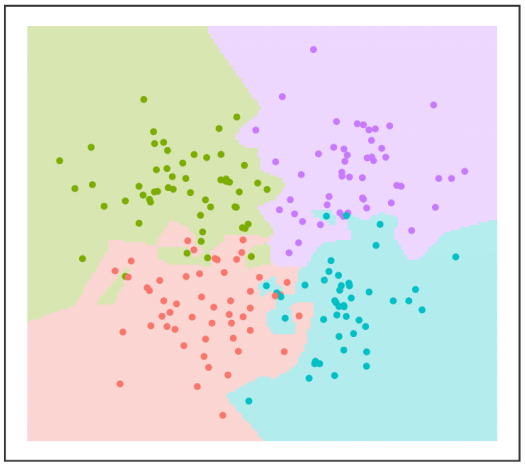
Dataset

mlbench.hypercube(n=200, d=2, sd=0.3)



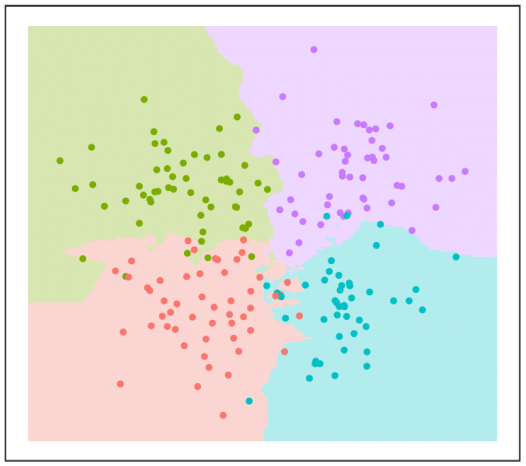
KNN1

training accuracy: 100 % – test accuracy: 86 %



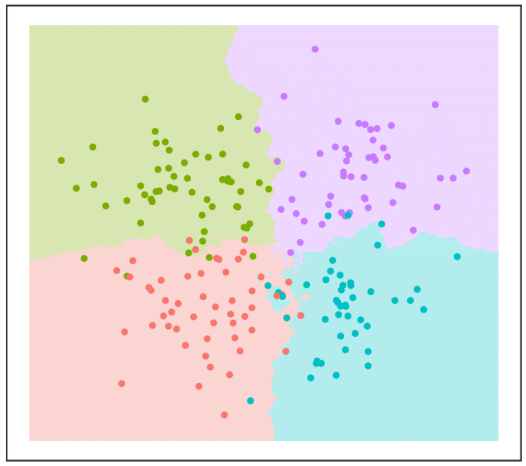
KNN3

training accuracy: 92.5 % – test accuracy: 88 %



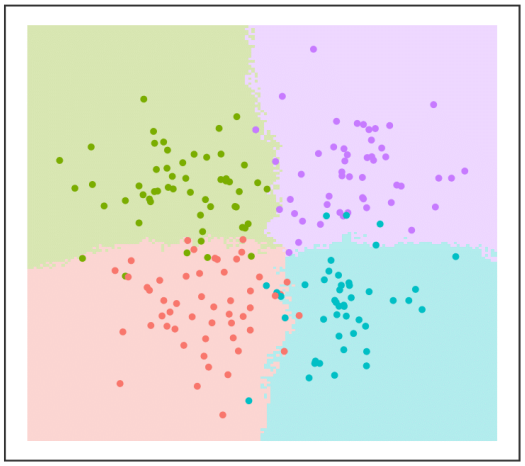
KNN5

training accuracy: 89.5 % – test accuracy: 87 %



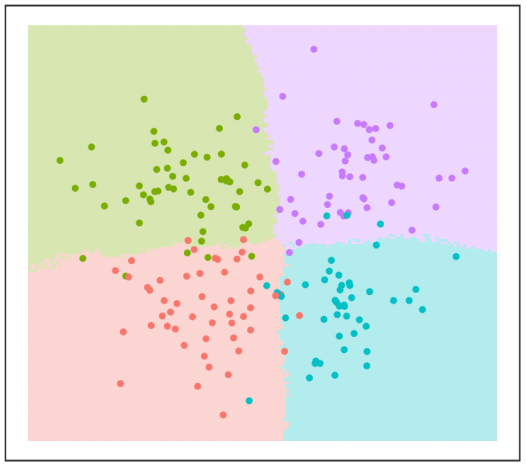
KNN10

training accuracy: 89 % – test accuracy: 86.5 %

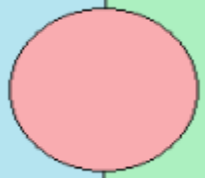


KNN30

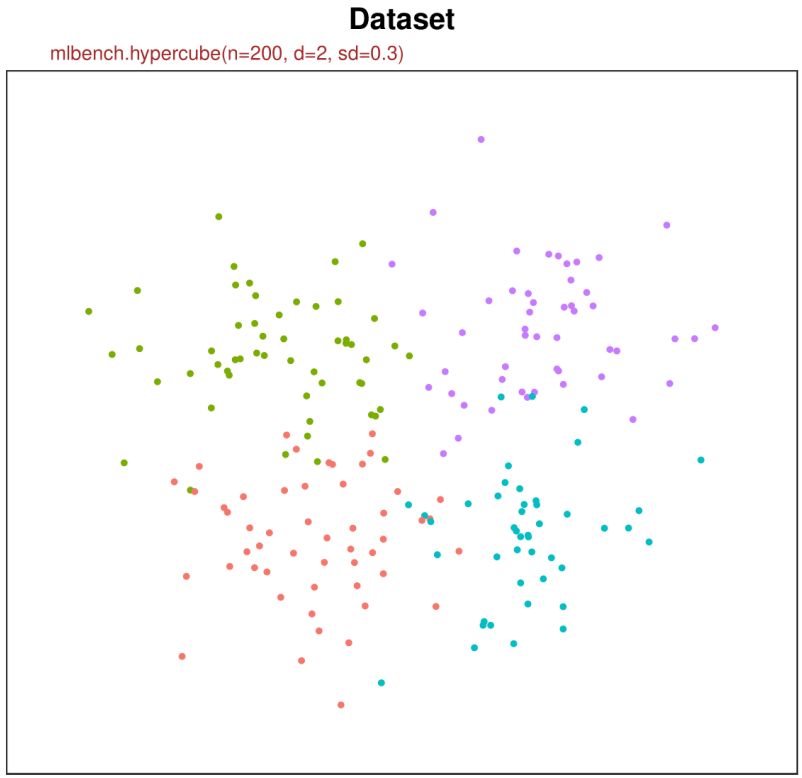
training accuracy: 90.5 % – test accuracy: 85.5 %

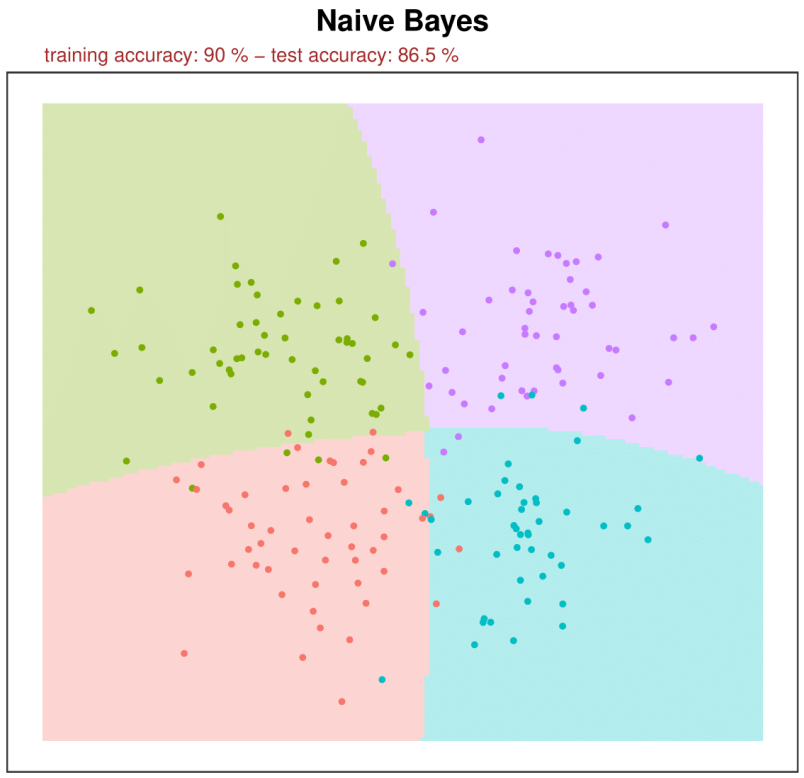


Hypercube

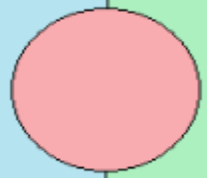


Naïve Bayes





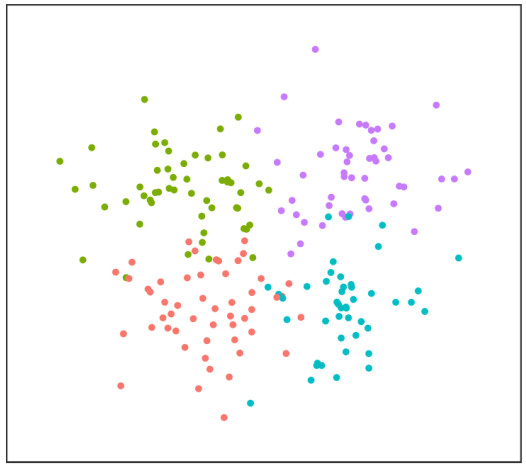
Hypercube



SVM

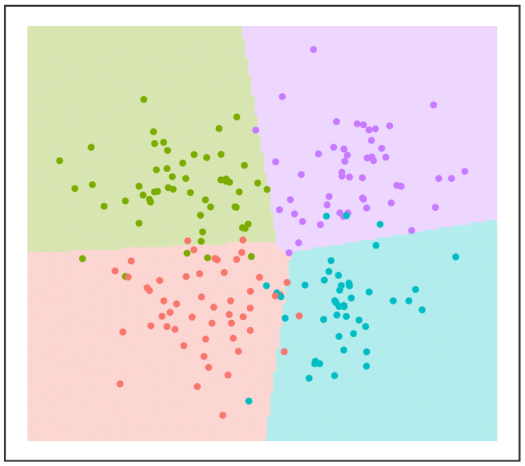
Dataset

mlbench.hypercube(n=200, d=2, sd=0.3)



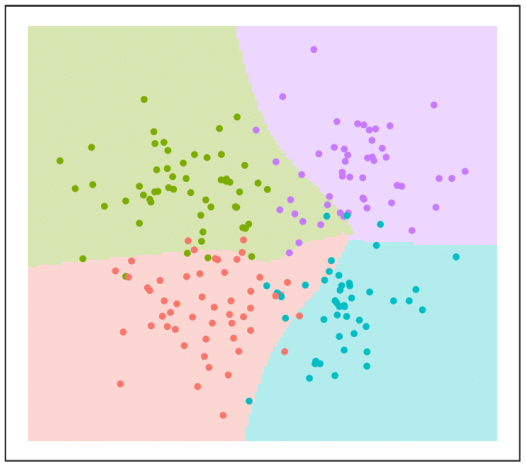
SVMlinear

training accuracy: 90.5 % – test accuracy: 87 %



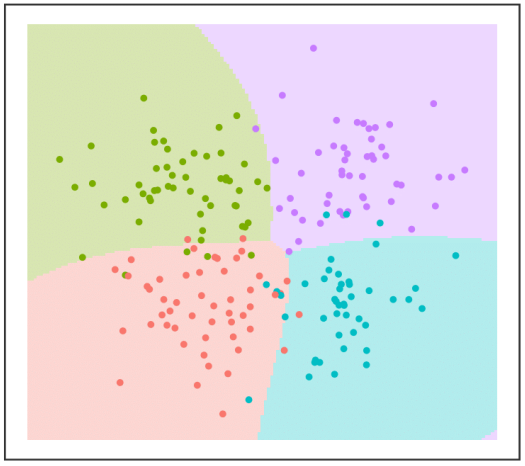
SVMpolynomial

training accuracy: 83 % – test accuracy: 81 %



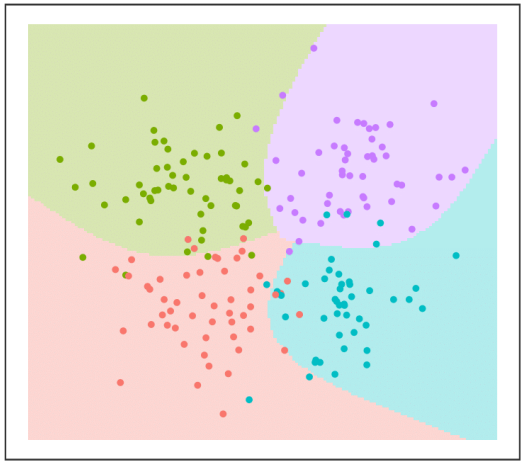
SVMradial

training accuracy: 92 % – test accuracy: 85.5 %

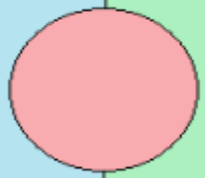


SVMsigmoid

training accuracy: 89 % – test accuracy: 86.5 %



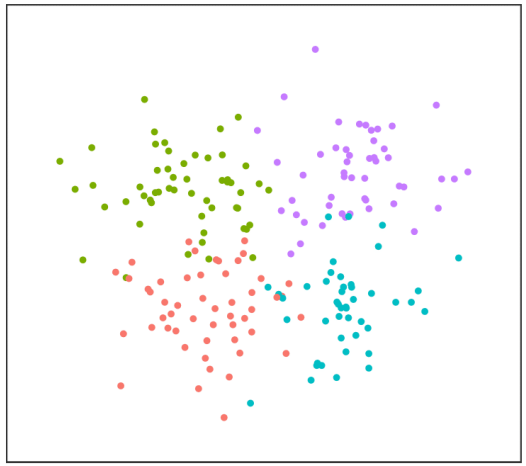
Hypercube



Trees

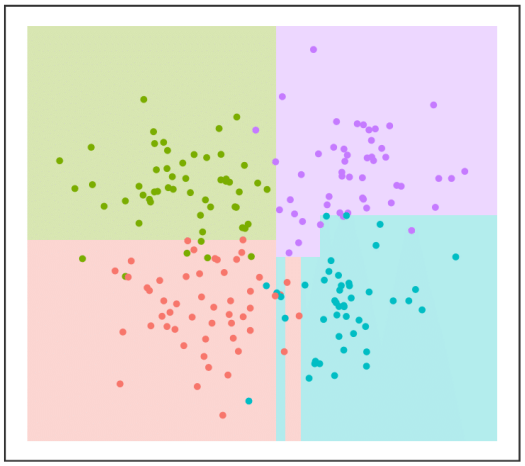
Dataset

mlbench.hypercube(n=200, d=2, sd=0.3)



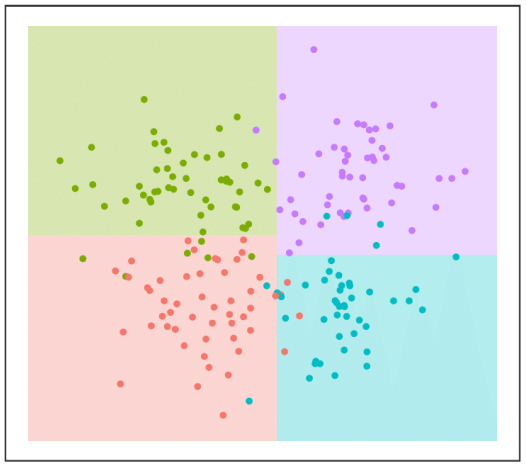
TreesC.50

training accuracy: 93.5 % – test accuracy: 83.5 %



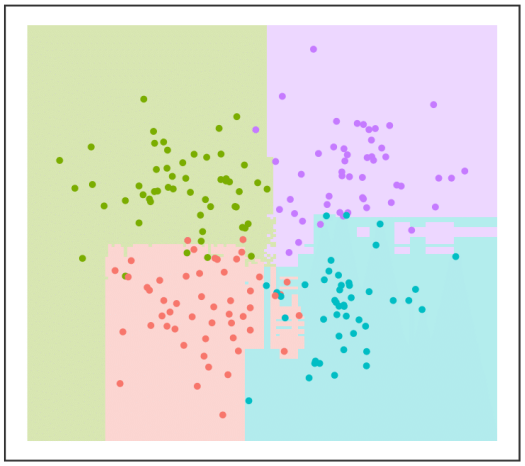
TreesCART

training accuracy: 91.5 % – test accuracy: 85 %



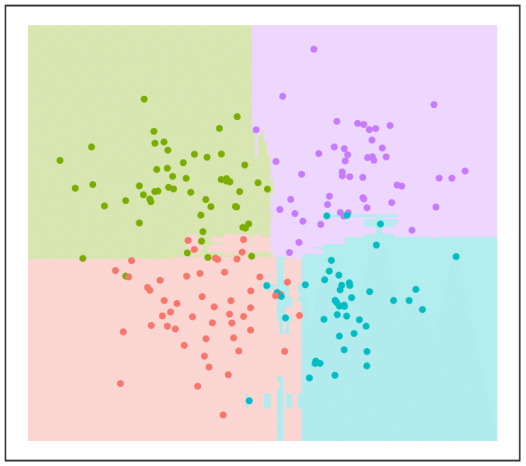
TreesRandom Forest 3 trees

training accuracy: 96 % – test accuracy: 85 %



TreesRandom Forest 128 trees

training accuracy: 100 % – test accuracy: 87 %



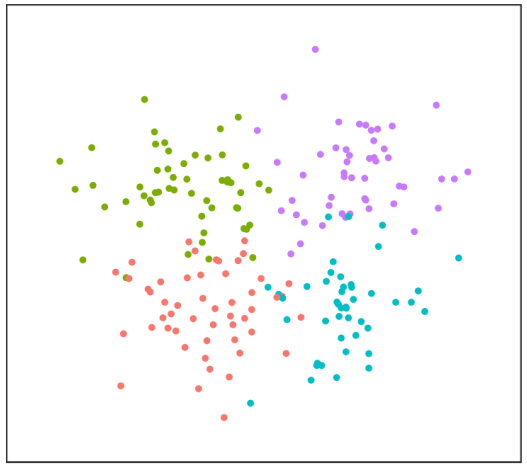
Hypercube

ANN

fixed iterations

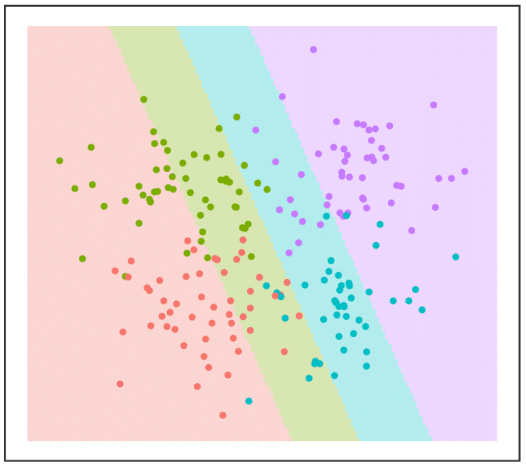
Dataset

mlbench.hypercube(n=200, d=2, sd=0.3)



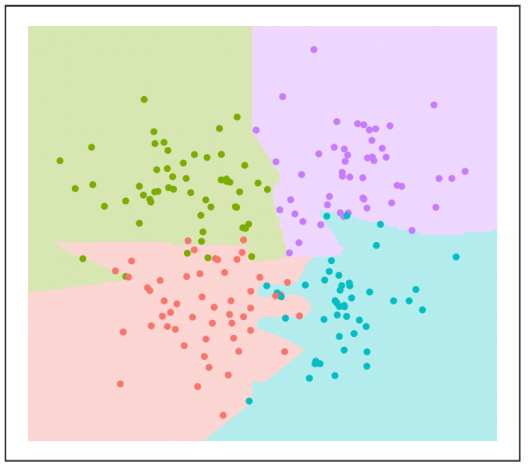
ANN; iters=100; neurons=1

training accuracy: 62.5 % – test accuracy: 61.5 %



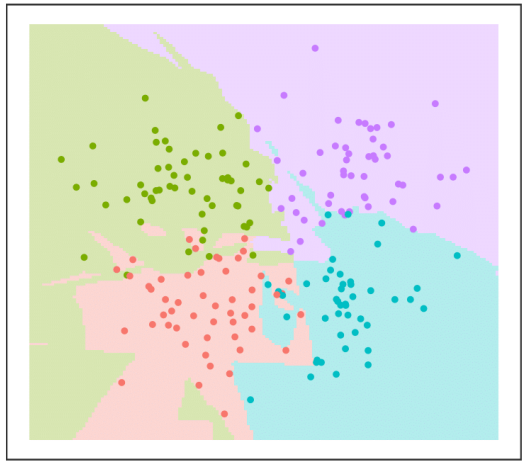
ANN; iters=100; neurons=10

training accuracy: 96 % – test accuracy: 85 %



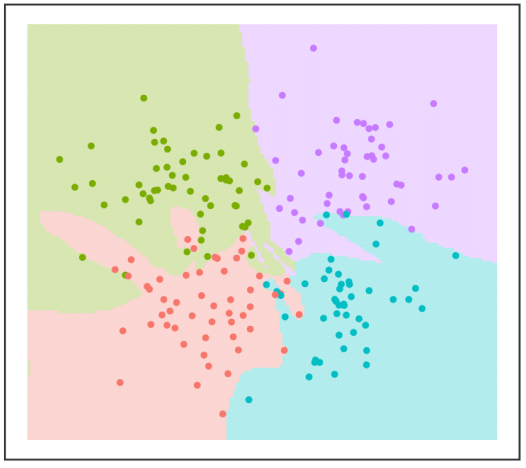
ANN; iters=100; neurons=100

training accuracy: 99.5 % – test accuracy: 83.5 %



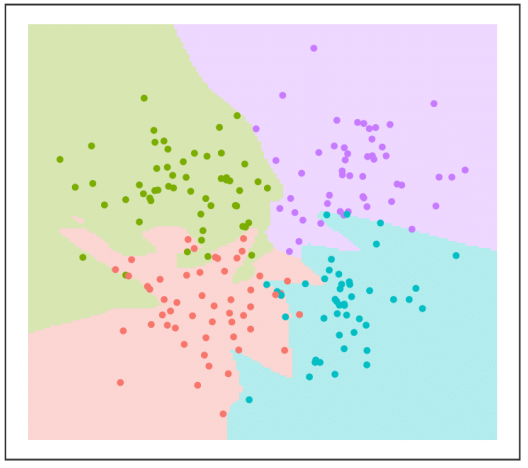
ANN; iters=100; neurons=500

training accuracy: 99 % – test accuracy: 81.5 %

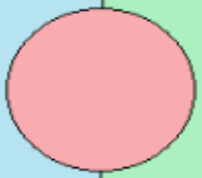


ANN; iters=100; neurons=1000

training accuracy: 96 % – test accuracy: 84 %



Hypercube

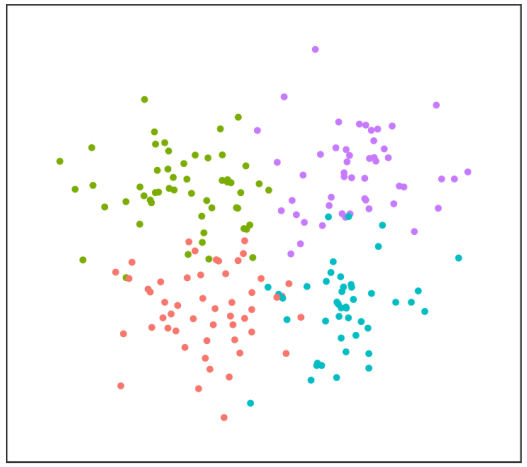


ANN

fixed neurons

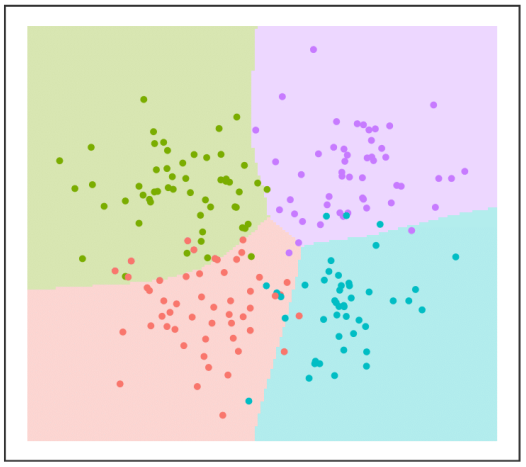
Dataset

mlbench.hypercube(n=200, d=2, sd=0.3)



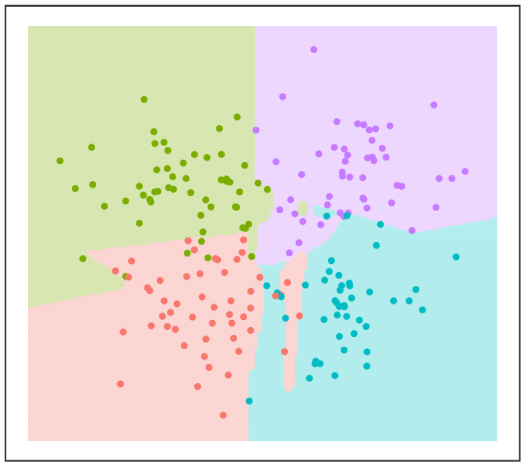
ANN; neurons=10; iters=10

training accuracy: 89.5 % – test accuracy: 84.5 %



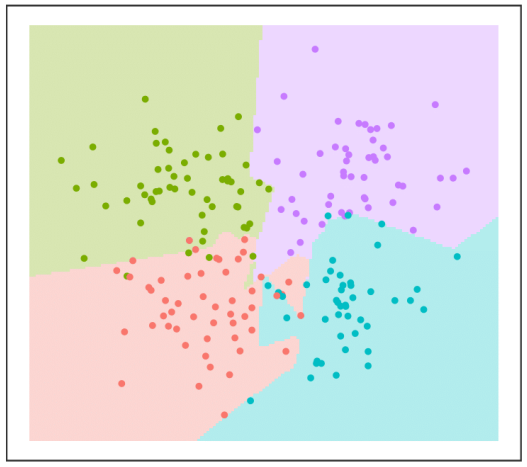
ANN; neurons=10; iters=100

training accuracy: 96.5 % – test accuracy: 84 %



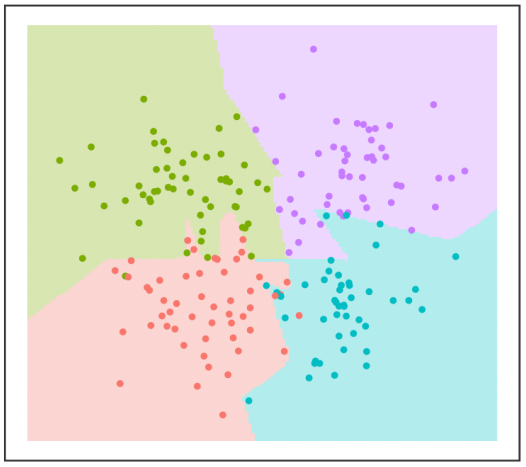
ANN; neurons=10; iters=500

training accuracy: 96 % – test accuracy: 85 %



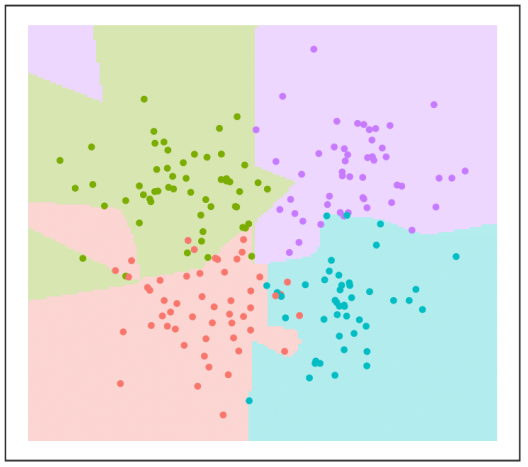
ANN; neurons=10; iters=1000

training accuracy: 96.5 % – test accuracy: 84.5 %

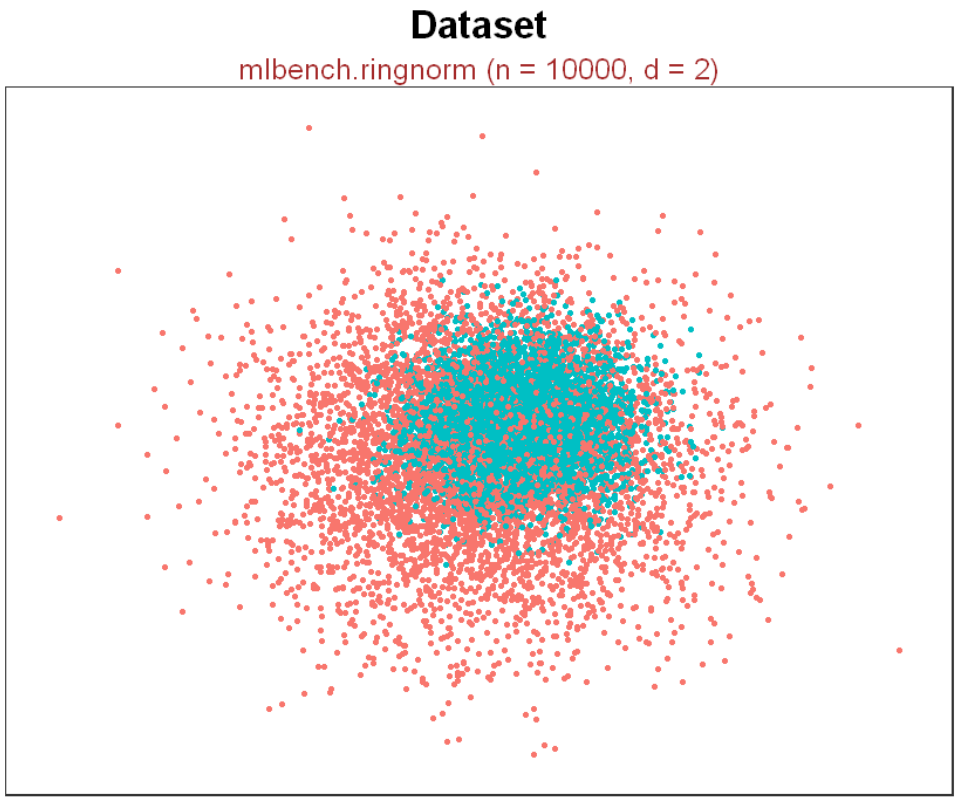


ANN; neurons=10; iters=10000

training accuracy: 95.5 % – test accuracy: 84.5 %



Ringnorm



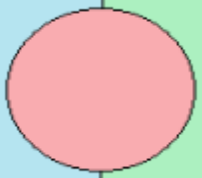
This dataset image contains 10,000 points, so we can se its true form. The next ones will contain a low number in order to have a better view of the decision boundary of each algorithm.

Ringnorm Benchmark Problem

The inputs of the ringnorm problem are points from two Gaussian distributions. Class 1 is multivariate normal with mean 0 and covariance 4 times the identity matrix. Class 2 has unit covariance and mean $(a, a, \dots, a), a = d^{-0.5}$.

Usage:
mlbench.ringnorm(n, d)

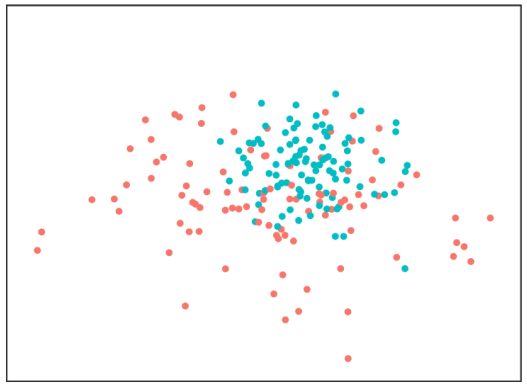
Ringnorm



KNN

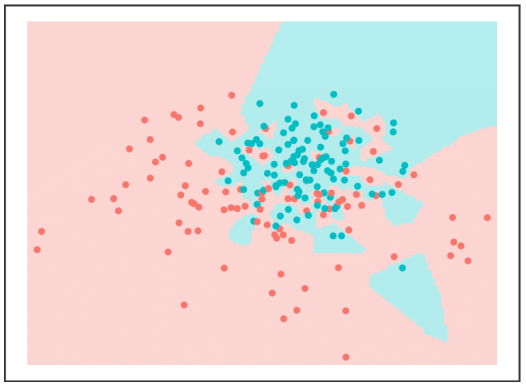
Dataset

mlbench.ringnorm (n=200, d=2)



KNN1

training accuracy: 100 % – test accuracy: 66.5 %

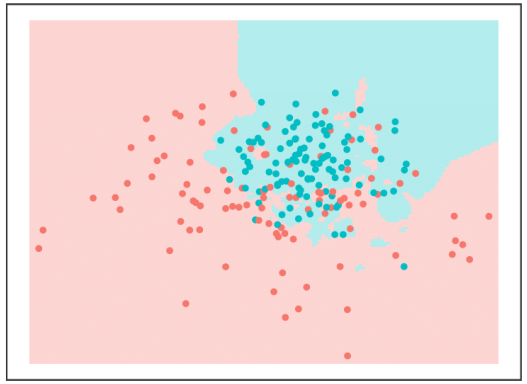


KNN3

training accuracy: 81.5 % – test accuracy: 71.5 %

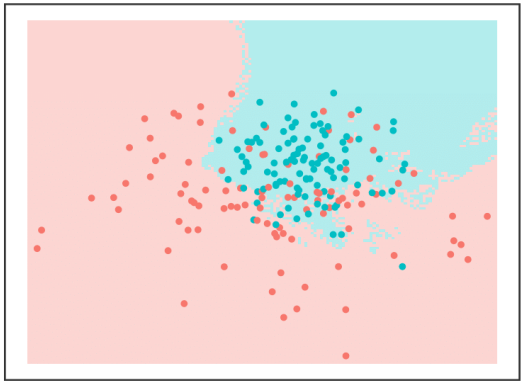
KNN5

training accuracy: 79 % – test accuracy: 72 %



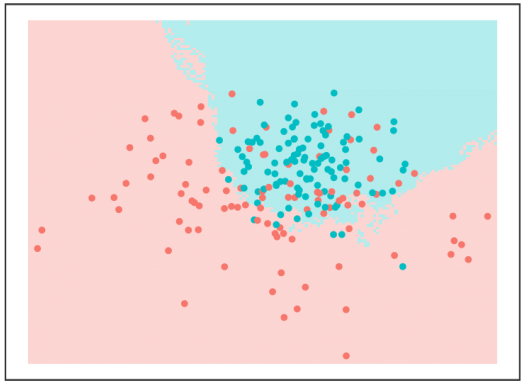
KNN10

training accuracy: 80 % – test accuracy: 70 %

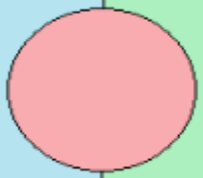


KNN30

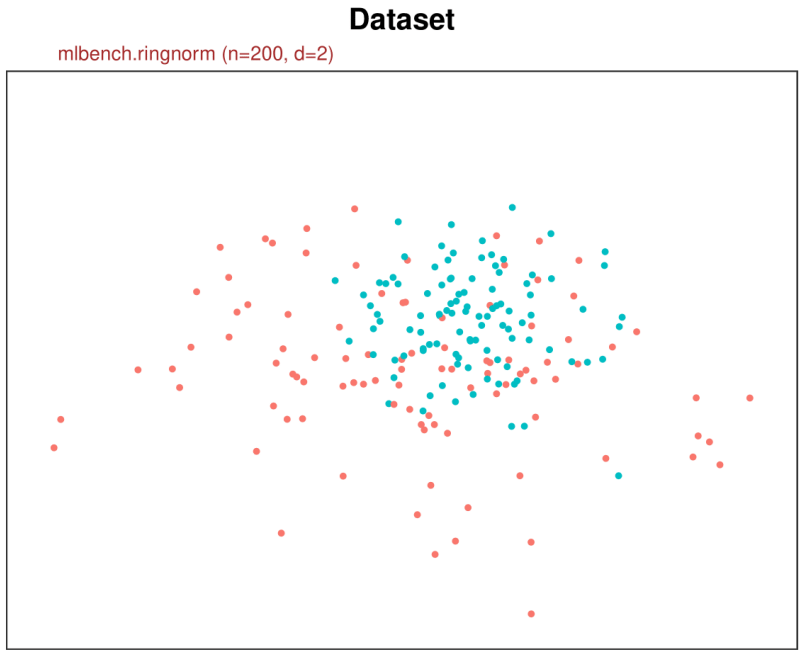
training accuracy: 77.5 % – test accuracy: 72.5 %

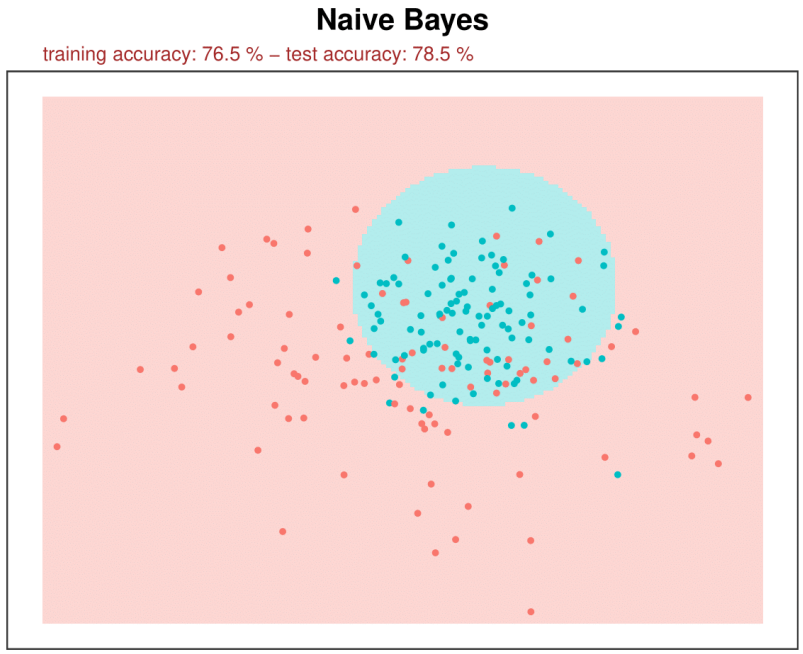


Ringnorm

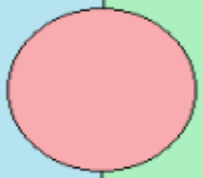


Naïve Bayes





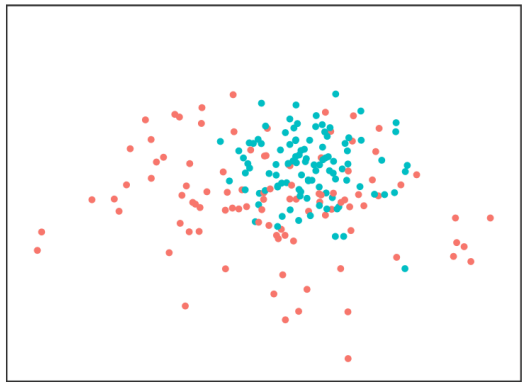
Ringnorm



SVM

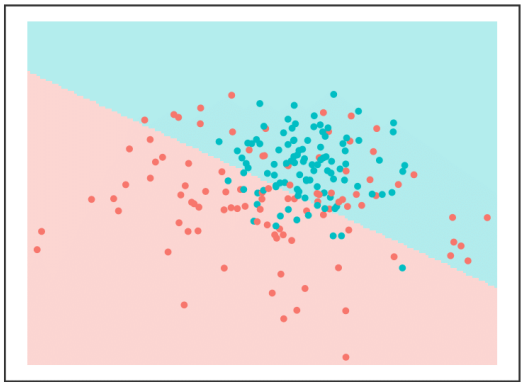
Dataset

mlbench.ringnorm (n=200, d=2)



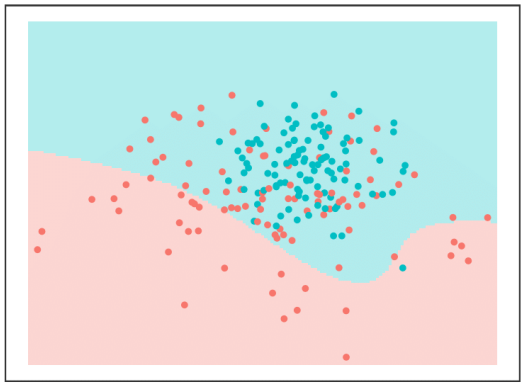
SVMlinear

training accuracy: 71 % – test accuracy: 68.5 %



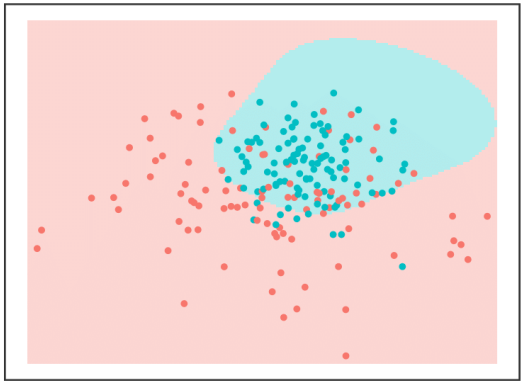
SVMpolynomial

training accuracy: 65 % – test accuracy: 62.5 %



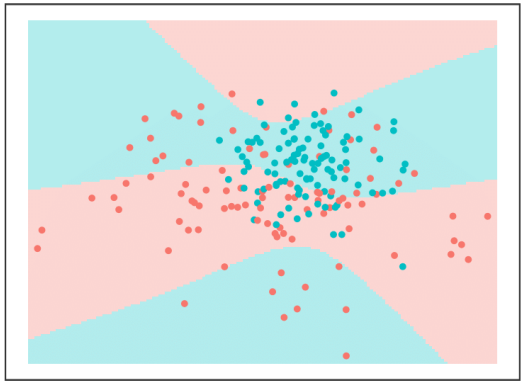
SVMradial

training accuracy: 76 % – test accuracy: 76 %

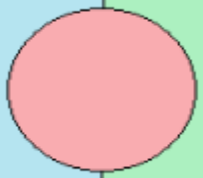


SVMsigmoid

training accuracy: 62.5 % – test accuracy: 61.5 %



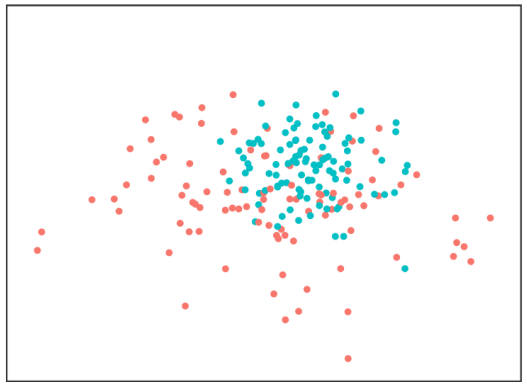
Ringnorm



Trees

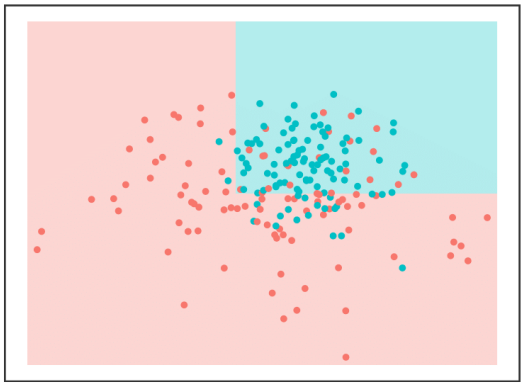
Dataset

mlbench.ringnorm (n=200, d=2)



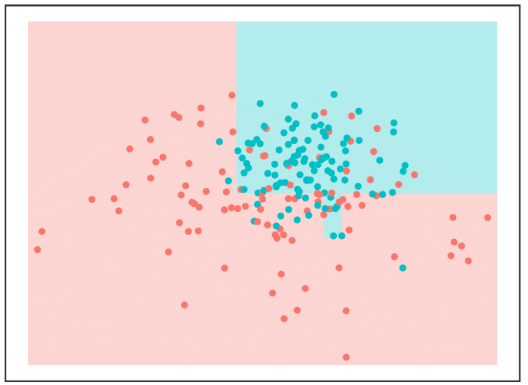
TreesC.50

training accuracy: 79 % – test accuracy: 71 %



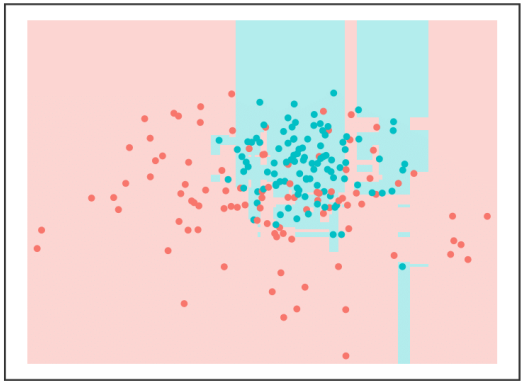
TreesCART

training accuracy: 81 % – test accuracy: 73 %



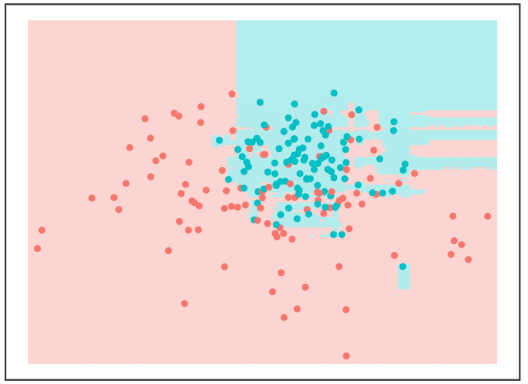
TreesRandom Forest 3 trees

training accuracy: 96.5 % – test accuracy: 66 %

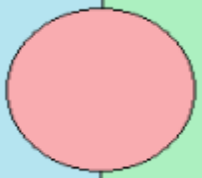


TreesRandom Forest 128 trees

training accuracy: 100 % – test accuracy: 68.5 %



Ringnorm

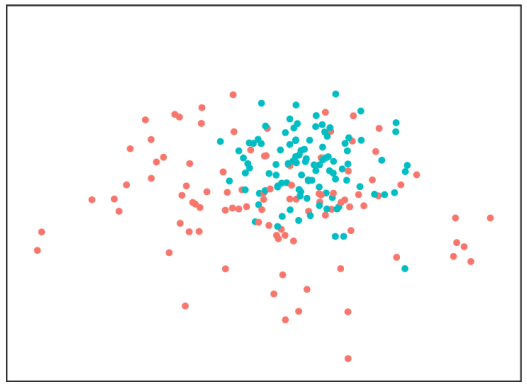


ANN

fixed iterations

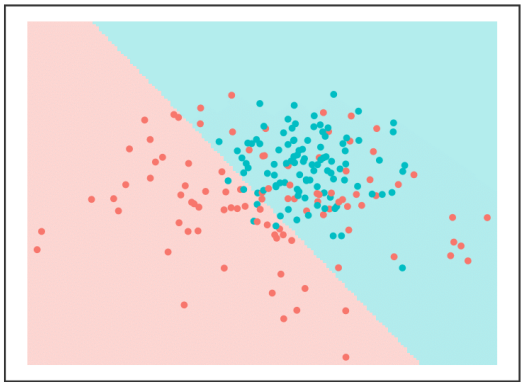
Dataset

mlbench.ringnorm(n=200, d=2)



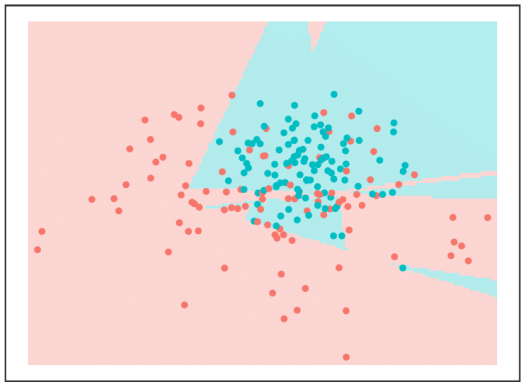
ANN; iters=100; neurons=1

training accuracy: 73.5 % – test accuracy: 65.5 %



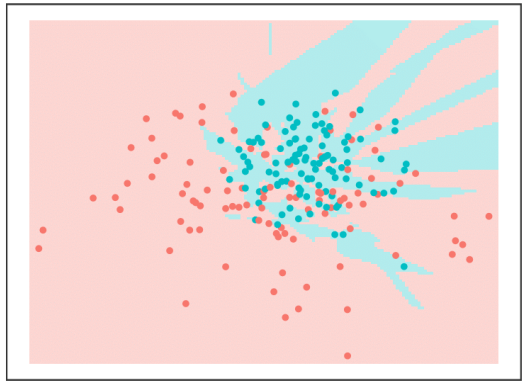
ANN; iters=100; neurons=10

training accuracy: 85.5 % – test accuracy: 73 %



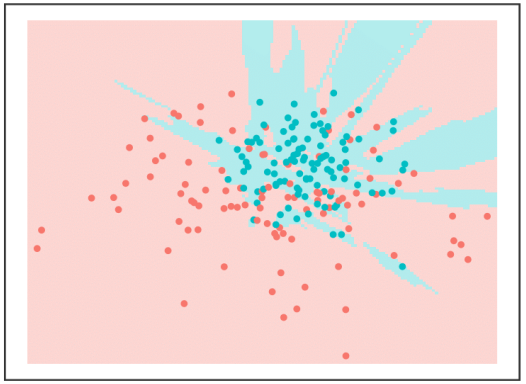
ANN; iters=100; neurons=100

training accuracy: 99 % – test accuracy: 67.5 %



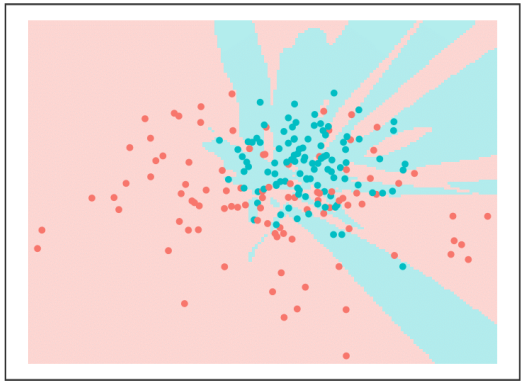
ANN; iters=100; neurons=500

training accuracy: 100 % – test accuracy: 71 %

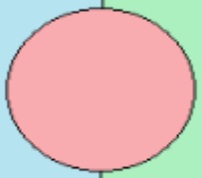


ANN; iters=100; neurons=1000

training accuracy: 100 % – test accuracy: 69 %



Ringnorm

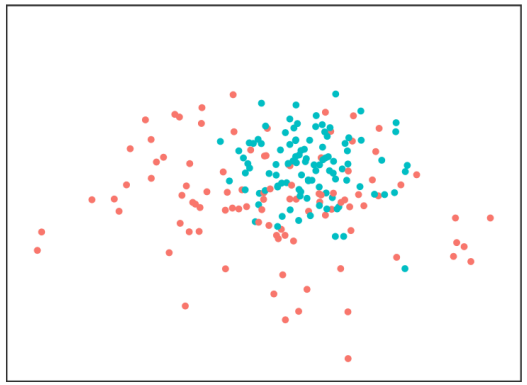


ANN

fixed neurons

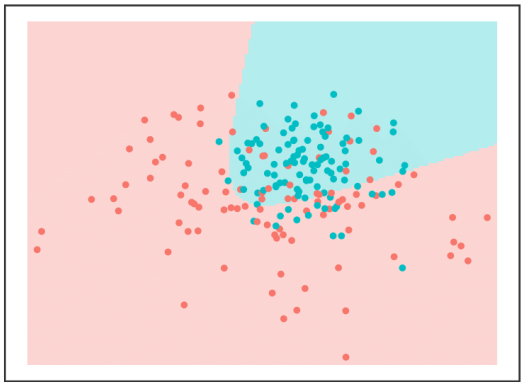
Dataset

mlbench.ringnorm(n=200, d=2)



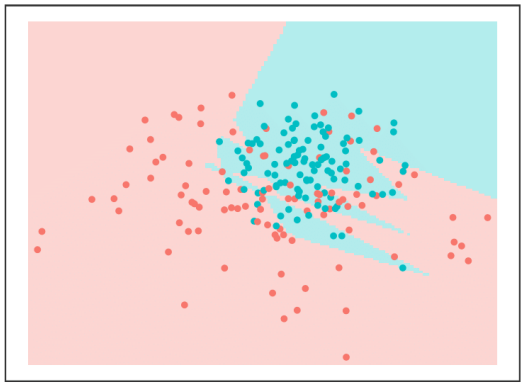
ANN; neurons=10; iters=10

training accuracy: 77.5 % – test accuracy: 72.5 %



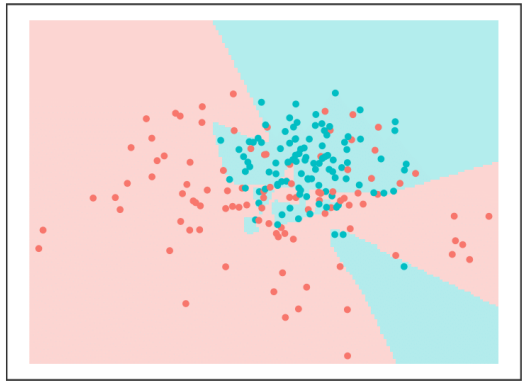
ANN; neurons=10; iters=100

training accuracy: 85.5 % – test accuracy: 71 %



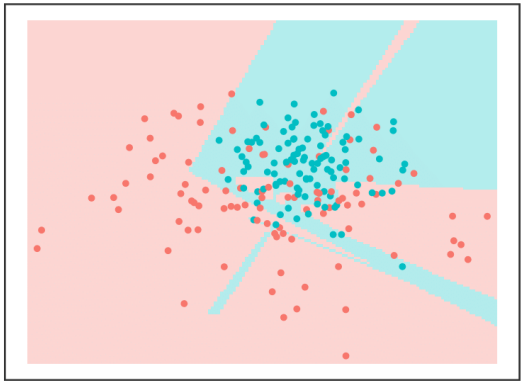
ANN; neurons=10; iters=500

training accuracy: 87.5 % – test accuracy: 72 %



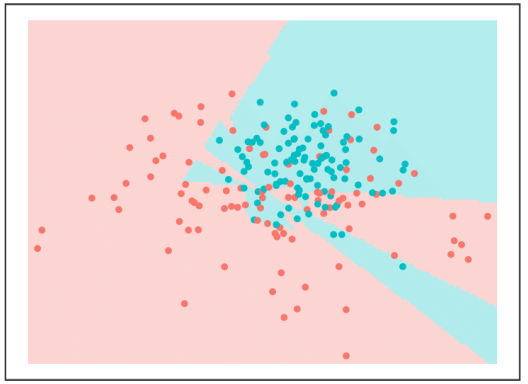
ANN; neurons=10; iters=1000

training accuracy: 83.5 % – test accuracy: 69 %

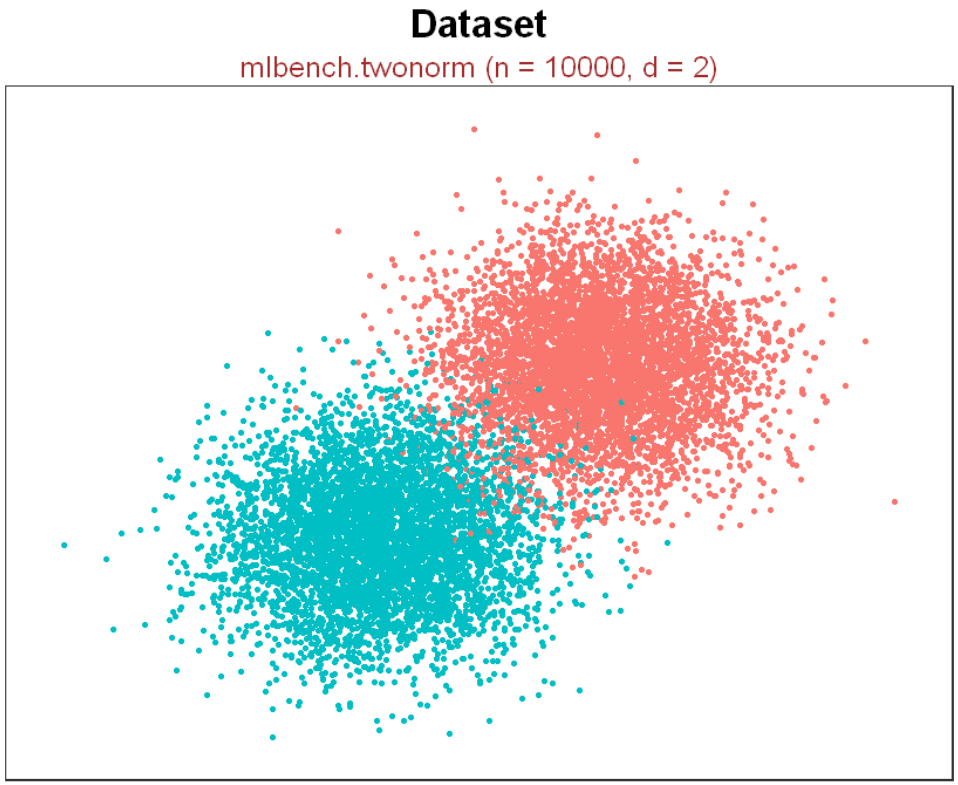


ANN; neurons=10; iters=10000

training accuracy: 87.5 % – test accuracy: 70.5 %



Twonorm



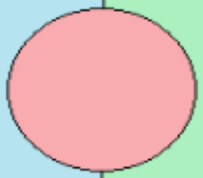
This dataset image contains 10,000 points, so we can se its true form. The next ones will contain a low number in order to have a better view of the decision boundary of each algorithm.

Twonorm Benchmark Problem

The inputs of the twonorm problem are points from two Gaussian distributions with unit covariance matrix. Class 1 is multivariate normal with mean $(a, a, \dots, a)$ and class 2 with mean at $(-a, -a, \dots, -a), a = 2/d^{0.5}$.

Usage:
mlbench. twonorm(n, d)

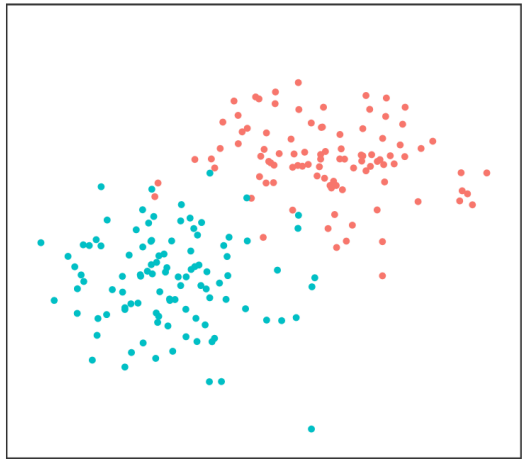
Twonorm



KNN

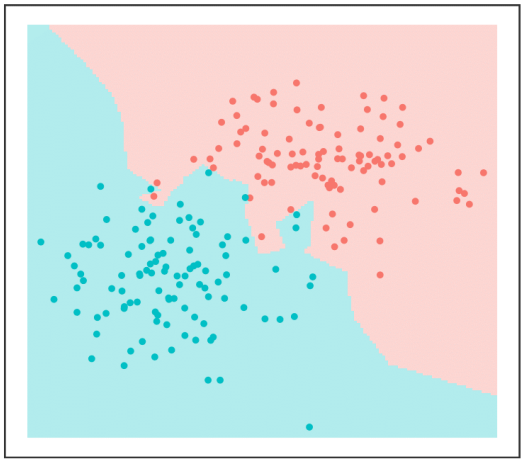
Dataset

mlbench.twonorm (n=200, d=2)



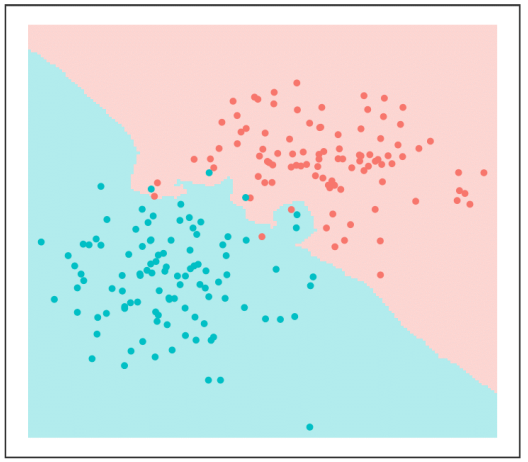
KNN1

training accuracy: 100 % – test accuracy: 96 %



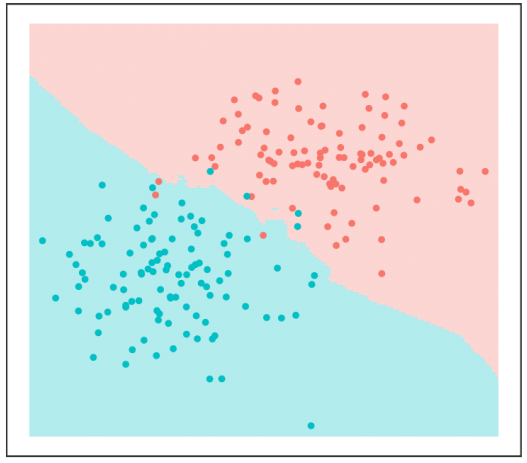
KNN3

training accuracy: 97.5 % – test accuracy: 96.5 %



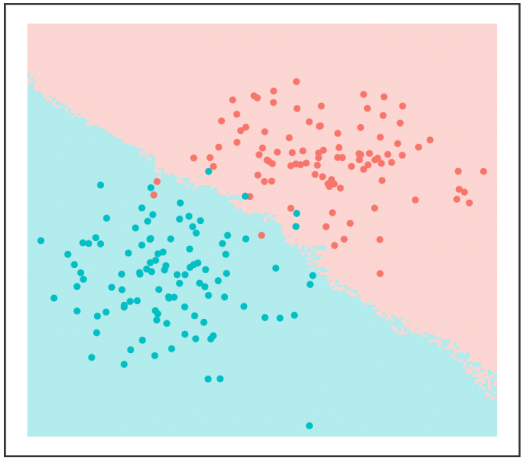
KNN5

training accuracy: 96.5 % – test accuracy: 97.5 %



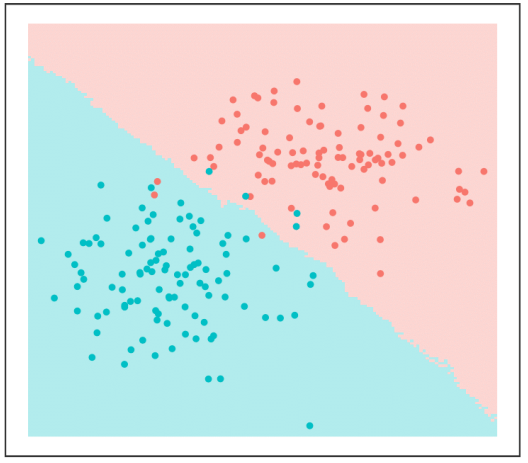
KNN10

training accuracy: 96.5 % – test accuracy: 97 %

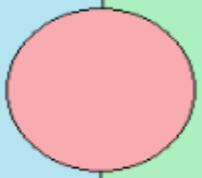


KNN30

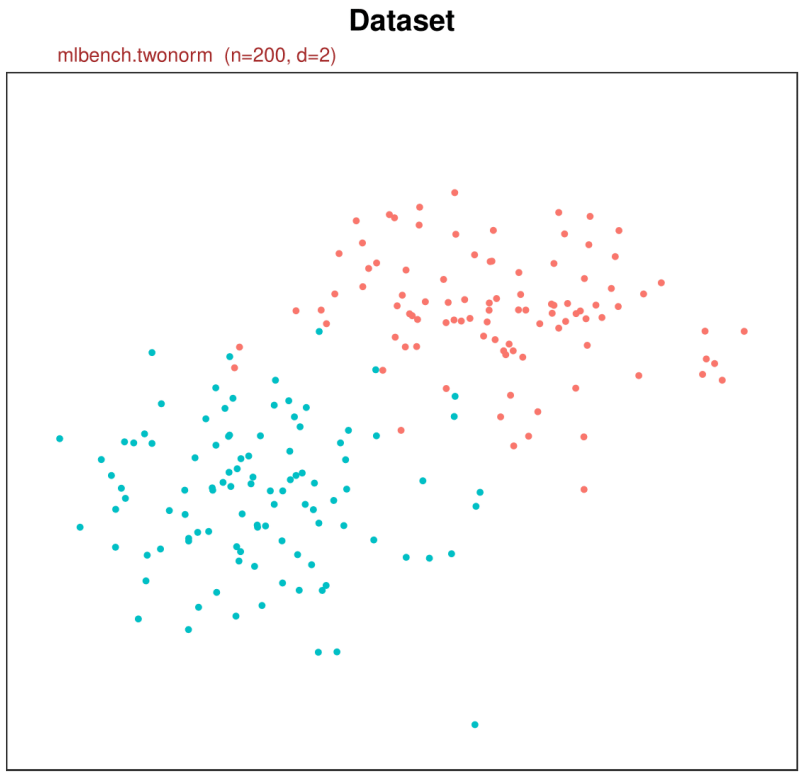
training accuracy: 96.5 % – test accuracy: 97 %

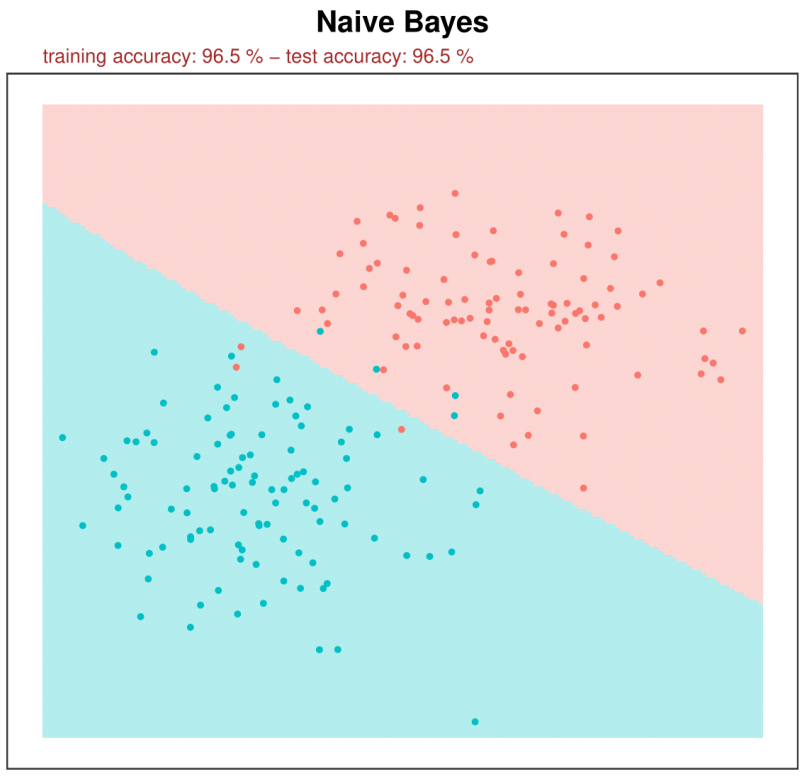


Twonorm

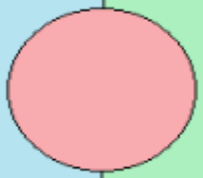


Naïve Bayes





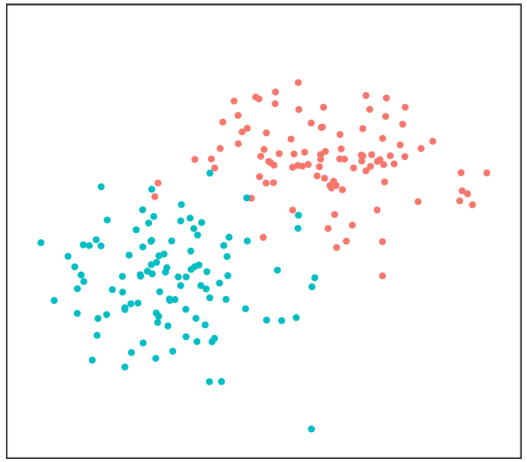
Twonorm



SVM

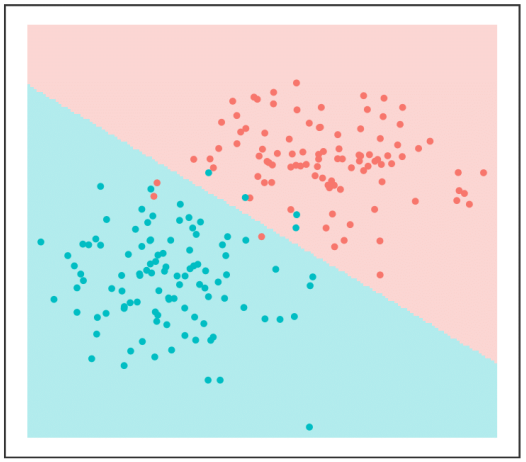
Dataset

mlbench.twonorm (n=200, d=2)



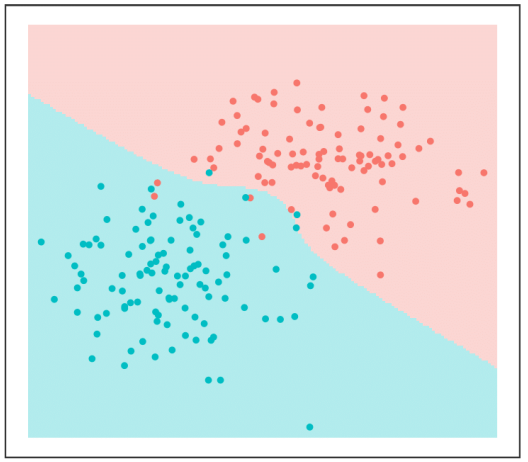
SVMlinear

training accuracy: 96.5 % – test accuracy: 96.5 %



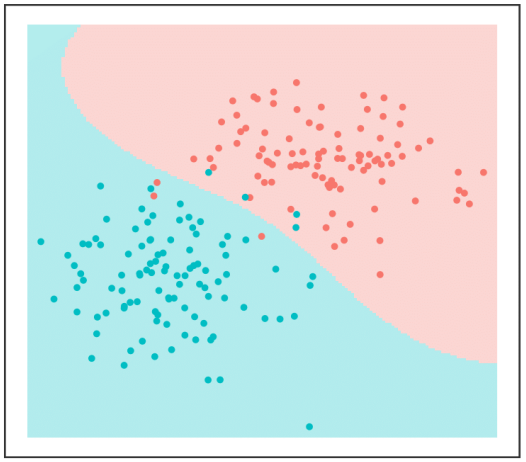
SVMpolynomial

training accuracy: 97 % – test accuracy: 97 %



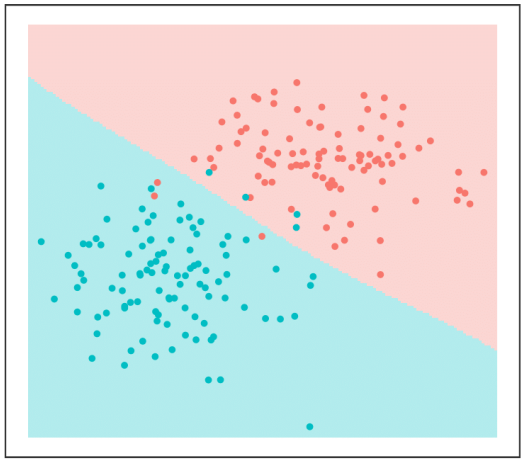
SVMradial

training accuracy: 96.5 % – test accuracy: 96.5 %

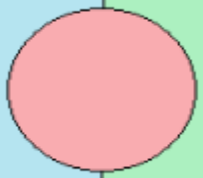


SVMsigmoid

training accuracy: 96.5 % – test accuracy: 97 %



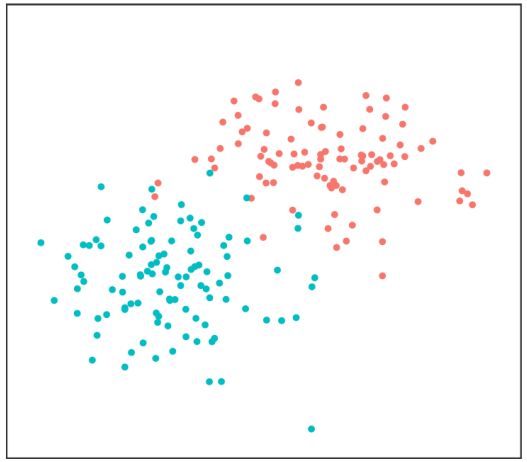
Twonorm



Trees

Dataset

mlbench.twonorm (n=200, d=2)



TreesC.50

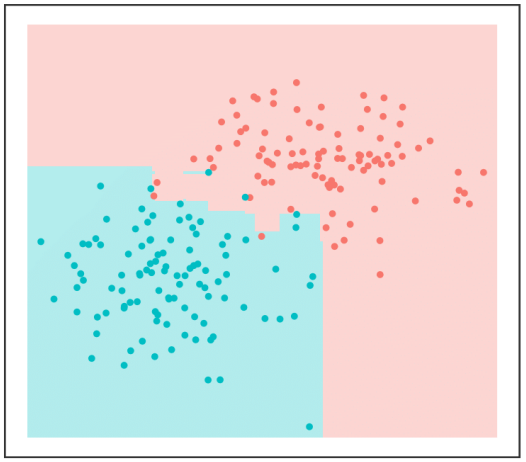
training accuracy: 98 % – test accuracy: 95 %

TreesCART

training accuracy: 97 % – test accuracy: 94.5 %

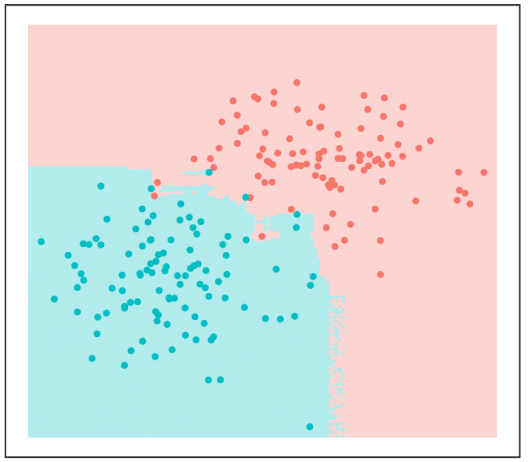
TreesRandom Forest 3 trees

training accuracy: 99.5 % – test accuracy: 94.5 %

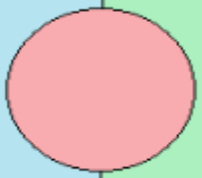


TreesRandom Forest 128 trees

training accuracy: 100 % – test accuracy: 97 %



Twonorm

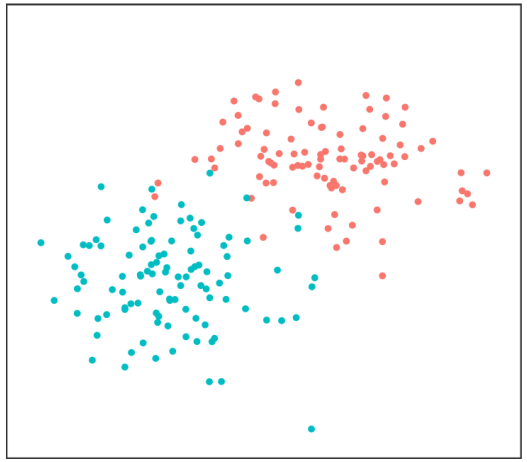


ANN

fixed iterations

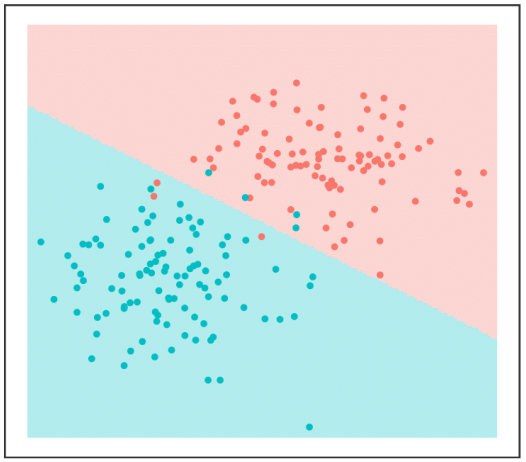
Dataset

mlbench.twonorm(n=200, d=2)



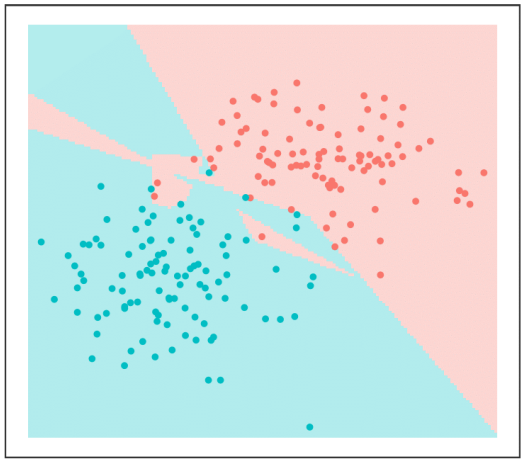
ANN; iters=100; neurons=1

training accuracy: 96.5 % – test accuracy: 96.5 %



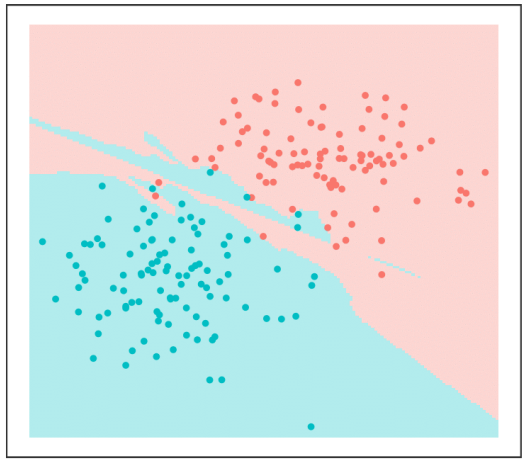
ANN; iters=100; neurons=10

training accuracy: 100 % – test accuracy: 95.5 %



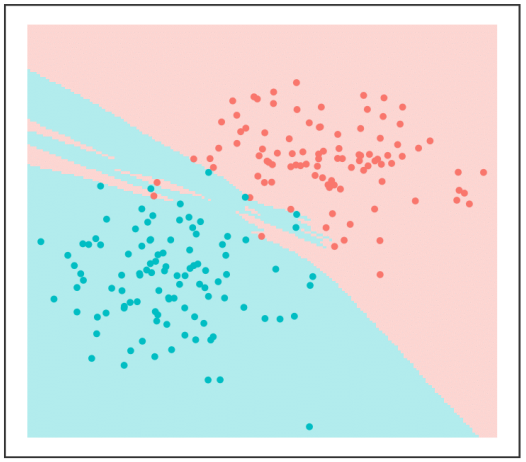
ANN; iters=100; neurons=100

training accuracy: 100 % – test accuracy: 95.5 %



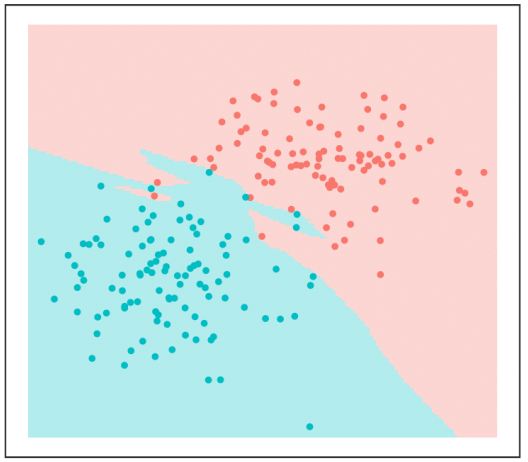
ANN; iters=100; neurons=500

training accuracy: 100 % – test accuracy: 97 %

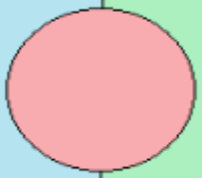


ANN; iters=100; neurons=1000

training accuracy: 100 % – test accuracy: 96 %



Twonorm

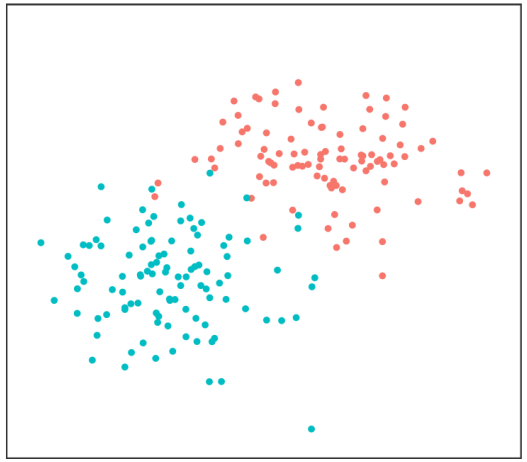


ANN

fixed neurons

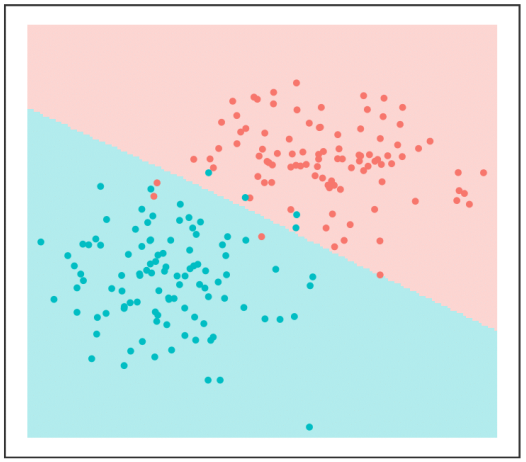
Dataset

mlbench.twonorm(n=200, d=2)



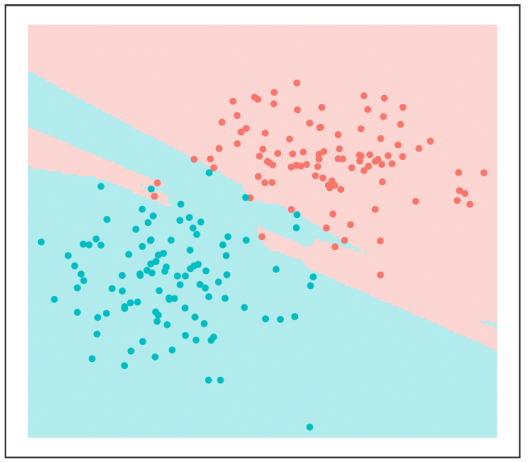
ANN; neurons=10; iters=10

training accuracy: 96.5 % – test accuracy: 97.5 %



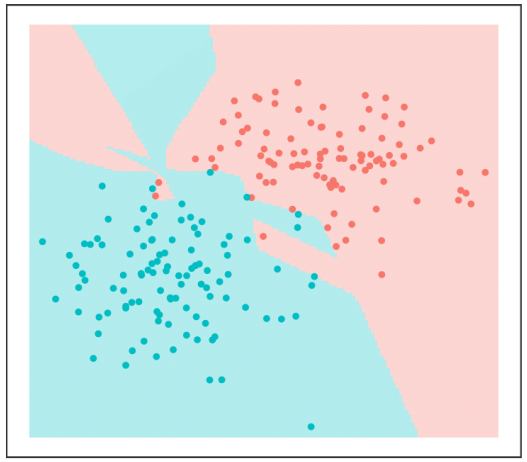
ANN; neurons=10; iters=100

training accuracy: 100 % – test accuracy: 96.5 %



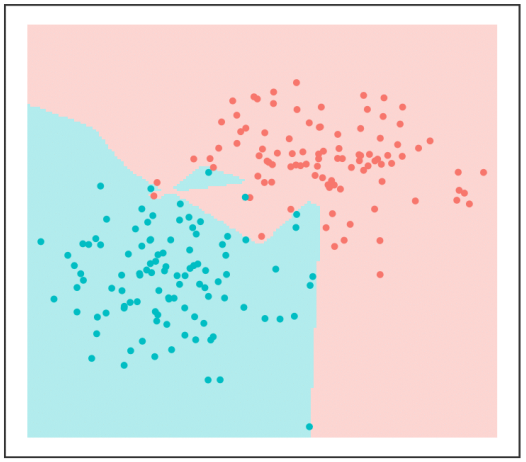
ANN; neurons=10; iters=500

training accuracy: 100 % – test accuracy: 96 %



ANN; neurons=10; iters=1000

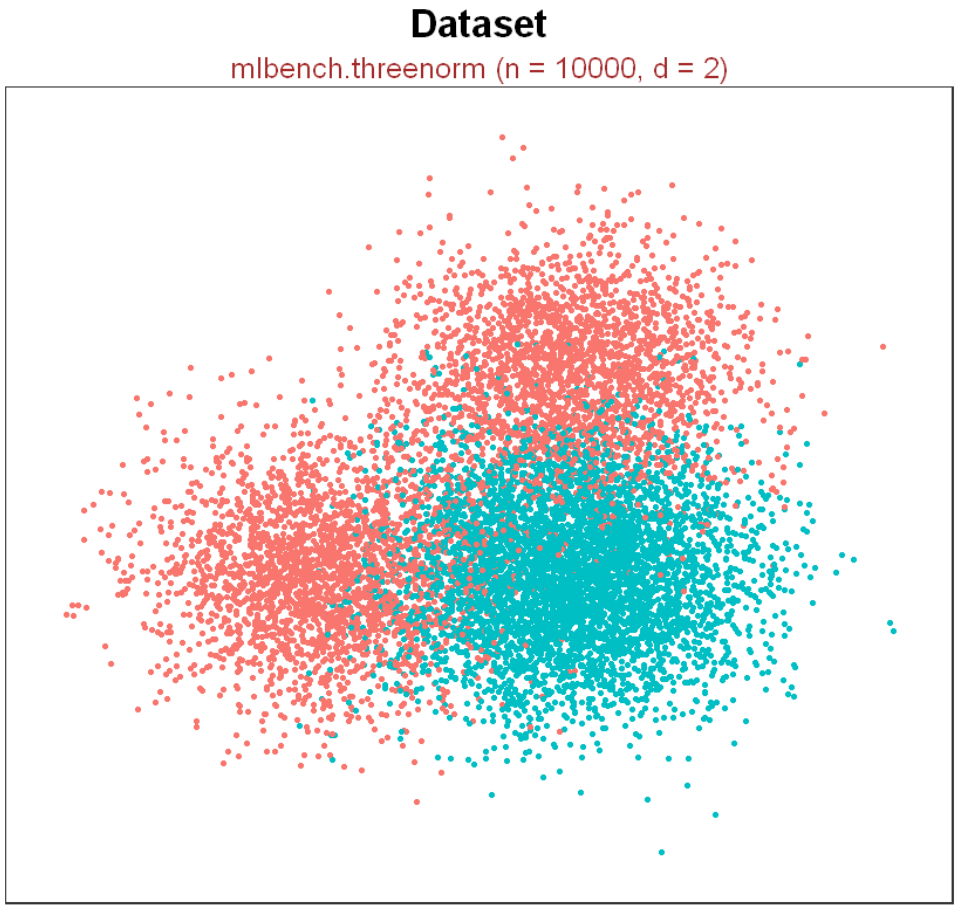
training accuracy: 99.5 % – test accuracy: 95.5 %



ANN; neurons=10; iters=10000

training accuracy: 99 % – test accuracy: 97 %

Threenorm



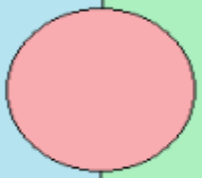
This dataset image contains 10,000 points, so we can se its true form. The next ones will contain a low number in order to have a better view of the decision boundary of each algorithm.

Threenorm Benchmark Problem

The inputs of the threenorm problem are points from two Gaussian distributions with unit covariance matrix. Class 1 is drawn with equal probability from a unit multivariate normal with mean $(a, a, \dots, a)$ and from a unit multivariate normal with mean $(-a, -a, \dots, -a)$. Class 2 is drawn from a multivariate normal with mean at $(a, -a, a, \dots, -a), a = 2/d^{0.5}$.

Usage:
mlbench. threenorm(n, d)

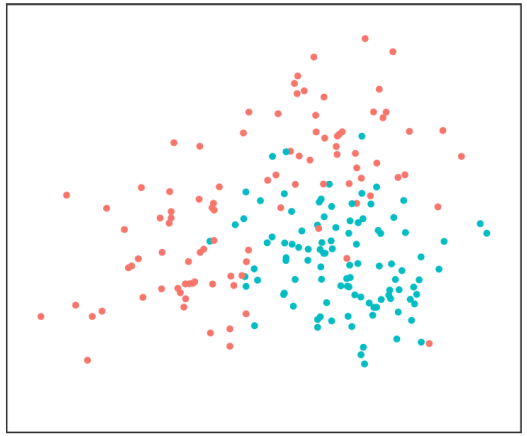
Threenorm



KNN

Dataset

mlbench.threenorm(n=200, d=2)

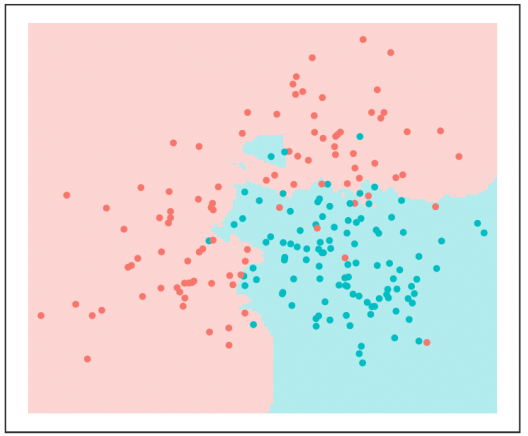


KNN1

training accuracy: 100 % – test accuracy: 82.5 %

KNN3

training accuracy: 92 % – test accuracy: 85.5 %



KNN5

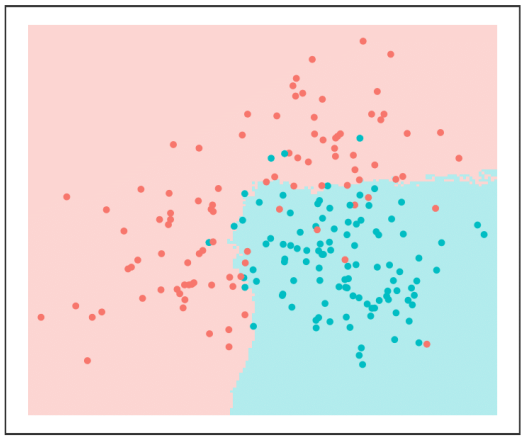
training accuracy: 91.5 % – test accuracy: 86.5 %

KNN10

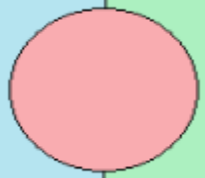
training accuracy: 89.5 % – test accuracy: 86.5 %

KNN30

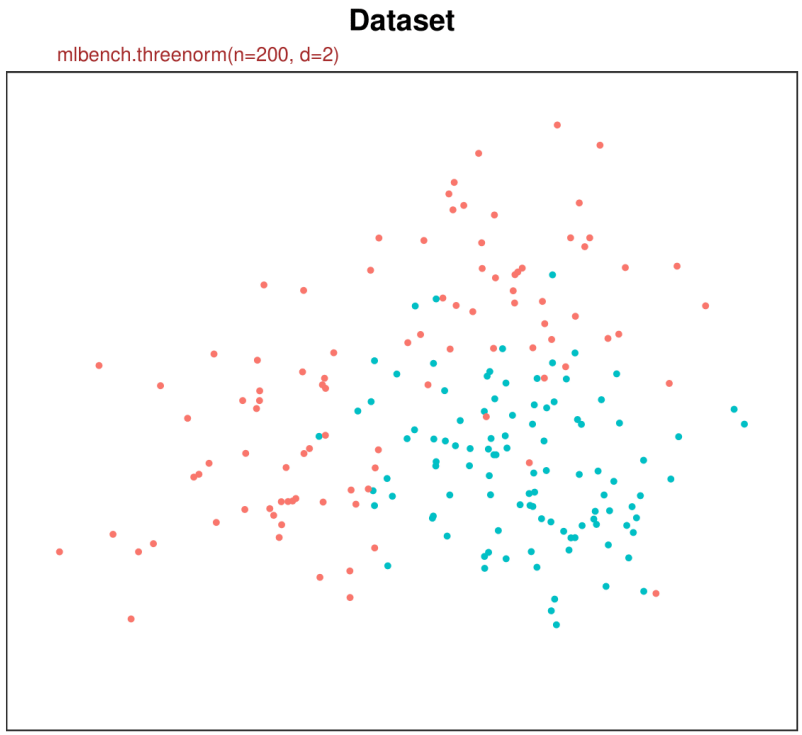
training accuracy: 90.5 % – test accuracy: 86 %

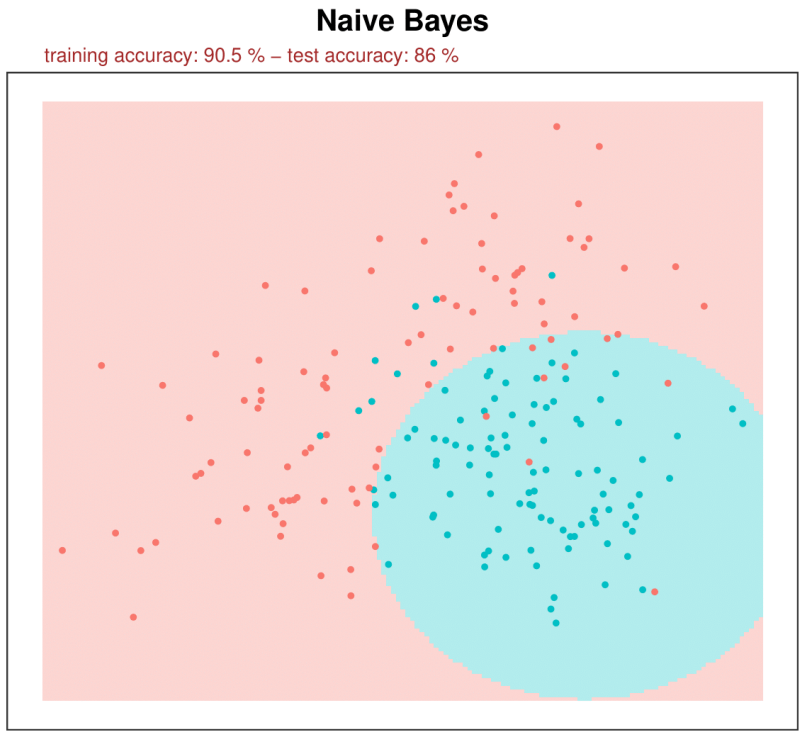


Threenorm

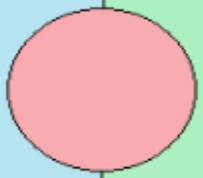


Naïve Bayes





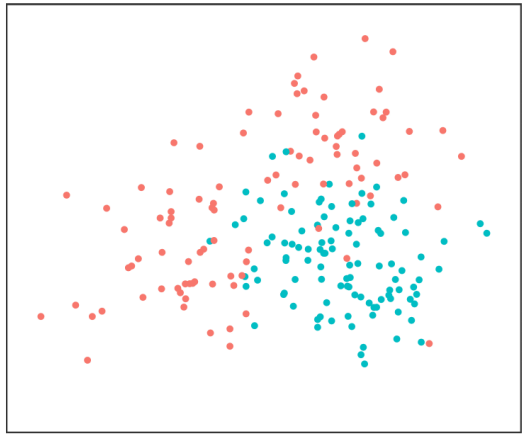
Threenorm



SVM

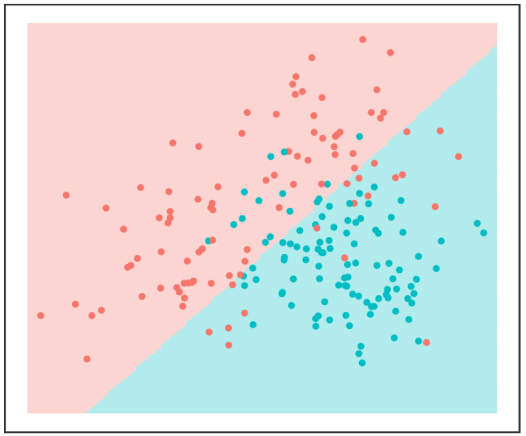
Dataset

mlbench.threenorm(n=200, d=2)



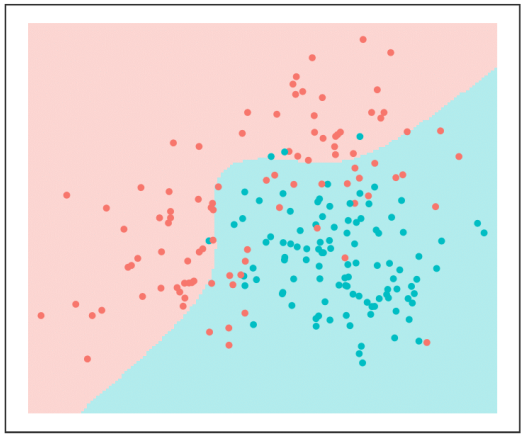
SVMlinear

training accuracy: 83 % – test accuracy: 80.5 %



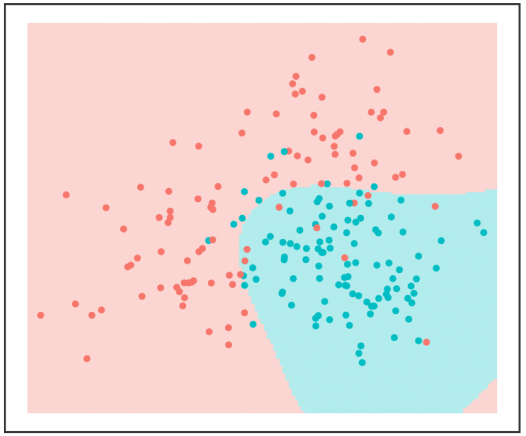
SVMpolynomial

training accuracy: 83 % – test accuracy: 81 %



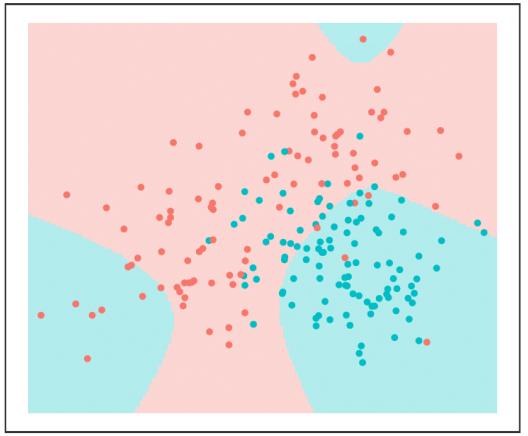
SVMradial

training accuracy: 89.5 % – test accuracy: 85 %

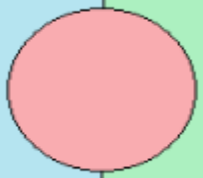


SVMsigmoid

training accuracy: 76.5 % – test accuracy: 74 %



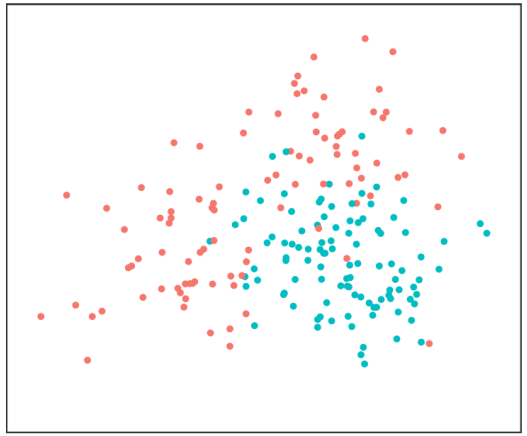
Threenorm



Trees

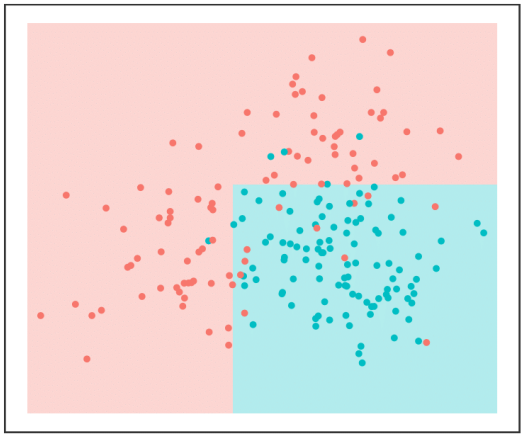
Dataset

mlbench.threenorm(n=200, d=2)



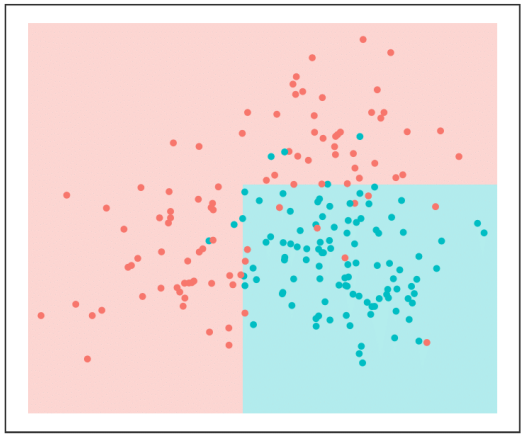
TreesC.50

training accuracy: 92 % – test accuracy: 87 %



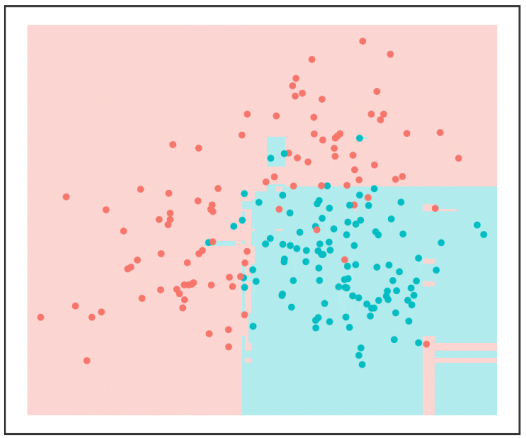
TreesCART

training accuracy: 92 % – test accuracy: 86.5 %



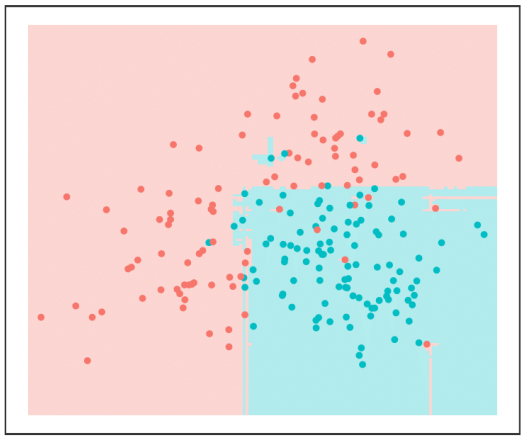
TreesRandom Forest 3 trees

training accuracy: 97 % – test accuracy: 87 %

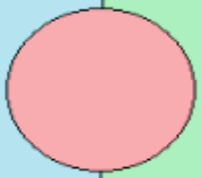


TreesRandom Forest 128 trees

training accuracy: 100 % – test accuracy: 85.5 %



Threenorm

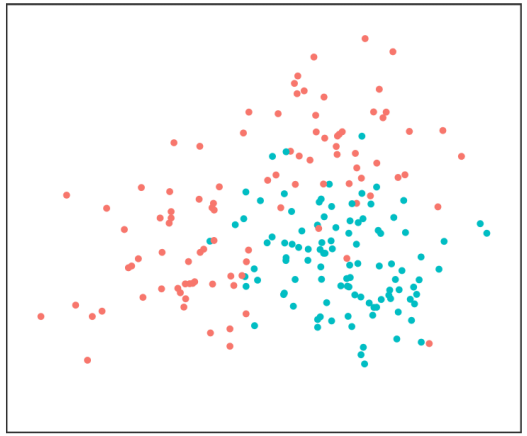


ANN

fixed iterations

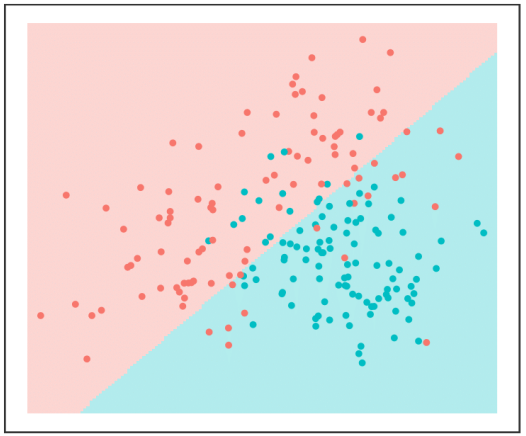
Dataset

mlbench.threenorm(n=200, d=2)



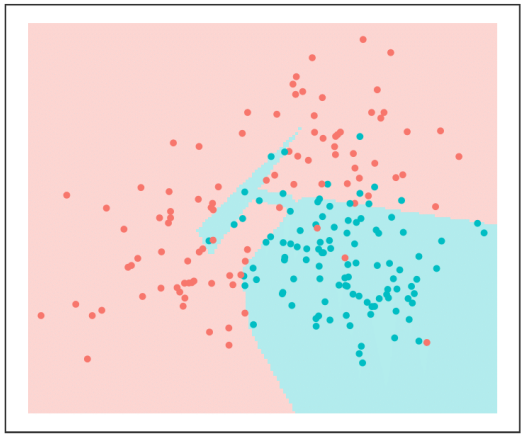
ANN; iters=100; neurons=1

training accuracy: 82.5 % – test accuracy: 81.5 %



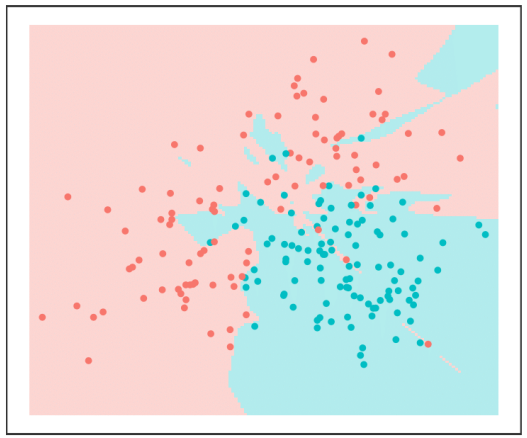
ANN; iters=100; neurons=10

training accuracy: 94.5 % – test accuracy: 84.5 %



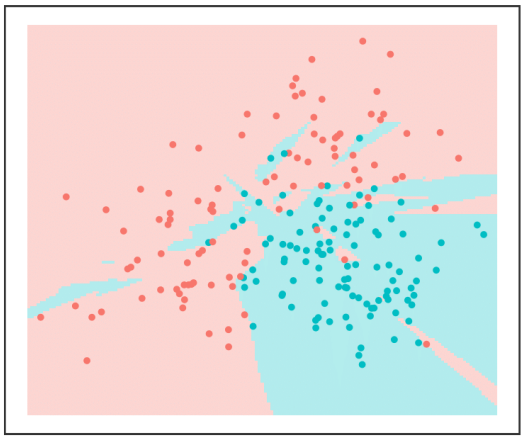
ANN; iters=100; neurons=100

training accuracy: 100 % – test accuracy: 82 %



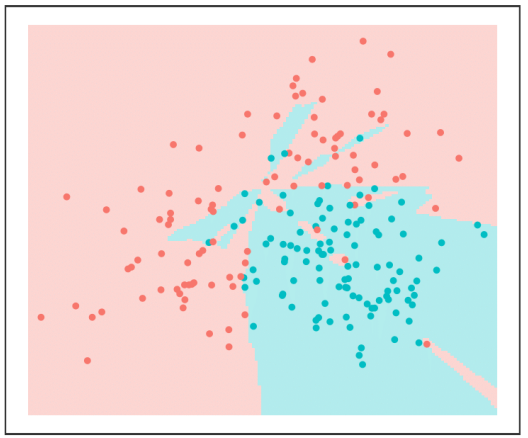
ANN; iters=100; neurons=500

training accuracy: 100 % – test accuracy: 80.5 %

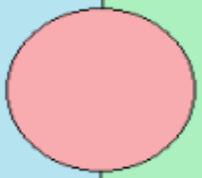


ANN; iters=100; neurons=1000

training accuracy: 100 % – test accuracy: 82.5 %



Threenorm

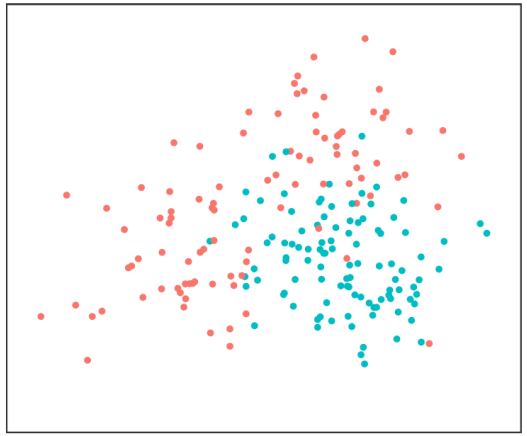


ANN

fixed neurons

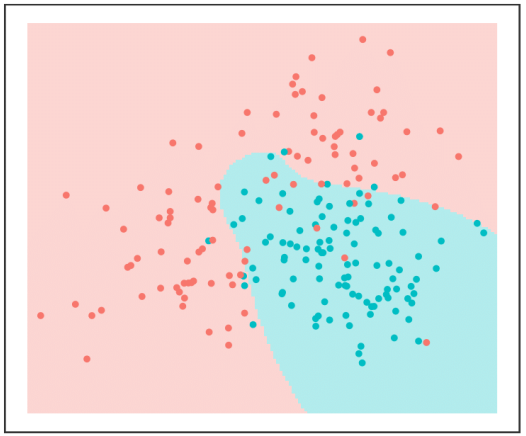
Dataset

mlbench.threenorm(n=200, d=2)



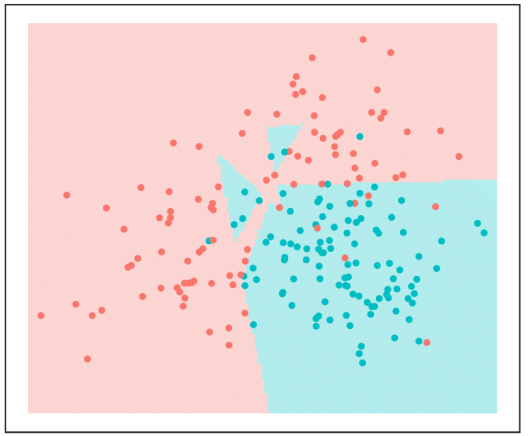
ANN; neurons=10; iters=10

training accuracy: 89.5 % – test accuracy: 85 %



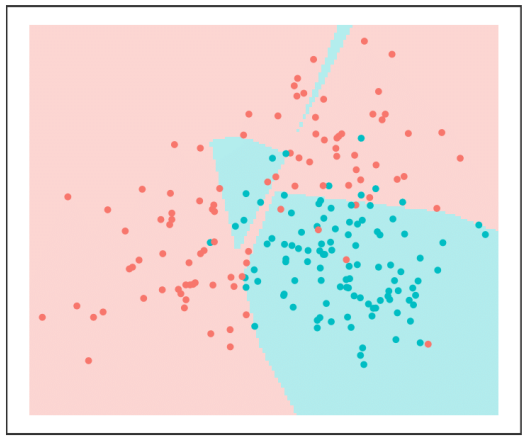
ANN; neurons=10; iters=100

training accuracy: 96 % – test accuracy: 87 %



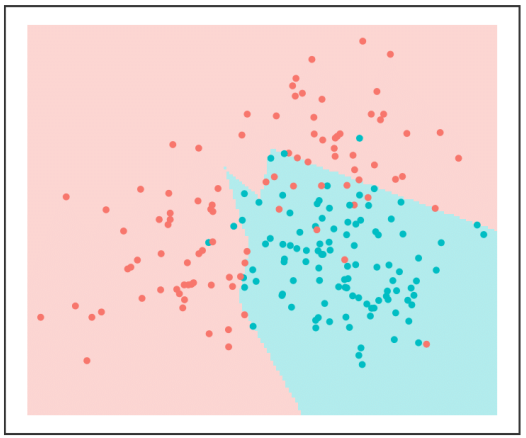
ANN; neurons=10; iters=500

training accuracy: 94 % – test accuracy: 85.5 %



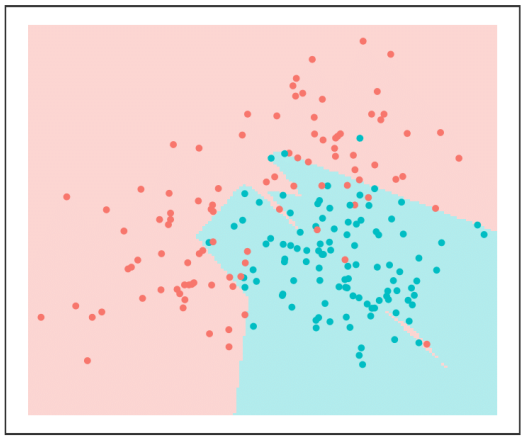
ANN; neurons=10; iters=1000

training accuracy: 93 % – test accuracy: 86 %

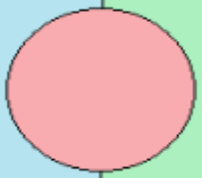


ANN; neurons=10; iters=10000

training accuracy: 94.5 % – test accuracy: 86 %

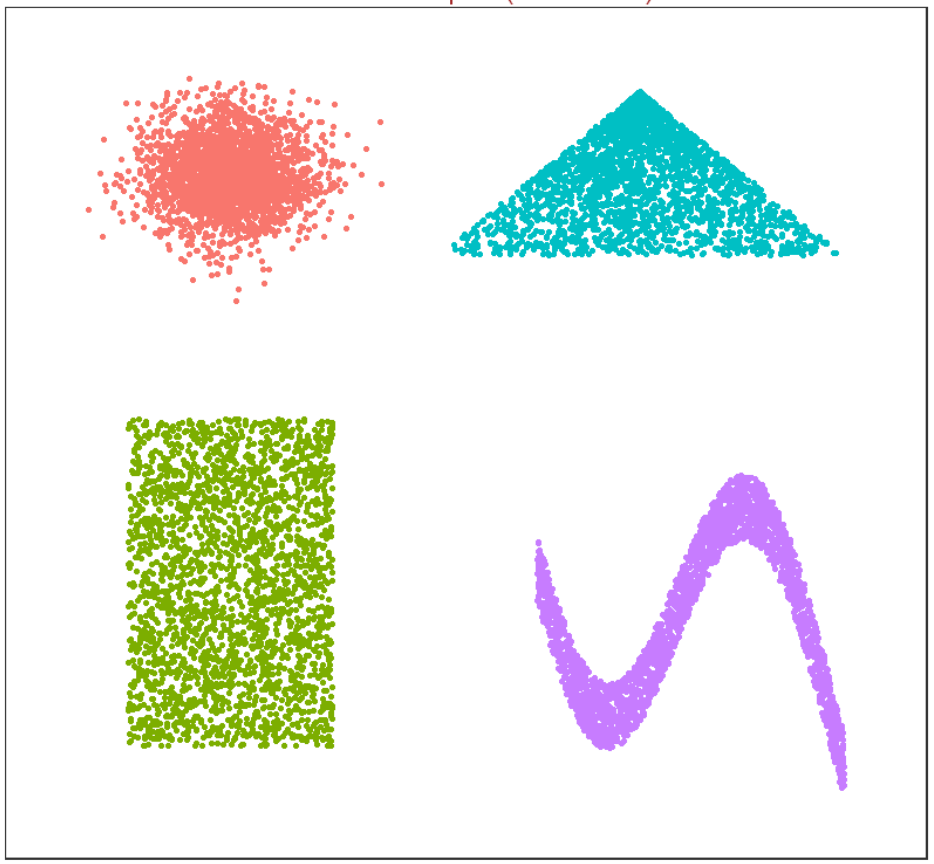


Shapes



Dataset

mlbench.shapes (n = 10000)



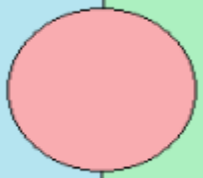
This dataset image contains 10,000 points, so we can se its true form. The next ones will contain a low number in order to have a better view of the decision boundary of each algorithm.

Shapes in 2d

A Gaussian, square, triangle and wave in 2 dimensions.

Usage:
mlbench.shapes(n)

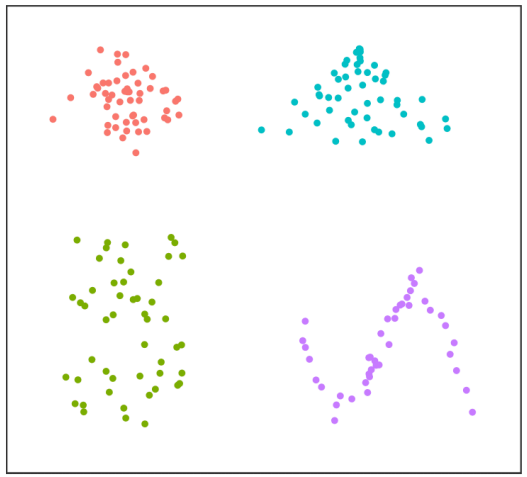
Shapes



KNN

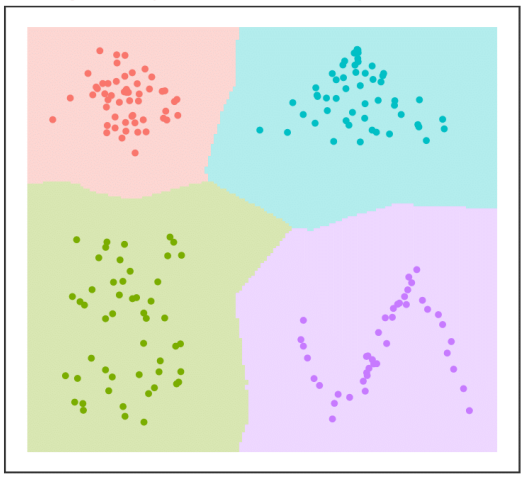
Dataset

mlbench.shapes (n=200)



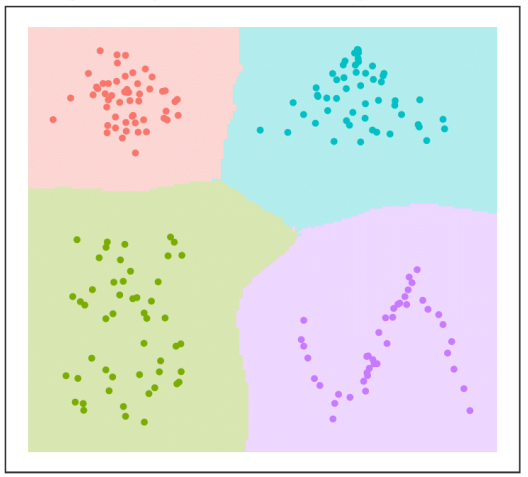
KNN1

training accuracy: 100 % – test accuracy: 100 %



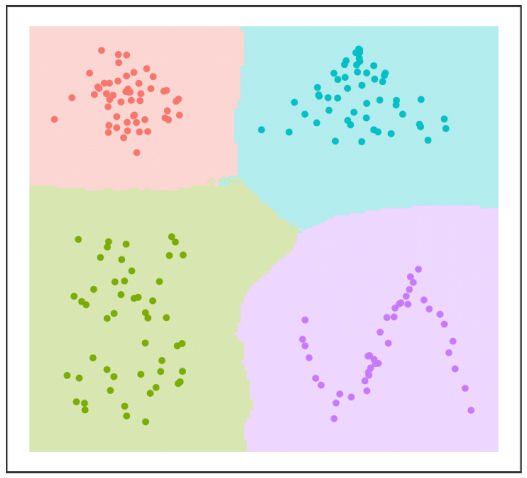
KNN3

training accuracy: 100 % – test accuracy: 100 %



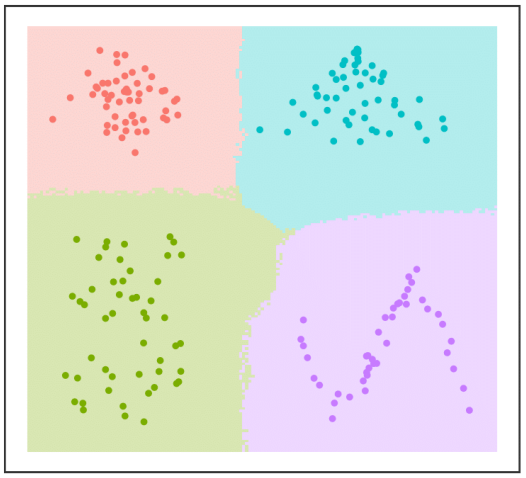
KNN5

training accuracy: 100 % – test accuracy: 100 %



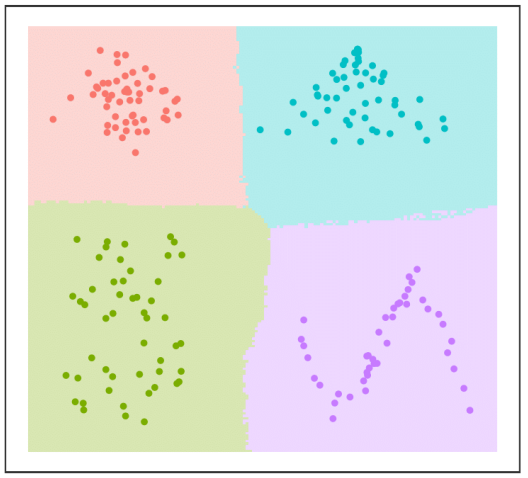
KNN10

training accuracy: 100 % – test accuracy: 100 %

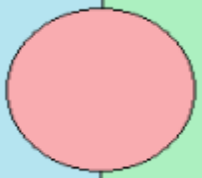


KNN30

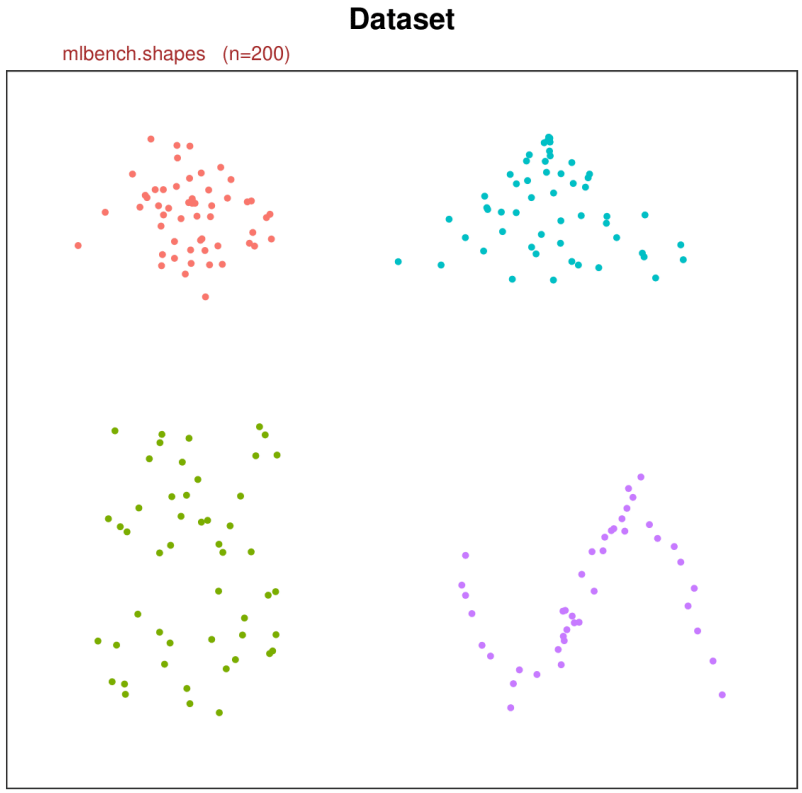
training accuracy: 100 % – test accuracy: 100 %

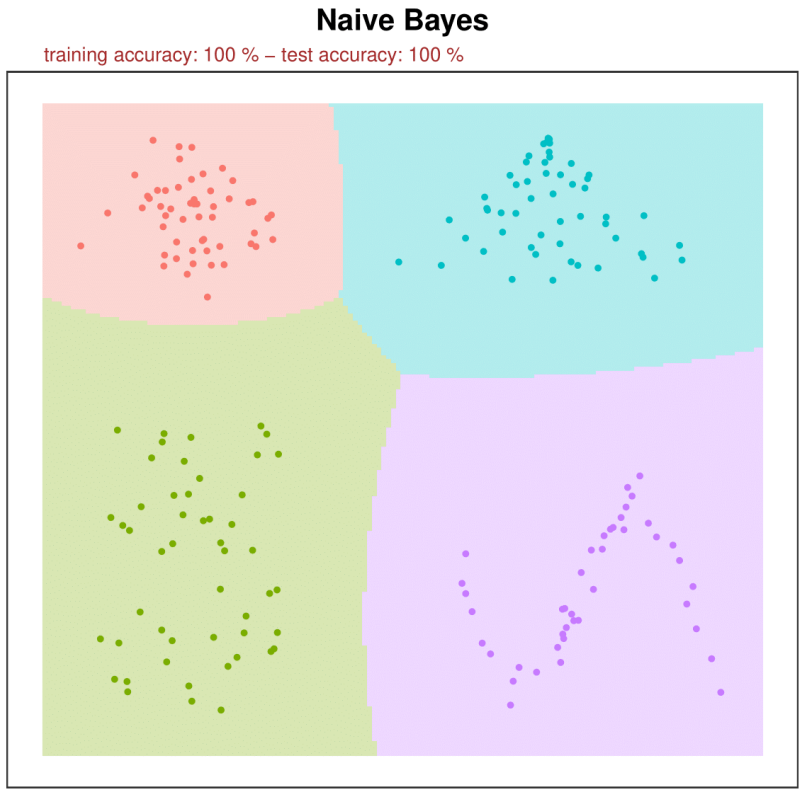


Shapes

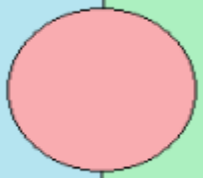


Naïve Bayes





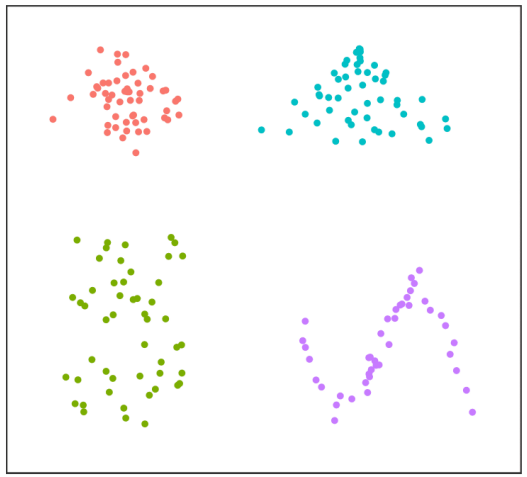
Shapes



SVM

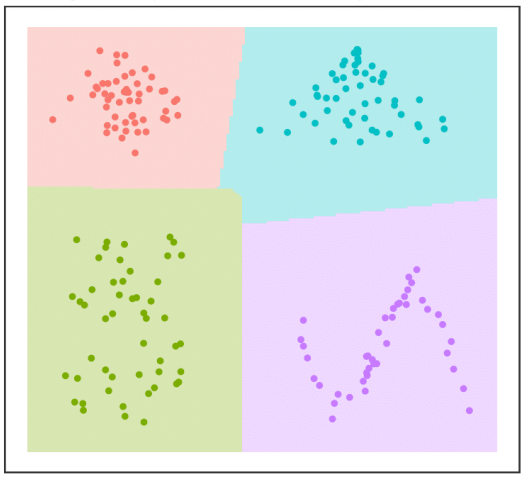
Dataset

mlbench.shapes (n=200)



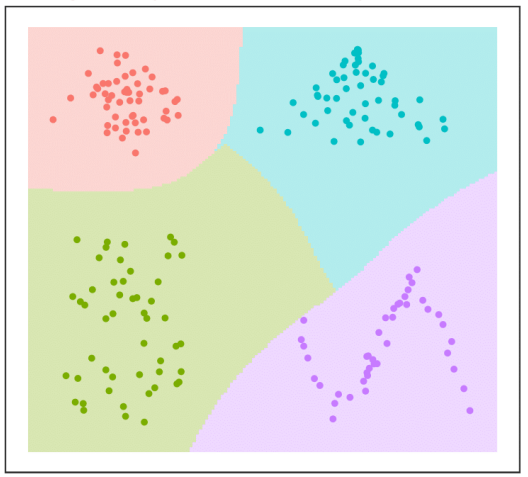
SVMlinear

training accuracy: 100 % – test accuracy: 100 %



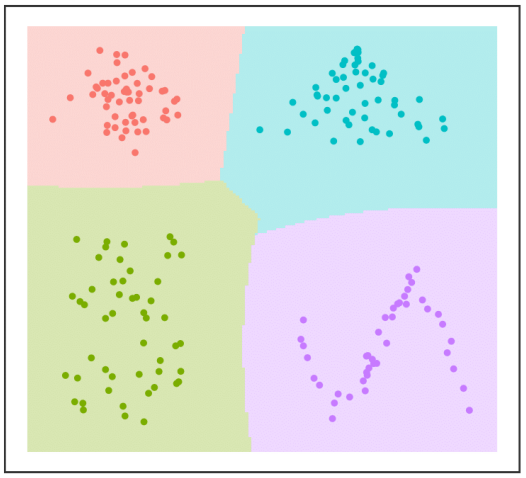
SVMpolynomial

training accuracy: 100 % – test accuracy: 100 %



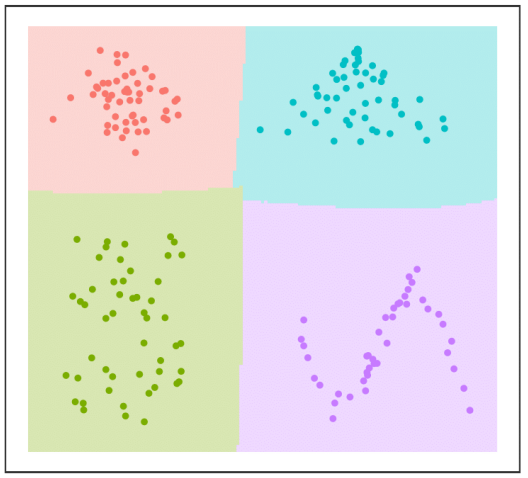
SVMradial

training accuracy: 100 % – test accuracy: 100 %

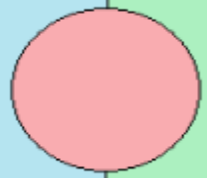


SVMsigmoid

training accuracy: 100 % – test accuracy: 100 %



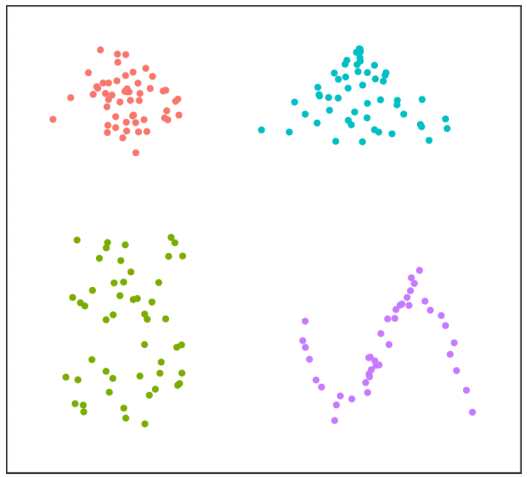
Shapes



Trees

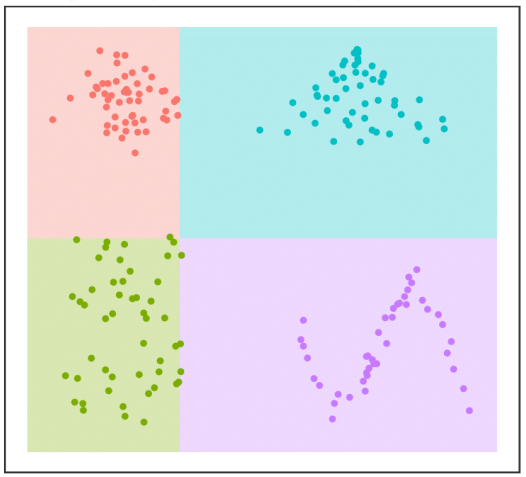
Dataset

mlbench.shapes (n=200)



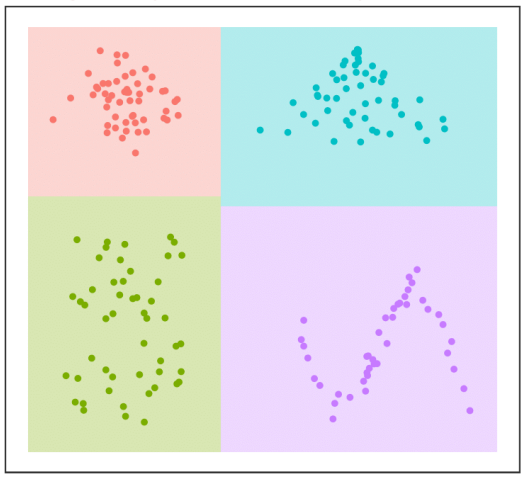
TreesC.50

training accuracy: 100 % – test accuracy: 99.5 %



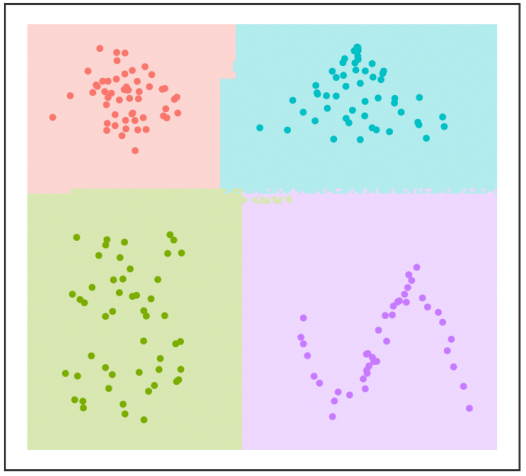
TreesCART

training accuracy: 100 % – test accuracy: 100 %



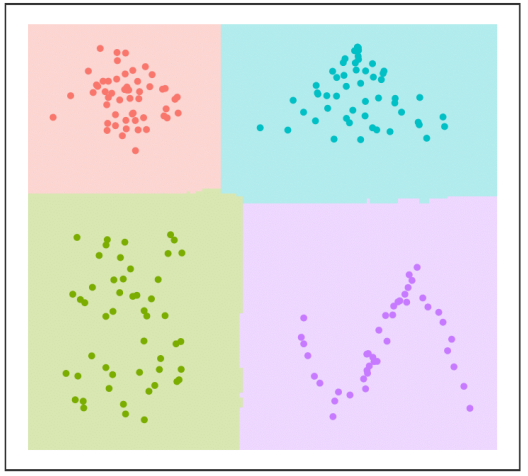
TreesRandom Forest 3 trees

training accuracy: 100 % – test accuracy: 100 %

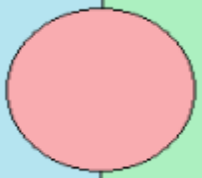


TreesRandom Forest 128 trees

training accuracy: 100 % – test accuracy: 100 %



Shapes

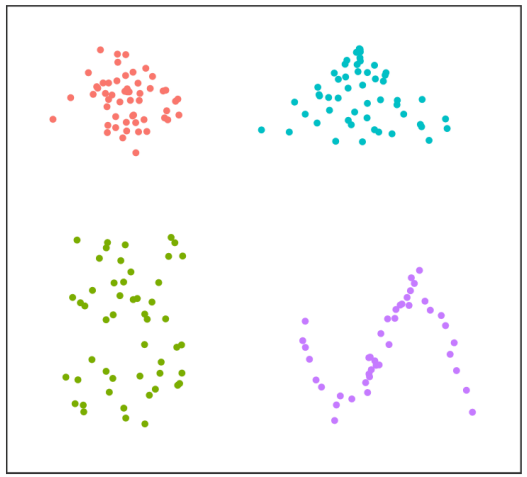


ANN

fixed iterations

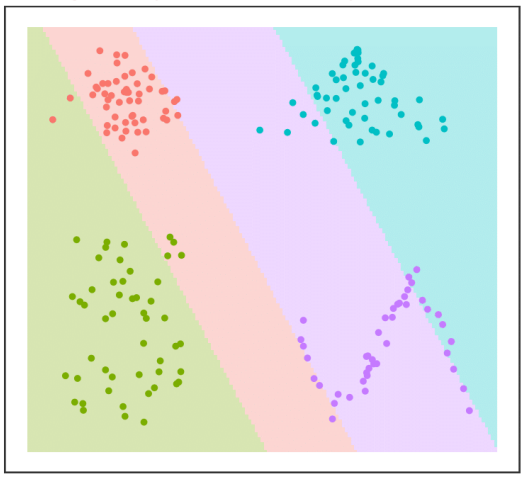
Dataset

mlbench.shapes(n=200)



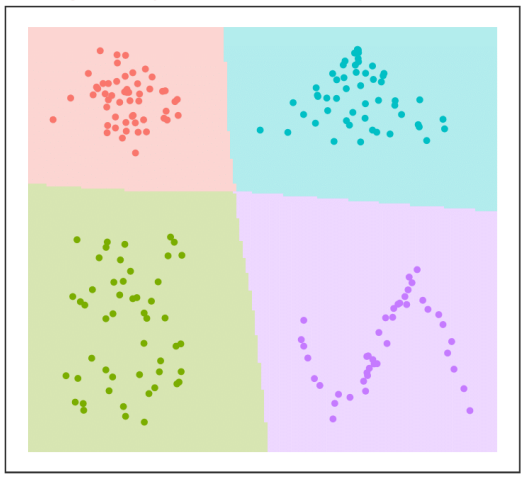
ANN; iters=100; neurons=1

training accuracy: 87 % – test accuracy: 83 %



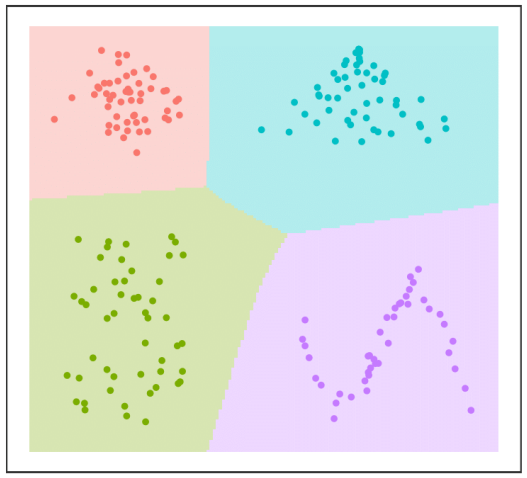
ANN; iters=100; neurons=10

training accuracy: 100 % – test accuracy: 100 %



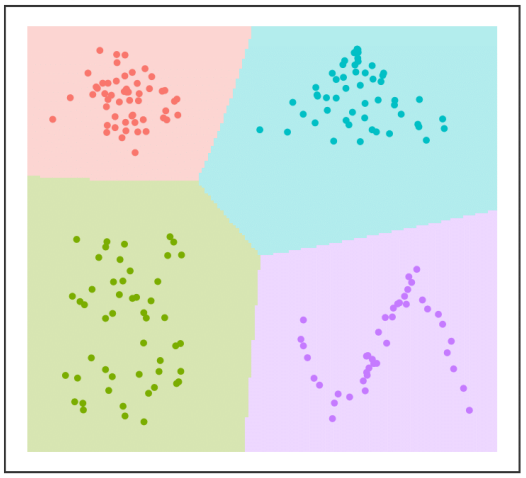
ANN; iters=100; neurons=100

training accuracy: 100 % – test accuracy: 100 %



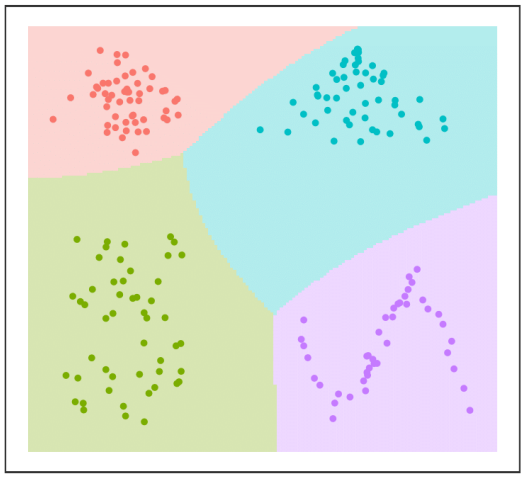
ANN; iters=100; neurons=500

training accuracy: 100 % – test accuracy: 100 %

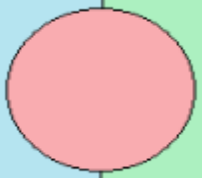


ANN; iters=100; neurons=1000

training accuracy: 100 % – test accuracy: 100 %



Shapes

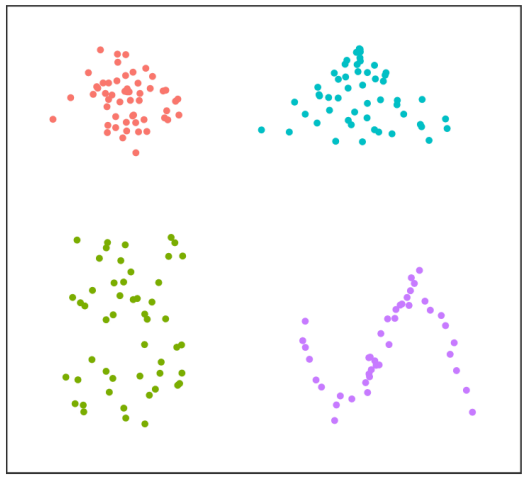


ANN

fixed neurons

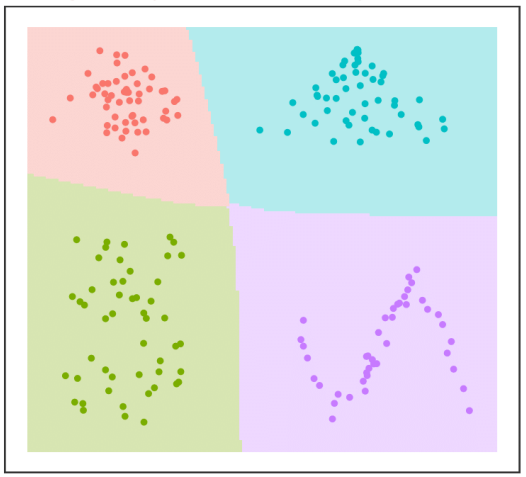
Dataset

mlbench.shapes(n=200)



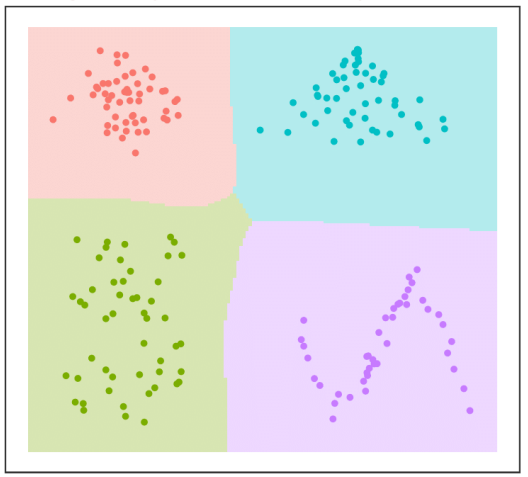
ANN; neurons=10; iters=10

training accuracy: 100 % – test accuracy: 100 %



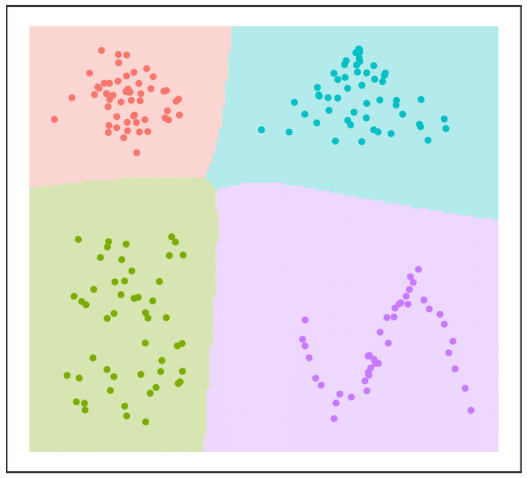
ANN; neurons=10; iters=100

training accuracy: 100 % – test accuracy: 100 %



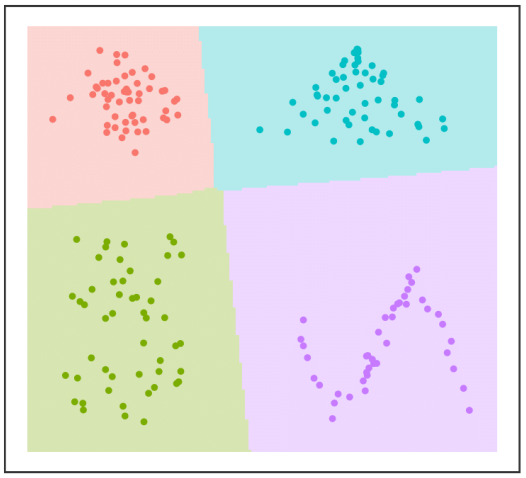
ANN; neurons=10; iters=500

training accuracy: 100 % – test accuracy: 100 %



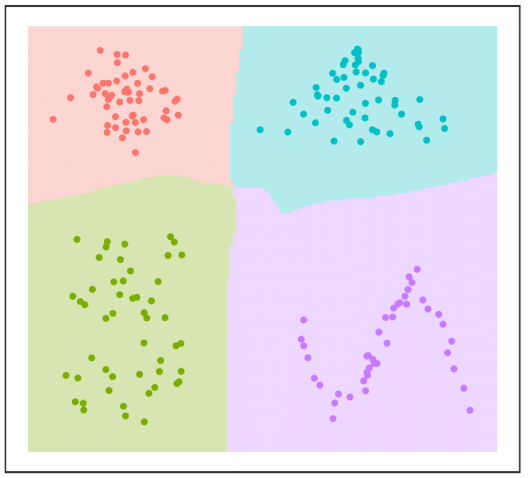
ANN; neurons=10; iters=1000

training accuracy: 100 % – test accuracy: 100 %

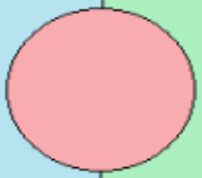


ANN; neurons=10; iters=10000

training accuracy: 100 % – test accuracy: 100 %

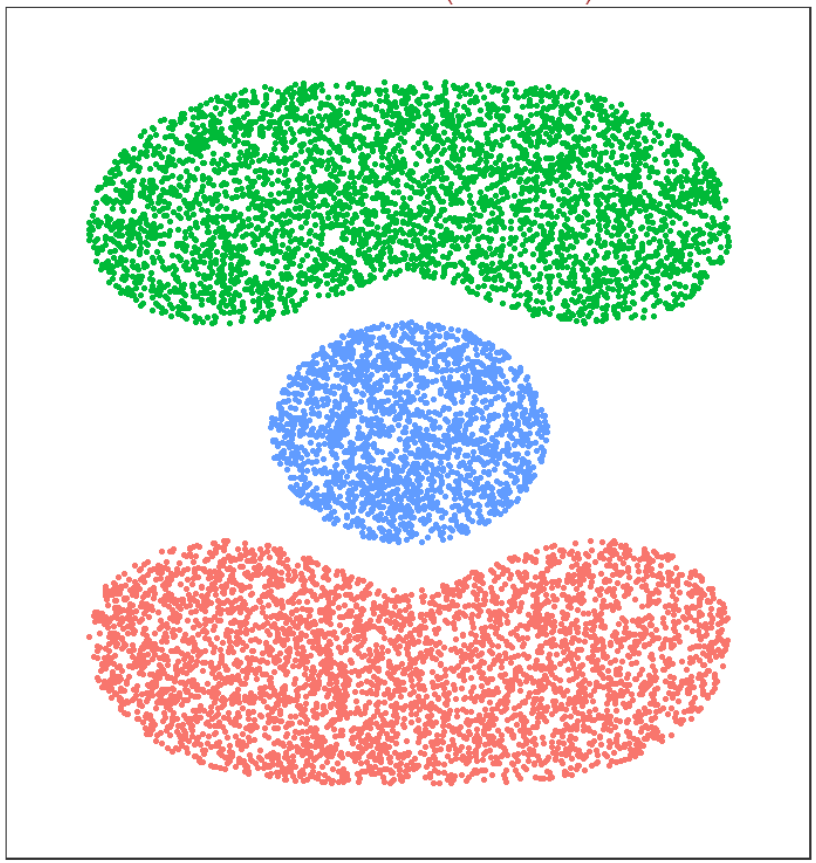


Cassini



Dataset

mlbench.cassini (n = 10000)



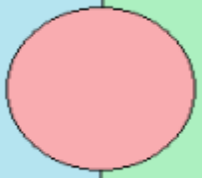
This dataset image contains 10,000 points, so we can se its true form. The next ones will contain a low number in order to have a better view of the decision boundary of each algorithm.

Cassini: A 2 Dimensional Problem

The inputs of the cassini problem are uniformly distributed on a 2-dimensional space within 3 structures. The 2 external structures (classes) are banana-shaped structures and in between them, the middle structure (class) is a circle.

Usage:
mlbench.cassini(n, relsize)

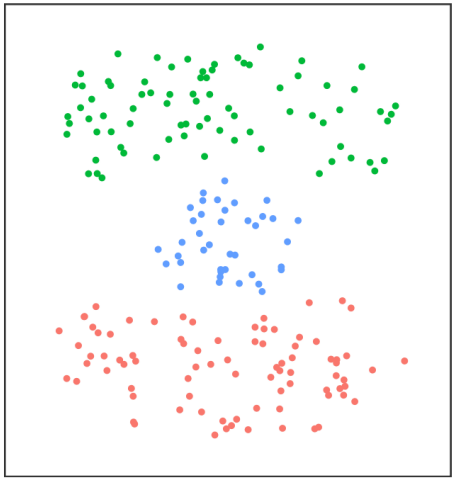
Cassini



KNN

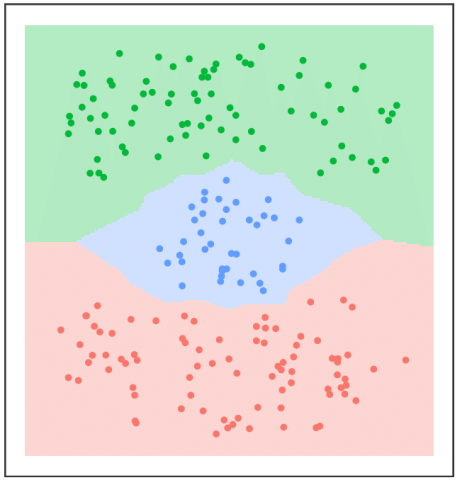
Dataset

mlbench.cassini (n=200)



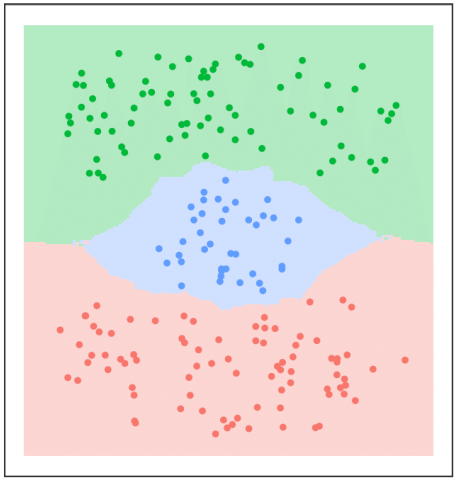
KNN1

training accuracy: 100 % – test accuracy: 100 %



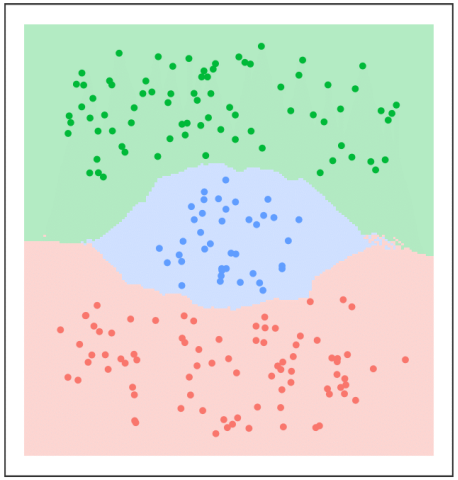
KNN3

training accuracy: 100 % – test accuracy: 100 %



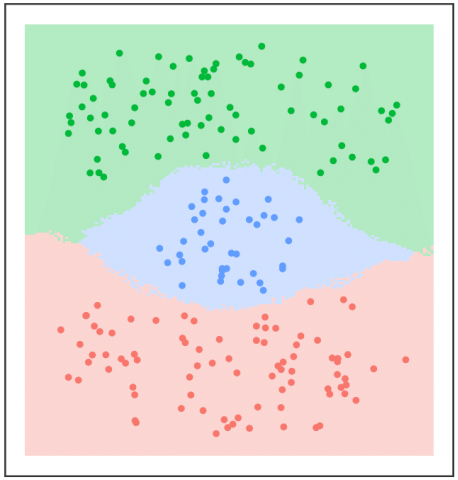
KNN5

training accuracy: 100 % – test accuracy: 100 %



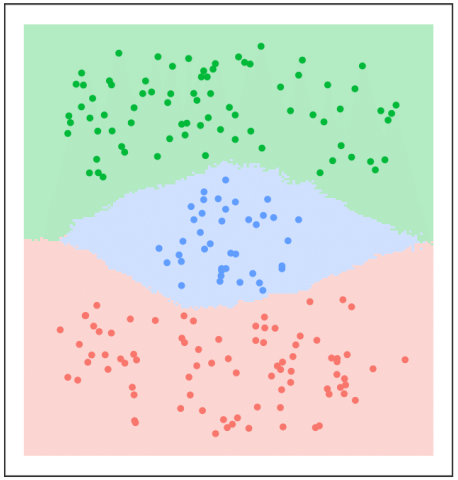
KNN10

training accuracy: 100 % – test accuracy: 100 %

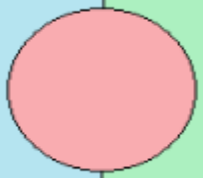


KNN30

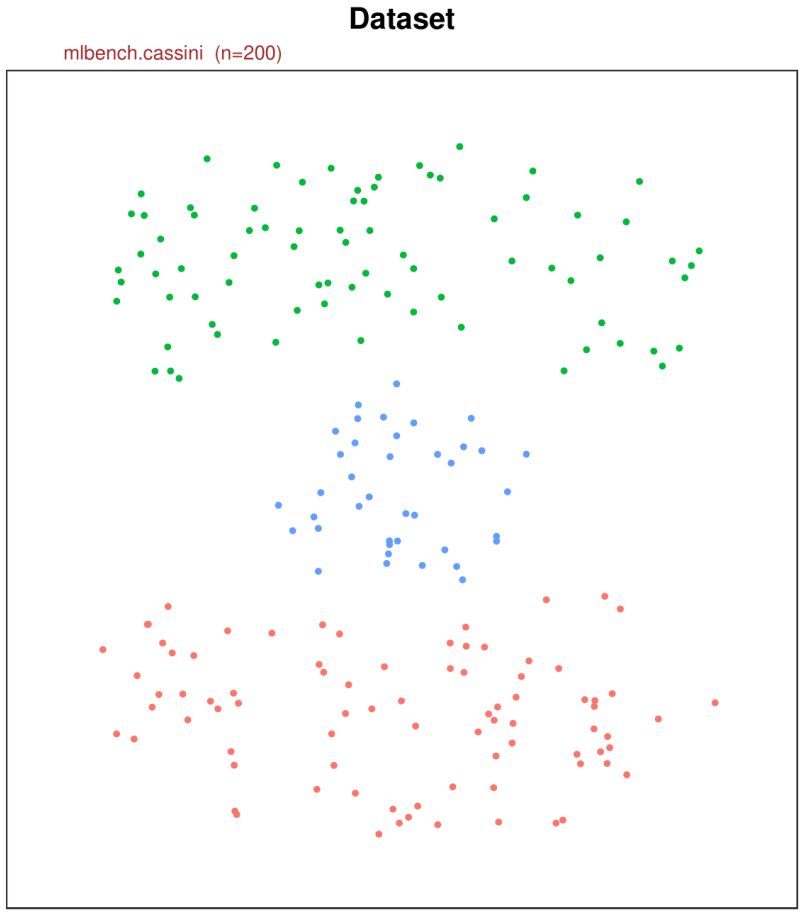
training accuracy: 100 % – test accuracy: 100 %

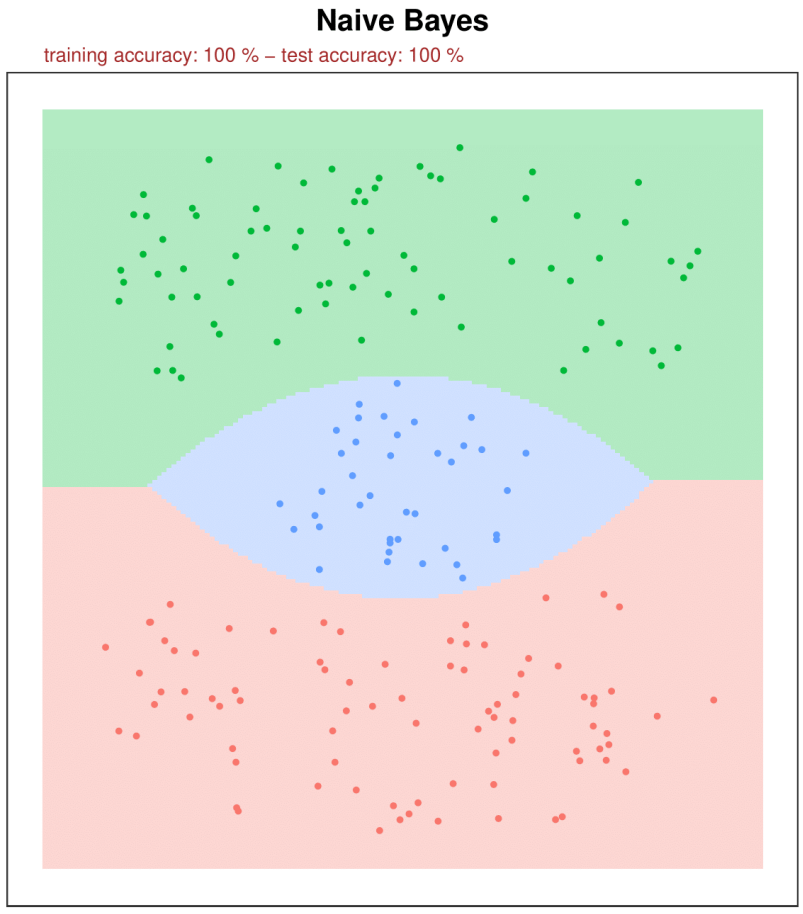


Cassini

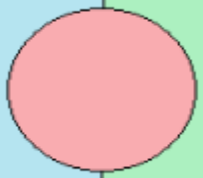


Naïve Bayes





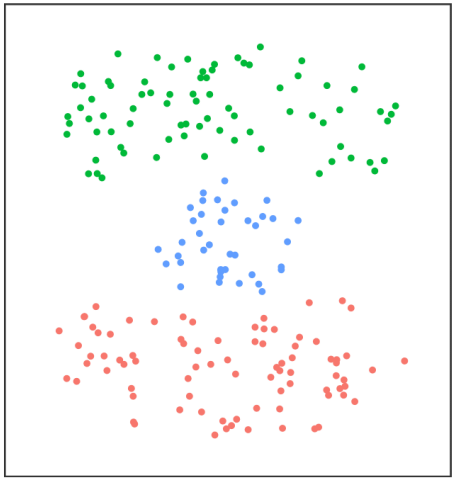
Cassini



SVM

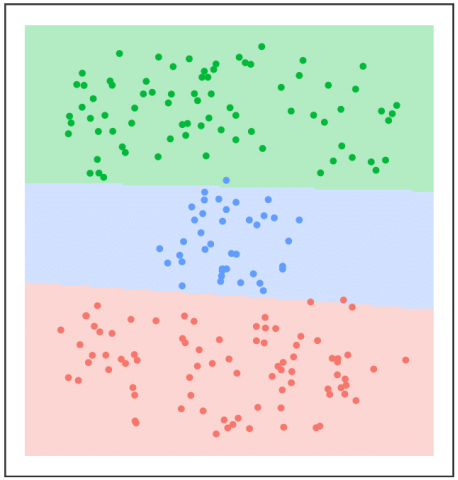
Dataset

mlbench.cassini (n=200)



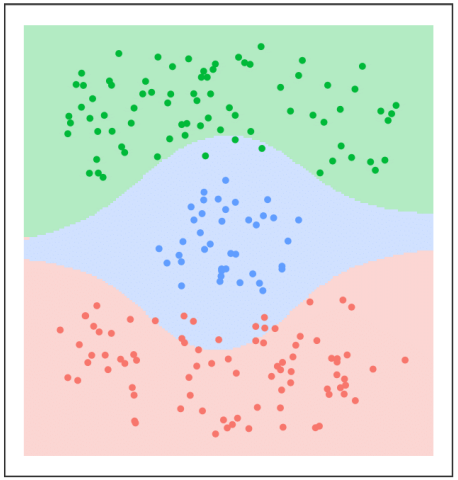
SVMlinear

training accuracy: 99 % – test accuracy: 98.5 %



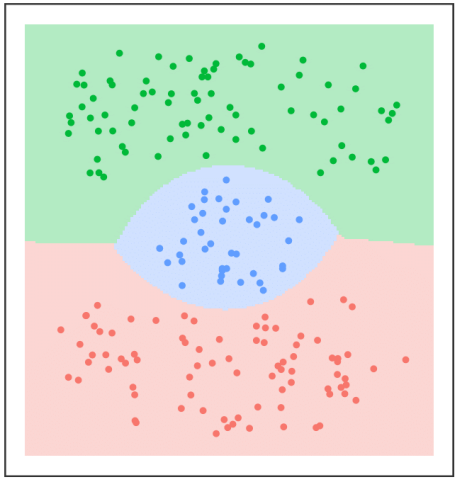
SVMpolynomial

training accuracy: 94.5 % – test accuracy: 96 %



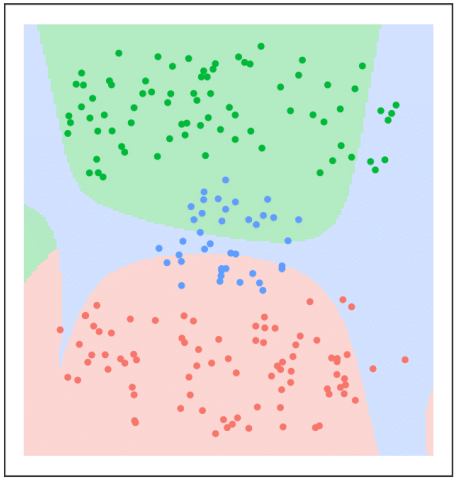
SVMradial

training accuracy: 100 % – test accuracy: 100 %

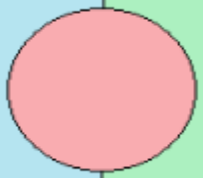


SVMsigmoid

training accuracy: 78 % – test accuracy: 76.5 %



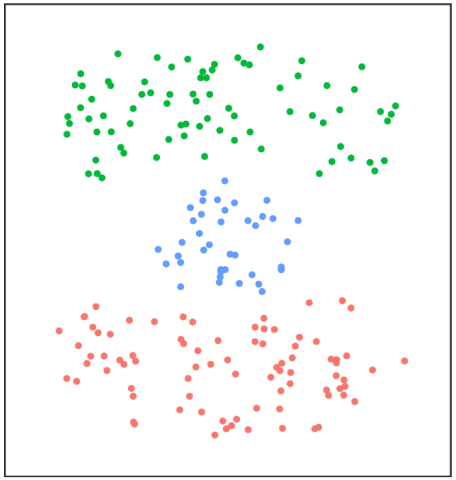
Cassini



Trees

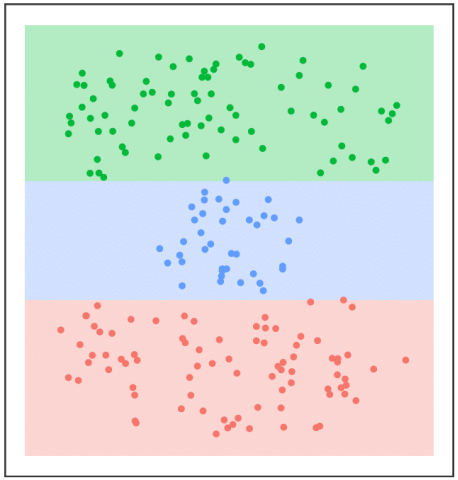
Dataset

mlbench.cassini (n=200)



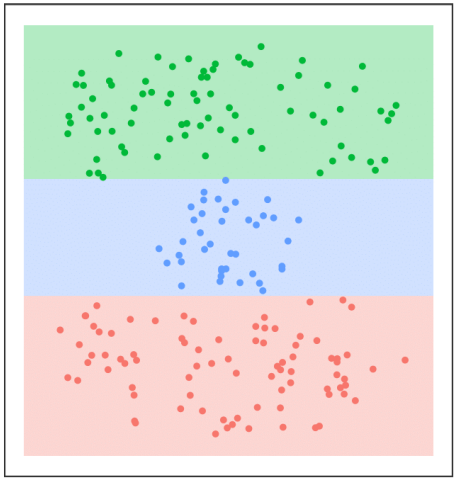
TreesC.50

training accuracy: 100 % – test accuracy: 99.5 %



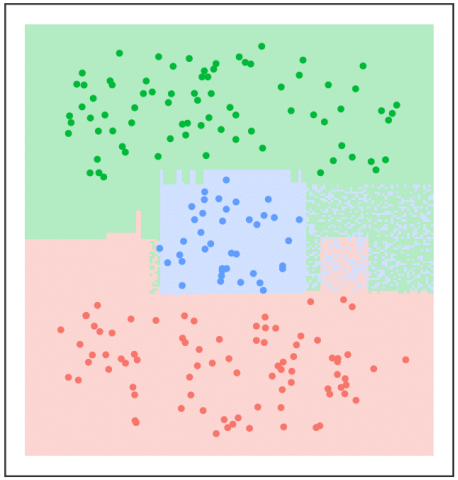
TreesCART

training accuracy: 100 % – test accuracy: 99 %



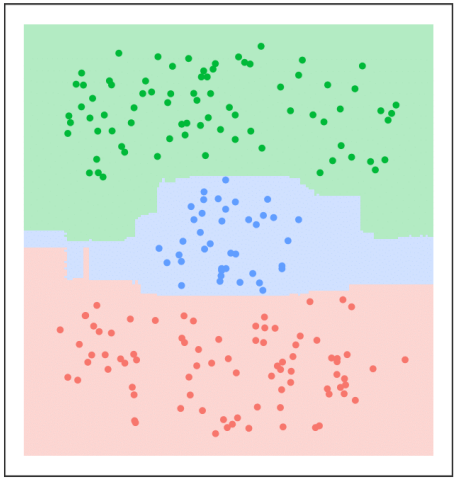
TreesRandom Forest 3 trees

training accuracy: 100 % – test accuracy: 99 %

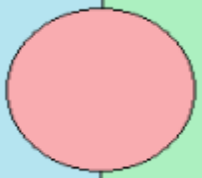


TreesRandom Forest 128 trees

training accuracy: 100 % – test accuracy: 99 %



Cassini

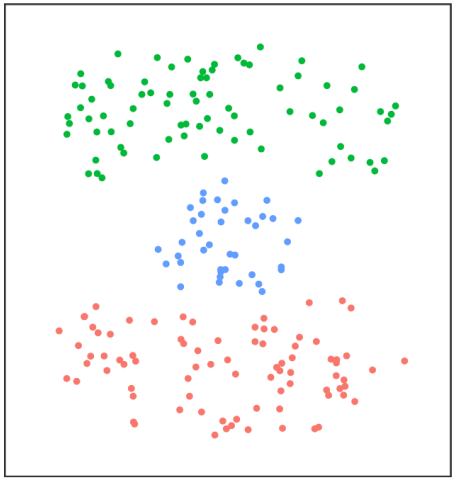


ANN

fixed iterations

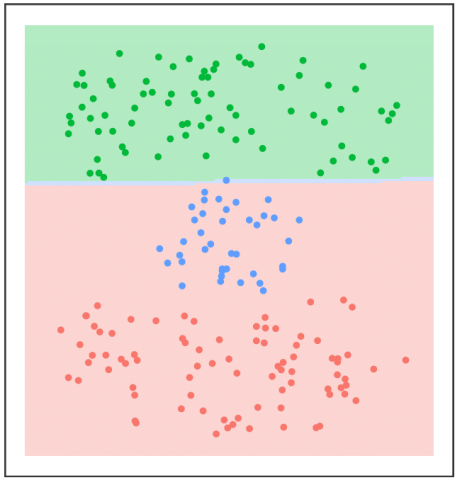
Dataset

mlbench.cassini(n=200)



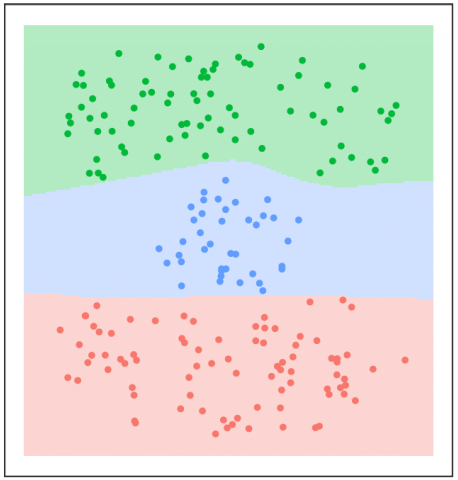
ANN; iters=100; neurons=1

training accuracy: 81 % – test accuracy: 79.5 %



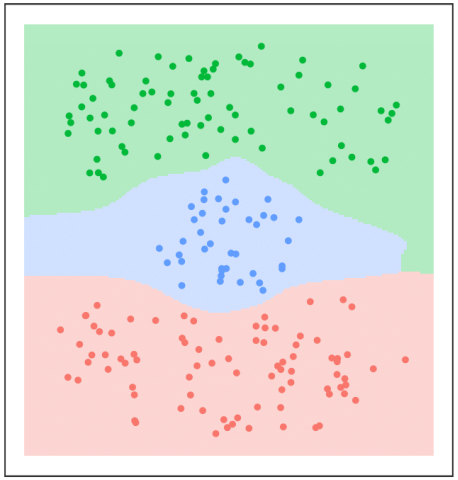
ANN; iters=100; neurons=10

training accuracy: 100 % – test accuracy: 99.5 %



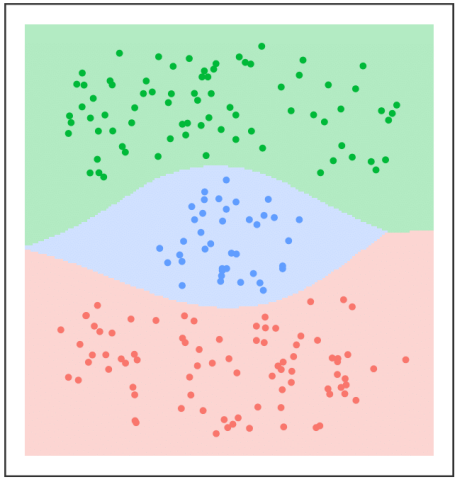
ANN; iters=100; neurons=100

training accuracy: 100 % – test accuracy: 100 %



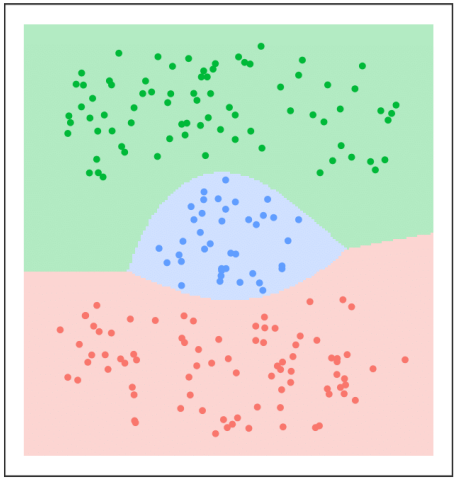
ANN; iters=100; neurons=500

training accuracy: 100 % – test accuracy: 100 %

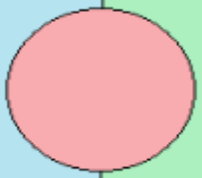


ANN; iters=100; neurons=1000

training accuracy: 100 % – test accuracy: 99.5 %



Cassini

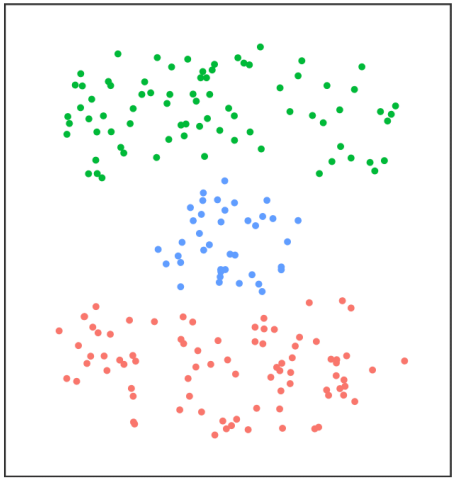


ANN

fixed neurons

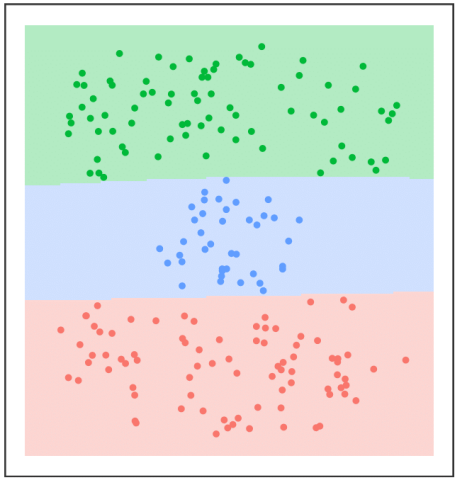
Dataset

mlbench.cassini(n=200)



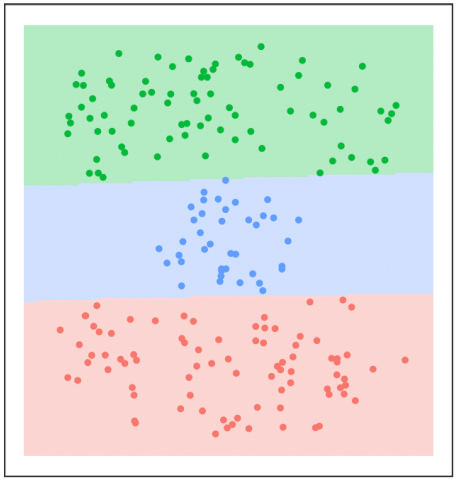
ANN; neurons=10; iters=10

training accuracy: 100 % – test accuracy: 99.5 %



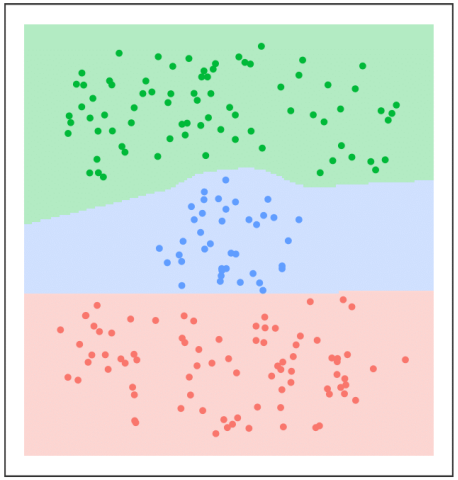
ANN; neurons=10; iters=100

training accuracy: 100 % – test accuracy: 99.5 %



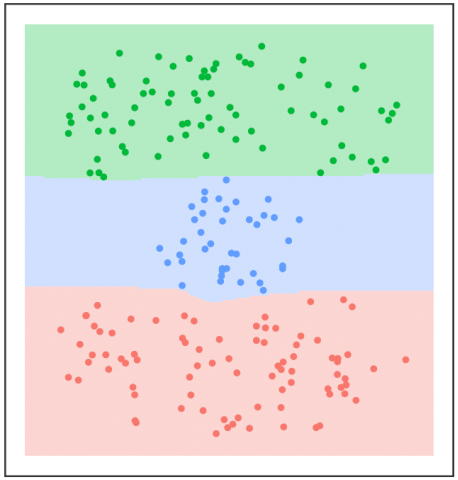
ANN; neurons=10; iters=500

training accuracy: 100 % – test accuracy: 99 %



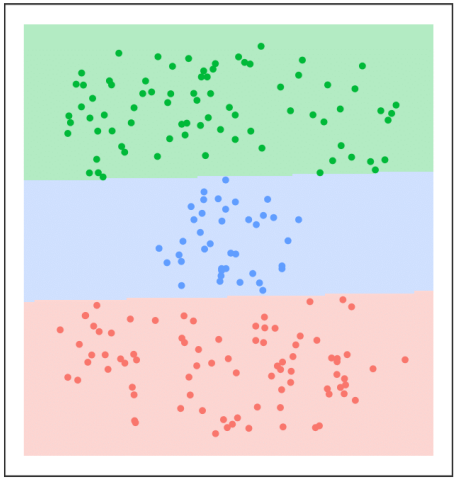
ANN; neurons=10; iters=1000

training accuracy: 100 % – test accuracy: 100 %



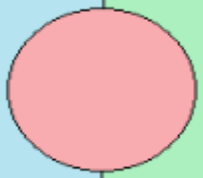
ANN; neurons=10; iters=10000

training accuracy: 100 % – test accuracy: 99.5 %



Spirals

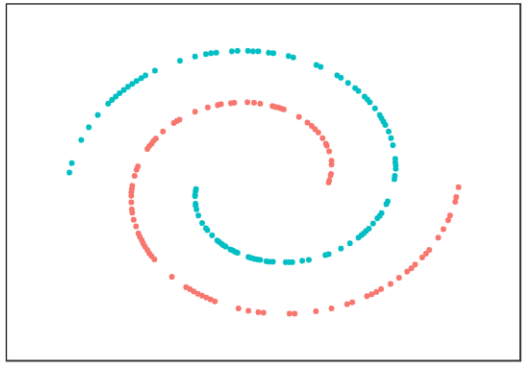
1 cycle



KNN

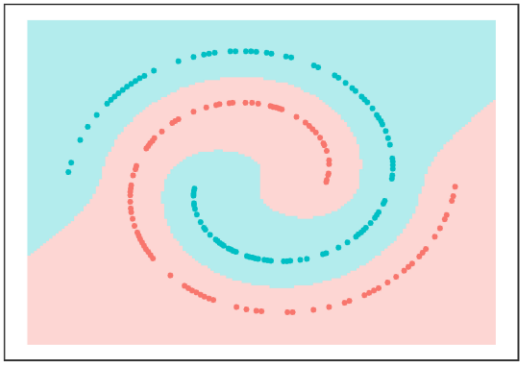
Dataset

mlbench.spirals(n=200, cycles=1, sd=0.0)



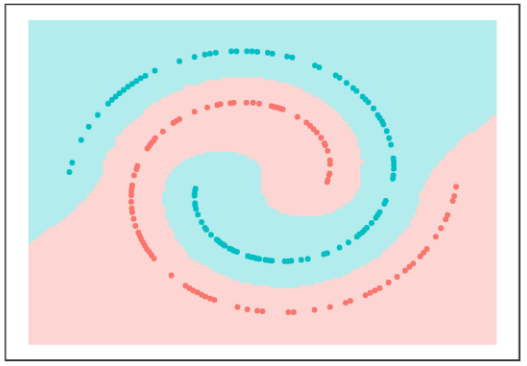
KNN 1

training accuracy: 100 % - test accuracy: 100 %



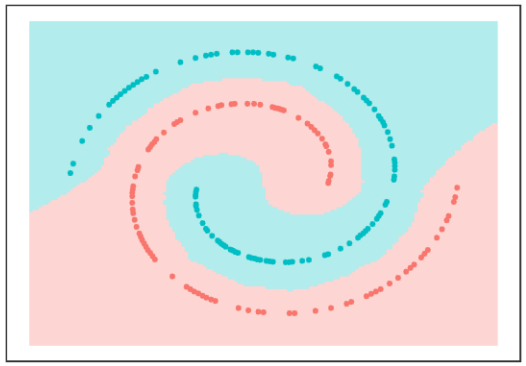
KNN 3

training accuracy: 100 % - test accuracy: 100 %



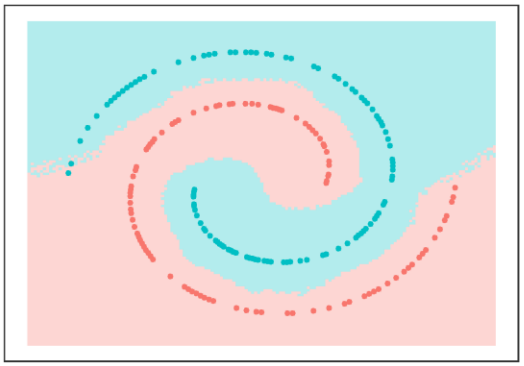
KNN 5

training accuracy: 100 % - test accuracy: 100 %



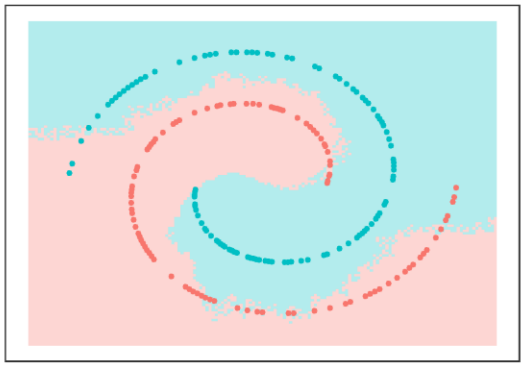
KNN 10

training accuracy: 99.5 % - test accuracy: 99 %



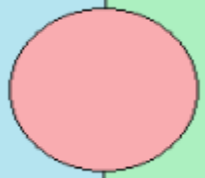
KNN 30

training accuracy: 88.5 % - test accuracy: 85 %

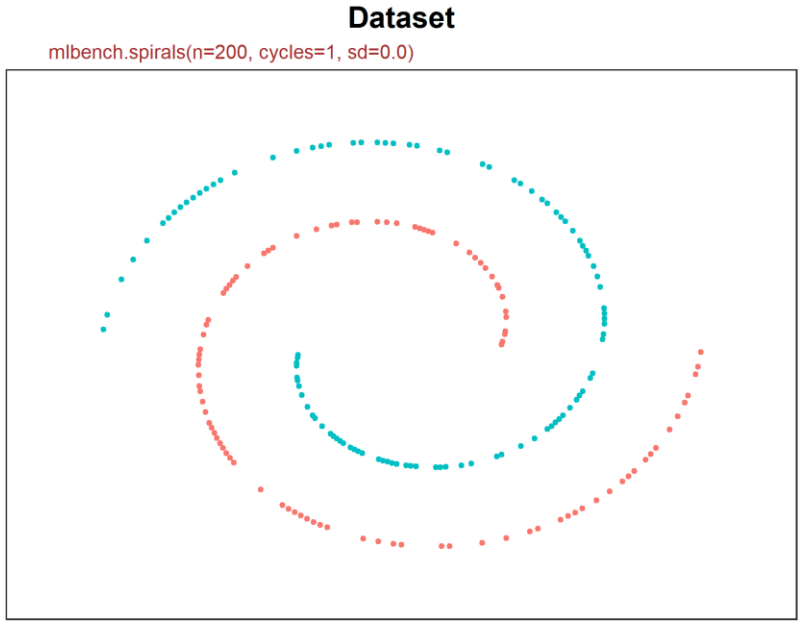


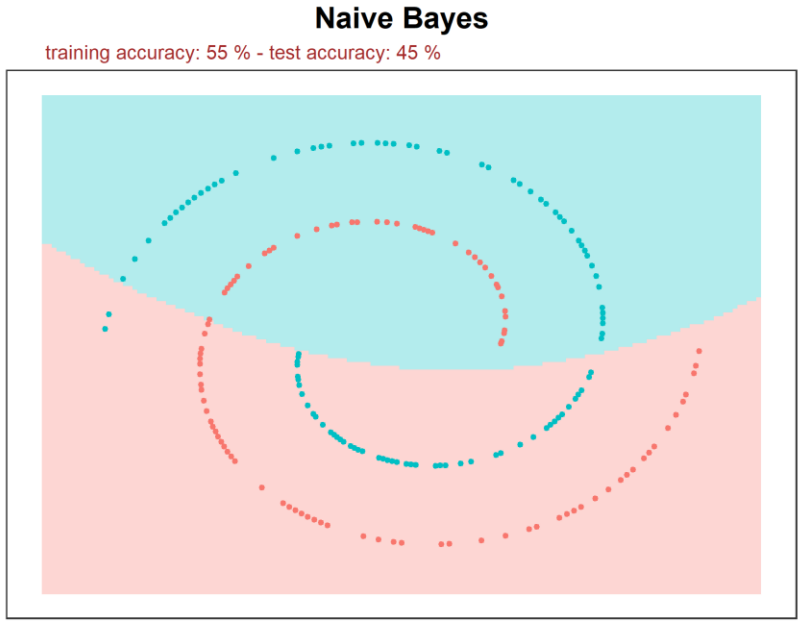
Spirals

1 cycle



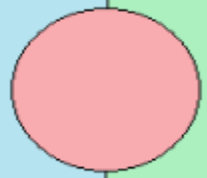
Naïve Bayes





Spirals

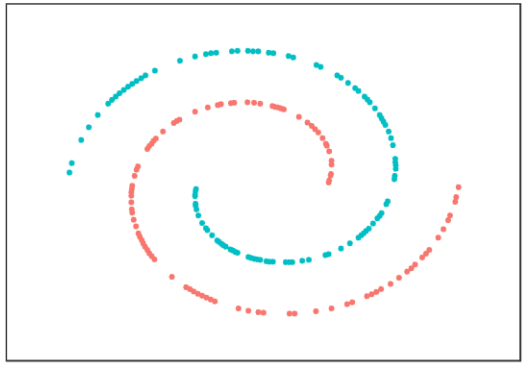
1 cycle



SVM

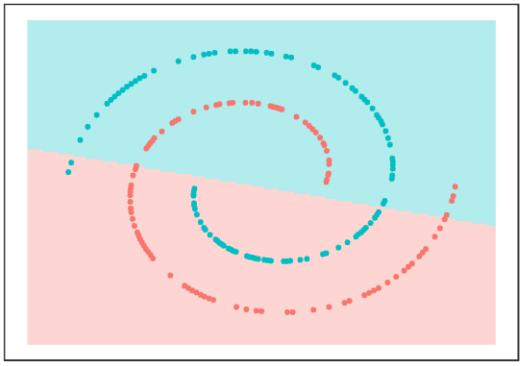
Dataset

mlbench.spirals(n=200, cycles=1, sd=0.0)



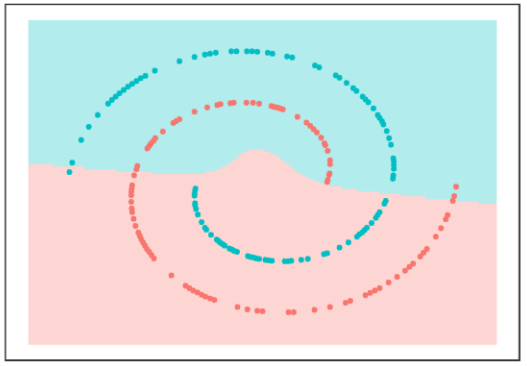
SVM linear

training accuracy: 54.5 % - test accuracy: 46 %



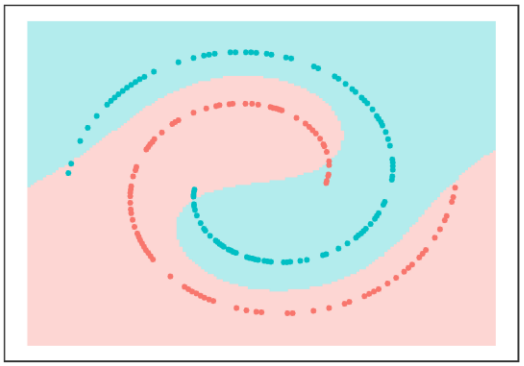
SVM Polynomial

training accuracy: 54 % - test accuracy: 46 %



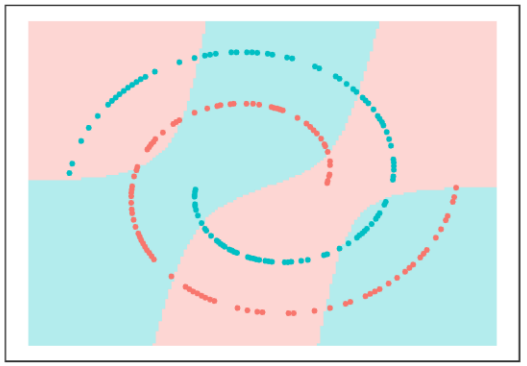
SVM Radial

training accuracy: 94 % - test accuracy: 92.5 %



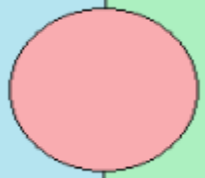
SVM Sigmoid

training accuracy: 36 % - test accuracy: 39 %



Spirals

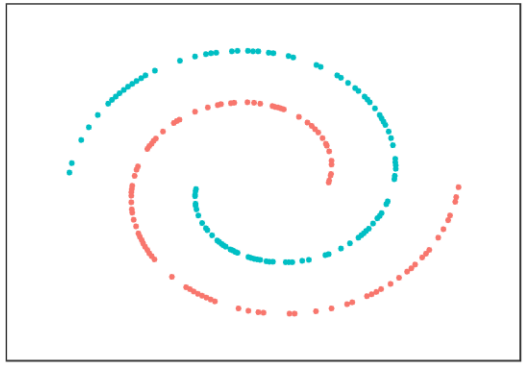
1 cycle



Trees

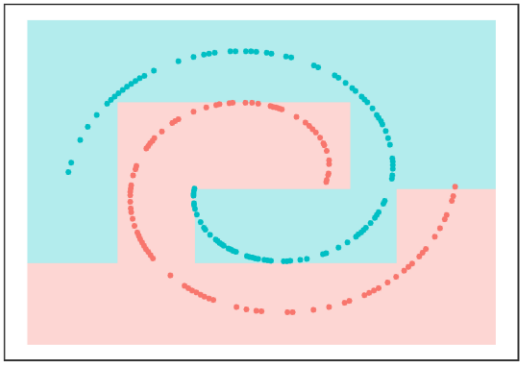
Dataset

mlbench.spirals(n=200, cycles=1, sd=0.0)



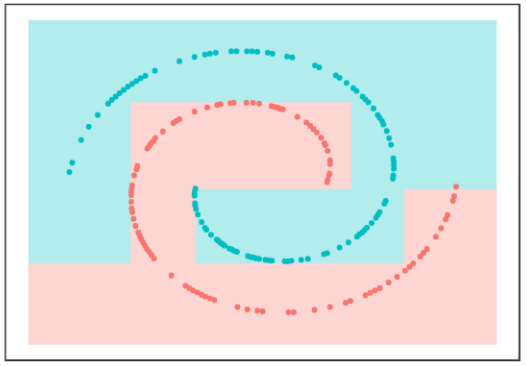
Trees C.50

training accuracy: 99.5 % - test accuracy: 97 %



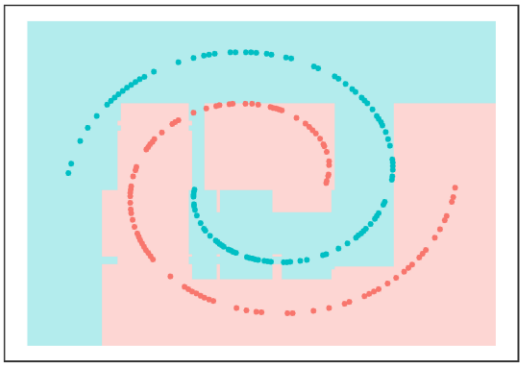
Trees CART

training accuracy: 99 % - test accuracy: 96.5 %



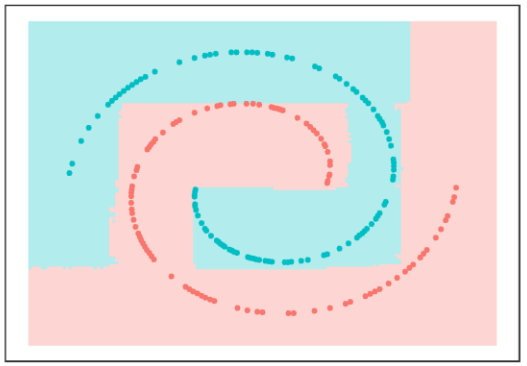
Trees Random Forest 3 trees

training accuracy: 99 % - test accuracy: 96.5 %



Trees Random Forest 128 trees

training accuracy: 100 % - test accuracy: 98.5 %



Spirals

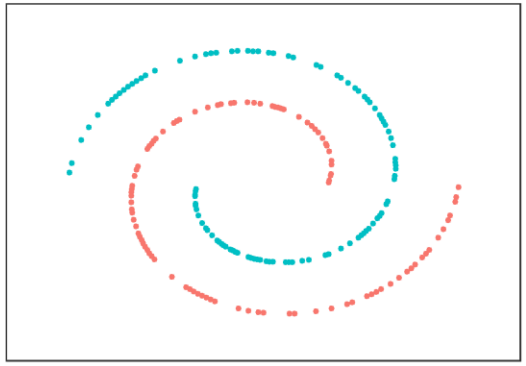
1 cycle

ANN

fixed iterations

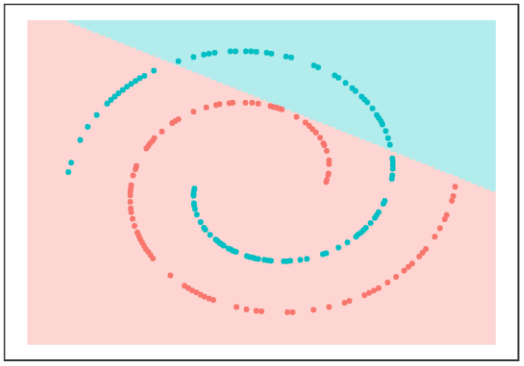
Dataset

mlbench.spirals(n=200, cycles=1, sd=0.0)



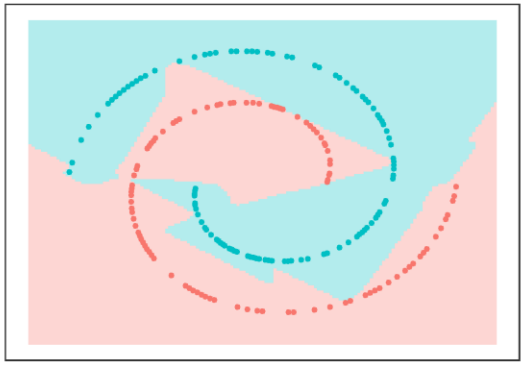
ANN_iters=100_neurons=1

training accuracy: 64.5 % - test accuracy: 64.5 %



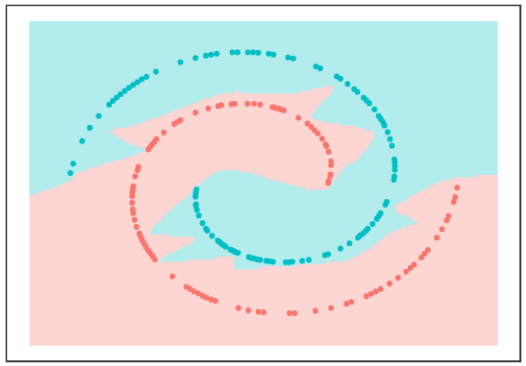
ANN_iters=100_neurons=10

training accuracy: 100 % - test accuracy: 97.5 %



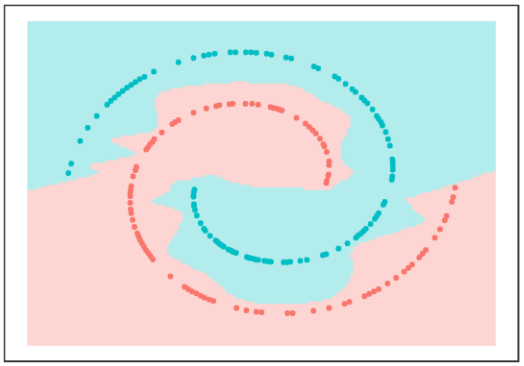
ANN_iters=100_neurons=100

training accuracy: 100 % - test accuracy: 99 %



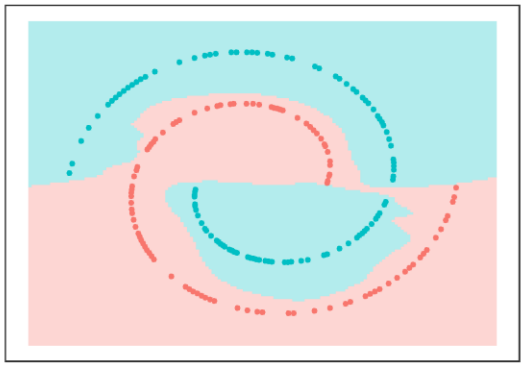
ANN_iters=100_neurons=500

training accuracy: 100 % - test accuracy: 100 %



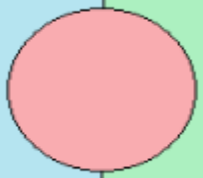
ANN_iters=100_neurons=1000

training accuracy: 100 % - test accuracy: 98.5 %



Spirals

1 cycle

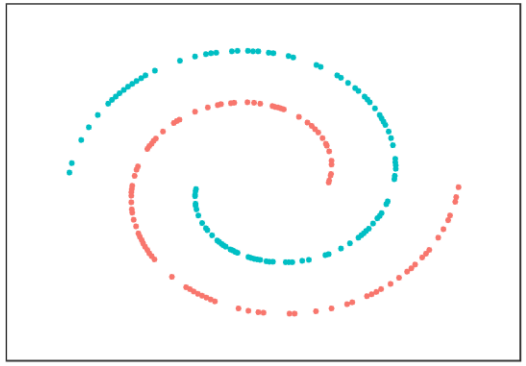


ANN

fixed neurons

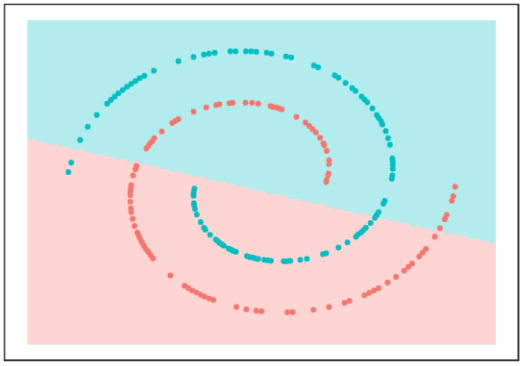
Dataset

mlbench.spirals(n=200, cycles=1, sd=0.0)



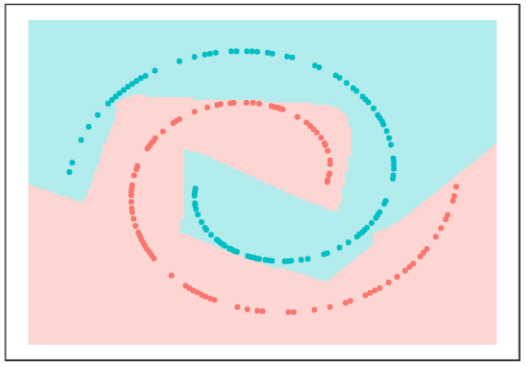
ANN_neurons=10_iters=10

training accuracy: 54.5 % - test accuracy: 46 %



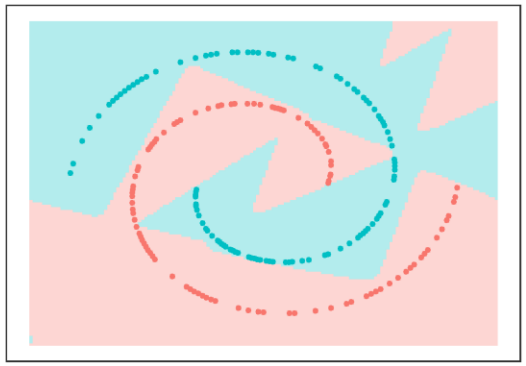
ANN_neurons=10_iters=100

training accuracy: 100 % - test accuracy: 100 %



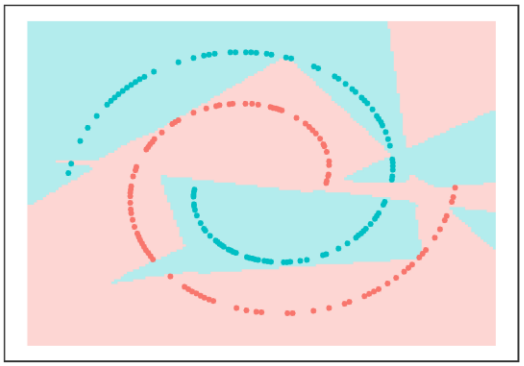
ANN_neurons=10_iters=500

training accuracy: 100 % - test accuracy: 99.5 %



ANN_neurons=10_iters=1000

training accuracy: 100 % - test accuracy: 95 %



ANN_neurons=10_iters=10000

training accuracy: 100 % - test accuracy: 99.5 %