Package ‘PDEnaiveBayes’

July 19, 2026

Type Package

Title Plausible Naive Bayes Classifier Using PDE

Version 0.4.0

Date 2026-07-19

Maintainer Michael Thrun <m.thrun@gmx.net>

Description Provides a nonparametric, multicore-capable plausible naive Bayes classifier based on Pareto density estimation (PDE). It addresses low-evidence cases through a plausibility correction. To enhance the interpretability of the flexible naive Bayes classifier by revealing its posterior structure and feature-wise, class-specific evidence, posterior probabilities can be visualized as class-wise line plots for one-dimensional data or color-coded Voronoi diagrams for pairwise feature projections, and class-conditional PDE likelihoods as overlaid, mirrored density profiles resembling violin plots. Methodological details are provided by Stier, Q., Hoffmann, J. and Thrun, M. C. (2026) ``Classifying with the Fine Structure of Distributions: Leveraging Distributional Information for Robust and Plausible Naive Bayes" <DOI:10.3390/make8010013>. For multicore computations, the implementation applies the general memory-sharing approach described by Thrun, M. C. and Märte, J. (2026) ``memshare: Memory Sharing for Multicore Computation in R with an Application to Feature Selection by Mutual Information using PDE" <DOI:10.32614/RJ-2025-043>.

LazyLoad yes

LazyData TRUE

NeedsCompilation yes

Imports Rcpp (>= 1.0.8), RcppParallel (>= 5.1.4), pracma, plotly, utils, grDevices, stats, graphics, methods, ggplot2, DatabionicSwarm, memshare

Suggests FCPS(>= 1.3.5), ABCanalysis, modeest, deldir, ScatterDensity (>= 0.1.1), gridExtra, parallelDist, parallel, DataVisualizations (>= 1.1.5), knitr

VignetteBuilder knitr

LinkingTo Rcpp, RcppParallel

Depends R (>= 2.10)

Encoding UTF-8

BugReports https://github.com/Mthrun/PDEbayes/issues

Author Michael Thrun [aut, cph, cre] (ORCID:
<https://orcid.org/0000-0001-9542-5543>),
Quirin Stier [aut, rev] (ORCID:
<https://orcid.org/0000-0002-7896-4737>),
Tim Robin Neldner [ctr, ctb]

Repository CRAN

Date/Publication 2026-07-19 18:00:19 UTC

Contents

PDEnaiveBayes-package *Plausible Naive Bayes Classifier Using PDE*

Description

Provides a nonparametric, multicore-capable plausible naive Bayes classifier based on Pareto density estimation (PDE). It addresses low-evidence cases through a plausibility correction. To enhance the interpretability of the flexible naive Bayes classifier by revealing its posterior structure and feature-wise, class-specific evidence, posterior probabilities can be visualized as class-wise line plots for one-dimensional data or color-coded Voronoi diagrams for pairwise feature projections, and class-conditional PDE likelihoods as overlaid, mirrored density profiles resembling violin plots. Methodological details are provided by Stier, Q., Hoffmann, J. and Thrun, M. C. (2026) "Classifying with the Fine Structure of Distributions: Leveraging Distributional Information for Robust and Plausible Naive Bayes" <DOI:10.3390/make8010013>. For multicore computations, the implementation applies the general memory-sharing approach described by Thrun, M. C. and Märte, J. (2026) "memshare: Memory Sharing for Multicore Computation in R with an Application to Feature Selection by Mutual Information using PDE" <DOI:10.32614/RJ-2025-043>.

Details

Pareto Density Estimated naive Bayes Classifier Index of help topics:

```
ApplyBayesTheorem4Likelihoods
                        ApplyBayesTheorem4Likelihoods
GetLikelihoods          GetLikelihoods
Hepta                   Hepta introduced in [Ultsch, 2003]
PDEnaiveBayes-package   Plausible Naive Bayes Classifier Using PDE
PlotBayesianDecision2D
                        PlotBayesianDecision2D
PlotLikelihoodFuns      PlotLikelihoodFuns
PlotLikelihoods         PlotLikelihoods
PlotNaiveBayes          PlotNaiveBayes
PlotPosteriors          PlotPosteriors
Predict_naiveBayes      Predict_naiveBayes
Train_naiveBayes        Train_naiveBayes
Train_naiveBayes_multicore
                        Train a Multicore Pareto Density Naive Bayes
                        Classifier
defineOrEstimateDistribution
                        defineOrEstimateDistribution
fitParameters           fitParameters
getPriors               getPriors
predict.PDEbayes        predict.PDEbayes
```

(PDENB) of [Stier et al., 2026].

Author(s)

Michal Thrun

Maintainer: Michael Thrun <mthrun@informatik.uni-marburg.de>

References

[Stier et al., 2026] Stier, Q.,Hoffmann, J. & Thrun, M. C.: Classifying with the Fine Structure of Distributions: Leveraging Distributional Information for Robust and Plausible Naïve Bayes, Machine Learning and Knowledge Extraction (MAKE), Vol. 8(1), 13, doi 10.3390/make8010013, MDPI, 2026.

[Thrun et al., 2020] Thrun, M. C., Gehlert, T., & Ultsch, A.: Analyzing the Fine Structure of Distributions, PloS one, Vol. 15(10), pp. e0238835, doi 10.1371/journal.pone.0238835 2020.

[Thrun/Ultsch, 2020] Thrun, M. C., & Ultsch, A.: Clustering Benchmark Datasets Exploiting the Fundamental Clustering Problems, Data in Brief, Vol. 30(C), pp. 105501, doi 10.1016/j.dib.2020.105501, 2020.

[Ultsch et al., 2015] Ultsch, A., Thrun, M. C., Hansen-Goos, O., & L?tsch, J.: Identification of Molecular Fingerprints in Human Heat Pain Thresholds by Use of an Interactive Mixture Model R Toolbox (AdaptGauss), International journal of molecular sciences, Vol. 16(10), pp. 25897-25911, doi 10.3390/ijms161025897, 2015.

Examples

```
if(requireNamespace("FCPS")){
V=FCPS::ClusterChallenge("Hepta",1000)
Data=V$Hepta
Cls=V$Cls
ind=1:length(Cls)
indtrain=sample(ind,800)
indtest=setdiff(ind,indtrain)
#parametric
#model=Train_naiveBayes(Data[indtrain,],Cls[indtrain],Gaussian=TRUE)
#ClsTrain=model$ClsTrain
#table(Cls[indtrain],ClsTrain)

#res=Predict_naiveBayes(Data[indtest,], Model = model)
#table(Cls[indtest],res$ClsTest)

#PDEbayes
model=Train_naiveBayes(Data[indtrain,],Cls[indtrain],Gaussian=FALSE)
ClsTrain=model$ClsTrain
table(Cls[indtrain],ClsTrain)

res=Predict_naiveBayes(Data[indtest,], Model = model)
table(Cls[indtest],res$ClsTest)
}
```

ApplyBayesTheorem4Likelihoods

ApplyBayesTheorem4Likelihoods

Description

Calculates the posteriors, for given likelihoods and priors using the Bayes Theorem

Usage

```
ApplyBayesTheorem4Likelihoods(Likelihoods,Priors,threshold=.Machine$double.eps*1000)
```

Arguments

`Likelihoods`	List of d numeric matrices, one per feature, each matrix with 1:k columns containing the distribution of class 1:k.
`Priors`	[1:k] Numeric vector with prior probability for each class.
`threshold`	(Optional: Default=0.00001).

Value

`Posteriors` [1:n, 1:d] Numeric matrix with posterior probability according to the bayes theorem.

Author(s)

Michael Thrun

References

[Stier et al., 2026] Stier, Q.,Hoffmann, J. & Thrun, M. C.: Classifying with the Fine Structure of Distributions: Leveraging Distributional Information for Robust and Plausible Naïve Bayes, Machine Learning and Knowledge Extraction (MAKE), Vol. 8(1), 13, doi 10.3390/make8010013, MDPI, 2026.

Examples

```
if(requireNamespace("FCPS")){
  data(Hepta)
  Data=Hepta$Data
  Cls=Hepta$Cls
  #parametric
  #V=Train_naiveBayes(Data,Cls,Gaussian=TRUE)
  #ClsTrain=V$ClsTrain
  #table(Cls,ClsTrain)

  #non-parametric
  V=Train_naiveBayes(Data,Cls,Gaussian=FALSE)
  ClsTrain=V$ClsTrain
  table(Cls,ClsTrain)
}
```

`defineOrEstimateDistribution`

defineOrEstimateDistribution

Description

The function estimates the distribution of values within a features that belong to a specific class, i.e., the conditional probability of the likelihood

Usage

```
defineOrEstimateDistribution(Feature,ClassInd,Gaussian=FALSE,ParetoRadius=NULL,
InternalPlotIt=FALSE,SD_Threshold=0.001,...)
```

Arguments

`Feature`	[1:n] Numeric Vector
`ClassInd`	Integer Vector with class indices
`Gaussian`	(Optional: Default=TRUE). Assume gaussian distribution.
`ParetoRadius`	Optional [1:d] numerical vector for pareto radii computed priorly, see `ParetoRadius`
`InternalPlotIt`	Optional: Default=FALSE). Create plot if set to TRUE.
`SD_Threshold`	Optional: Default=0.001.
`...`	`Robust`: Optional: Default=FALSE, TRUE: robust estimation of mean and std in case of `Gaussian=TRUE` `Type`: (Optional: Default=2, 1=original PDE, 2= improved PDE `na.rm`: (Optional: Default=TRUE). Remove na.

Value

`Kernels`	[1:m] Numeric vector with kernels (x-values) of a 1D pdf.
`PDF`	[1:m] Numeric vector with the distribution values of a 1D pdf.
`Theta`	Numeric vector with parameters of gaussian of mean and standard deviation - NULL if no gaussian used.

Author(s)

Michael Thrun

References

[Stier et al., 2026] Stier, Q.,Hoffmann, J. & Thrun, M. C.: Classifying with the Fine Structure of Distributions: Leveraging Distributional Information for Robust and Plausible Naïve Bayes, Machine Learning and Knowledge Extraction (MAKE), Vol. 8(1), 13, doi 10.3390/make8010013, MDPI, 2026.

Examples

```
if(requireNamespace("FCPS")){
data(Hepta)
Data=Hepta$Data
Cls=Hepta$Cls
Priors=getPriors(Cls)
}
```

fitParameters *fitParameters*

Description

Fit Gaussian parameters.

Usage

```
fitParameters(Feature,ClassInd,Robust=FALSE,na.rm=TRUE,SD_Threshold=0.0001)
```

Arguments

`Feature`	[1:n] Numeric Vector
`ClassInd`	Integer Vector with class indices
`Robust`	(Optional: Default=FALSE). Robust computation if set to TRUE.
`na.rm`	(Optional: Default=TRUE). Remove na.
`SD_Threshold`	(Optional: Default=0.00001).

Value

`Parameters`	[1:2] Numeric vector with Mean and Std.

Author(s)

Michael Thrun

Examples

```
if(requireNamespace("FCPS")){
data(Hepta)
Data=Hepta$Data
Cls=Hepta$Cls
Priors=getPriors(Cls)
}
```

GetLikelihoods *GetLikelihoods*

Description

Yields the likelihoods per feauture and class as values of distribution either defined by Gaussian or estimated form the data using pareto density estimation.

Usage

```
GetLikelihoods(Data,Cls,...)
```

Arguments

`Data`	[1:n,1:d] matrix of training data. It consists of n cases of d-dimensional data points. Every case has d attributes, variables or features.
`Cls`	[1:n] numerical vector with n numbers defining the classification. It has k unique numbers representing the arbitrary labels of the classification.
`...`	Further arguements for `defineOrEstimateDistribution` Robust=TRUE: robustly estimated gaussians na.rm=TRUE: remove NaNs Threshold: threshold for which the standard deviation cannot be smaller (defaul 0.0001)

Details

Due to pareto density estimation per class and feature, usually the number of rows in each element of `c_Kernels_list` and `ListOfLikelihoods` varies and does not equal the number of rows of data `n`.

Value

`c_Kernels_list`	List of d numeric matrices, one per feature, each matrix with 1:k columns containing the kernels of class 1:k
`ListOfLikelihoods`	List of d numeric matrices, one per feature, each matrix with 1:k columns containing distribution values (likelihood) of class 1:k
`Thetas`	If Gaussian=TRUE: List of d numeric matrices, one per feauture, each matrix with 1:k rows containing the mean in the first column and the standard deviation in teh seconf columd of class 1:k Otherwise: NULL
`ParetoRadiusPerFeauture`	Numeric vector with estimated pareto radius per feature.

Author(s)

Michael Thrun

References

[Stier et al., 2026] Stier, Q.,Hoffmann, J. & Thrun, M. C.: Classifying with the Fine Structure of Distributions: Leveraging Distributional Information for Robust and Plausible Naïve Bayes, Machine Learning and Knowledge Extraction (MAKE), Vol. 8(1), 13, doi 10.3390/make8010013, MDPI, 2026.

Examples

```
if(requireNamespace("FCPS")){
data(Hepta)
Data=Hepta$Data
Cls=Hepta$Cls
Priors=getPriors(Cls)
}
```

---

getPriors *getPriors*

Description

Get a prior via class proportions.

Usage

```
getPriors(Cls)
```

Arguments

`Cls`	[1:n] numerical vector with n numbers defining the classification. It has k unique numbers representing the arbitrary labels of the classification.

Value

`Priors`	[1:k] Numeric vector with prior probability for each class.

Author(s)

Michael Thrun

Examples

```
if(requireNamespace("FCPS")){
data(Hepta)
Data=Hepta$Data
Cls=Hepta$Cls
Priors=getPriors(Cls)
}
```

Hepta *Hepta introduced in [Ultsch, 2003]*

Description

Clearly defined clusters, different variances. Detailed description of dataset and its clustering challenge is provided in [Thrun/Ultsch, 2020].

Usage

```
data("Hepta")
```

Details

Size 212, Dimensions 3, stored in `Hepta$Data`

Classes 7, stored in `Hepta$Cls`

References

[Ultsch, 2003] Ultsch, A.: Maps for the visualization of high-dimensional data spaces, Proc. Workshop on Self organizing Maps (WSOM), pp. 225-230, Kyushu, Japan, 2003.

[Thrun/Ultsch, 2020] Thrun, M. C., & Ultsch, A.: Clustering Benchmark Datasets Exploiting the Fundamental Clustering Problems, Data in Brief, Vol. 30(C), pp. 105501, doi:10.1016/j.dib.2020.105501, 2020.

Examples

```
data(Hepta)
str(Hepta)
```

PlotBayesianDecision2D *PlotBayesianDecision2D*

Description

Plots estimation of decision boundary in a 2D slice of the data using the posteriors

Usage

```
PlotBayesianDecision2D(X, Y, Posteriors, Class = 1, NoBins,
CellColorsOrPallette, Showpoints = TRUE, xlim, ylim, xlab, ylab, main,
PlotIt = TRUE)
```

Arguments

`X`	Numeric vector with point coordinates of first dimension of data selection.
`Y`	Numeric vector with point coordinates of second dimension of data selection.
`Posteriors`	[1:n, 1:Class] matrix of posteriors.
`Class`	Optional,Integer defining which class to look at.
`NoBins`	Optional,Number of bins for class posteriori.
`CellColorsOrPallette`	Optional, Either a function defining the color palette of a character vector or character vector of length NoBins stating colors.
`Showpoints`	Optional, TRUE, points are displayed.
`xlim`	Optional,Numeric vector of length 2 stating limits of x axis.
`ylim`	Optional,Numeric vector of length 2 stating limits of y axis.
`xlab`	Optional,Character stating name of x axis.
`ylab`	Optional,Character stating name of y axis.
`main`	Optional, Character name of title
`PlotIt`	Optional, TRUE: prints GGPLOT2 object, FALSE: not shown plot.

Details

Boundaries are assumed to be zero for plotting.

Value

List of:

`Mapping`	List containing a map for colors, kernels and bin number.
`GGobj`	ggplot2 object containing 2D visualization of Posteriori.

Author(s)

Michael Thrun

References

[Stier et al., 2026] Stier, Q.,Hoffmann, J. & Thrun, M. C.: Classifying with the Fine Structure of Distributions: Leveraging Distributional Information for Robust and Plausible Naïve Bayes, Machine Learning and Knowledge Extraction (MAKE), Vol. 8(1), 13, doi 10.3390/make8010013, MDPI, 2026.

Examples

```
Data = as.matrix(iris[,1:4])
Cls = as.numeric(iris[,5])

TrainIdx = c(17, 73, 46, 29, 68, 35, 131, 62, 132, 127, 71, 72,
144, 99, 93, 13, 38, 21, 102, 53, 36, 111, 114, 96, 57, 74, 145,
86, 3, 16, 52, 59, 140, 40, 122, 109, 6, 91, 79, 15, 108, 139,
37, 76, 20, 115, 66, 28, 100, 117, 44, 78, 80, 150, 146, 142,
9, 90, 45, 58, 134, 11, 87, 125, 141, 118, 136, 48, 124, 47,
8, 27, 33, 92, 130, 54, 65, 104, 23, 98, 129, 123, 34, 128, 135,
51, 64, 5, 94, 83, 42, 116, 101, 43, 7, 12, 82, 1, 84, 138, 2,
56, 4, 106, 120)

TestIdx = c(60, 10, 75, 70, 81, 18, 97, 95, 67, 22, 55, 143,
88, 24, 105, 26, 119, 31, 107, 63, 41, 61, 32, 147, 89, 14, 121,
19, 113, 49, 126, 112, 25, 77, 137, 103, 50, 30, 149, 110, 39,
69, 148, 85, 133)

TrainX = Data[TrainIdx, ]
TestX  = Data[TestIdx, ]
TrainY = Cls[TrainIdx]
TestY  = Cls[TestIdx]

VPDENB = Train_naiveBayes(Data = TrainX, Cls = TrainY, Plausible = FALSE)

PlotBayesianDecision2D(X = TrainX[, 1], Y = TrainX[, 2],
Posteriors = VPDENB$Posteriors, Class = 1)
```

PlotLikelihoodFuns *PlotLikelihoodFuns*

Description

Plots the class-conditional Likelihoods per feature, given the generating likelihood functions.

Usage

```
PlotLikelihoodFuns(LikelihoodFuns,Data,PlausibleLikelihoodFuns=NULL,
Epsilon=NULL,PlausibleCenters=NULL,PlotCutOff=4,xlim)
```

Arguments

- `LikelihoodFuns` List with Likelihoods generating functions
- `Data` Numeric matrix with data.
- `PlausibleLikelihoodFuns` List with plausible Likelihoods.
- `Epsilon` Numeric scalar defining epsilon fo plausible likelihoods.

PlausibleCenters
: Numeric vector [1:k] plausible centers used to compute plausible likelihoods.

PlotCutOff
: scalar defining the how many feature starting from 1 should be plotted or numerical vector defining the index of features to be plotted in second case should not be too many otherwise plot yields an error.

xlim
: Numeric vector of length 2 stating limits of x axis.

Value

No return value.

Author(s)

Michael Thrun

References

[Stier et al., 2026] Stier, Q.,Hoffmann, J. & Thrun, M. C.: Classifying with the Fine Structure of Distributions: Leveraging Distributional Information for Robust and Plausible Naïve Bayes, Machine Learning and Knowledge Extraction (MAKE), Vol. 8(1), 13, doi 10.3390/make8010013, MDPI, 2026.

Examples

```
Data = as.matrix(iris[,1:4])
Cls = as.numeric(iris[,5])

TrainIdx = c(17, 73, 46, 29, 68, 35, 131, 62, 132, 127, 71, 72,
144, 99, 93, 13, 38, 21, 102, 53, 36, 111, 114, 96, 57, 74, 145,
86, 3, 16, 52, 59, 140, 40, 122, 109, 6, 91, 79, 15, 108, 139,
37, 76, 20, 115, 66, 28, 100, 117, 44, 78, 80, 150, 146, 142,
9, 90, 45, 58, 134, 11, 87, 125, 141, 118, 136, 48, 124, 47,
8, 27, 33, 92, 130, 54, 65, 104, 23, 98, 129, 123, 34, 128, 135,
51, 64, 5, 94, 83, 42, 116, 101, 43, 7, 12, 82, 1, 84, 138, 2,
56, 4, 106, 120)

TestIdx = c(60, 10, 75, 70, 81, 18, 97, 95, 67, 22, 55, 143,
88, 24, 105, 26, 119, 31, 107, 63, 41, 61, 32, 147, 89, 14, 121,
19, 113, 49, 126, 112, 25, 77, 137, 103, 50, 30, 149, 110, 39,
69, 148, 85, 133)

TrainX = Data[TrainIdx, ]
TestX  = Data[TestIdx, ]
TrainY = Cls[TrainIdx]
TestY  = Cls[TestIdx]

VPDENB = Train_naiveBayes(Data = TrainX, Cls = TrainY, Plausible = FALSE)

PlotLikelihoodFuns(LikelihoodFuns = VPDENB$Model$PDFs_funs, Data = TrainX)
```

`PlotLikelihoods` *PlotLikelihoods*

Description

Plots the Likelihoods per feature.

Usage

```
PlotLikelihoods(Likelihoods, Data, PlausibleLikelihoods=NULL,Epsilon=NULL,
PlausibleCenters=NULL,PlotCutOff=4,xlim)
```

Arguments

`Likelihoods`	List with Likelihoods.
`Data`	Numeric matrix with data.
`PlausibleLikelihoods`	List with plausible Likelihoods.
`Epsilon`	Numeric scalar defining epsilon fo plausible likelihoods.
`PlausibleCenters`	Numeric vector [1:k] plausible centers used to compute plausible likelihoods.
`PlotCutOff`	scalar defining the how many feature starting from 1 should be plotted or numerical vector defining the index of features to be plotted in second case should not be too many otherwise plot yields an error.
`xlim`	Numeric vector of length 2 stating limits of x axis.

Details

Boundaries are assumed to be zero for plotting.

Value

No return value.

Author(s)

Michael Thrun

References

[Stier et al., 2026] Stier, Q.,Hoffmann, J. & Thrun, M. C.: Classifying with the Fine Structure of Distributions: Leveraging Distributional Information for Robust and Plausible Naïve Bayes, Machine Learning and Knowledge Extraction (MAKE), Vol. 8(1), 13, doi 10.3390/make8010013, MDPI, 2026.

Examples

```
Data = as.matrix(iris[,1:4])
Cls = as.numeric(iris[,5])

TrainIdx = c(17, 73, 46, 29, 68, 35, 131, 62, 132, 127, 71, 72,
144, 99, 93, 13, 38, 21, 102, 53, 36, 111, 114, 96, 57, 74, 145,
86, 3, 16, 52, 59, 140, 40, 122, 109, 6, 91, 79, 15, 108, 139,
37, 76, 20, 115, 66, 28, 100, 117, 44, 78, 80, 150, 146, 142,
9, 90, 45, 58, 134, 11, 87, 125, 141, 118, 136, 48, 124, 47,
8, 27, 33, 92, 130, 54, 65, 104, 23, 98, 129, 123, 34, 128, 135,
51, 64, 5, 94, 83, 42, 116, 101, 43, 7, 12, 82, 1, 84, 138, 2,
56, 4, 106, 120)

TestIdx = c(60, 10, 75, 70, 81, 18, 97, 95, 67, 22, 55, 143,
88, 24, 105, 26, 119, 31, 107, 63, 41, 61, 32, 147, 89, 14, 121,
19, 113, 49, 126, 112, 25, 77, 137, 103, 50, 30, 149, 110, 39,
69, 148, 85, 133)

TrainX = Data[TrainIdx, ]
TestX  = Data[TestIdx, ]
TrainY = Cls[TrainIdx]
TestY  = Cls[TestIdx]

VPDENB = Train_naiveBayes(Data = TrainX, Cls = TrainY, Plausible = FALSE)

PlotLikelihoods(Likelihoods = VPDENB$Model$ListOfLikelihoods, Data = TrainX)
```

PlotNaiveBayes *PlotNaiveBayes*

Description

Visualize the class-conditional distributions of the Pareto Density estimated naive Bayes model (PDENB) [Stier et al., 2026].

Usage

```
PlotNaiveBayes(Model, FeatureNames, ClassNames, DatasetName = "Data",
nrows = 1, FeatureOrderOrSubset, NumFeaturesPerRow = 4, Colors,
IndividualFigures = FALSE)
```

Arguments

Model	List with elements Priors,c_2List_Train.
FeatureNames	Character vector of names with a name for each feature contained in the data used to create the naive bayes model.
ClassNames	Character vector of class names to present in the legend of the plots.
DatasetName	Character title for each plot.

`nrows` Number of rows inside one plot.

`FeatureOrderOrSubset`
: Numeric vector representing the order of the features to be displayed or a subset as col indices om data column order.

`NumFeaturesPerRow`
: Maximum number of features to be displayed in one plot.

`Colors` Character vector of color names. The length of the vector must be the same as the number of classes within the data modeled by the naive Bayes classifier.

`IndividualFigures`
: Optional boolean: If set to TRUE, it returns a list of the individual figures for customization.

Details

Boundaries are assumed to be zero for plotting.

Value

`Cls` [1:n] numerical vector with n numbers defining the classification. It has k unique numbers representing the arbitrary labels of the classification.

`Posteriors` [1:n, 1:l] Numeric matrices with posterior probabilities.

`DataLikelihoodsPerClass`
: list of length `d`, each element is a matrix `[1:n,1:k]` of interpolated class likelihoods per feature `d`

Author(s)

Quirin Stier

References

[Stier et al., 2026] Stier, Q.,Hoffmann, J. & Thrun, M. C.: Classifying with the Fine Structure of Distributions: Leveraging Distributional Information for Robust and Plausible Naïve Bayes, Machine Learning and Knowledge Extraction (MAKE), Vol. 8(1), 13, doi 10.3390/make8010013, MDPI, 2026.

Examples

```
Data = as.matrix(iris[,1:4])
Cls = as.numeric(iris[,5])
DatasetName = "Iris"

TrainIdx = c(17, 73, 46, 29, 68, 35, 131, 62, 132, 127, 71, 72,
144, 99, 93, 13, 38, 21, 102, 53, 36, 111, 114, 96, 57, 74, 145,
86, 3, 16, 52, 59, 140, 40, 122, 109, 6, 91, 79, 15, 108, 139,
37, 76, 20, 115, 66, 28, 100, 117, 44, 78, 80, 150, 146, 142,
9, 90, 45, 58, 134, 11, 87, 125, 141, 118, 136, 48, 124, 47,
8, 27, 33, 92, 130, 54, 65, 104, 23, 98, 129, 123, 34, 128, 135,
51, 64, 5, 94, 83, 42, 116, 101, 43, 7, 12, 82, 1, 84, 138, 2,
```

```
56, 4, 106, 120)

TestIdx = c(60, 10, 75, 70, 81, 18, 97, 95, 67, 22, 55, 143,
88, 24, 105, 26, 119, 31, 107, 63, 41, 61, 32, 147, 89, 14, 121,
19, 113, 49, 126, 112, 25, 77, 137, 103, 50, 30, 149, 110, 39,
69, 148, 85, 133)

TrainX = Data[TrainIdx, ]
TestX  = Data[TestIdx, ]
TrainY = Cls[TrainIdx]
TestY  = Cls[TestIdx]

VPDENB = Train_naiveBayes(Data = TrainX, Cls = TrainY, Plausible = FALSE)

FeatureNames = colnames(Data)

PlotNaiveBayes(Model = VPDENB$Model, FeatureNames = FeatureNames)
```

`PlotPosteriors` *PlotPosteriors*

Description

Plots posteriors either using a panel of plots based on PlotBayesianDecision2D or in 1D as a line plot [Stier et al., 2026].

Usage

```
PlotPosteriors(Data, Posteriors, Class = 1,
CellColorsOrPallette,Showpoints=TRUE,NoBins,ShowLegend=TRUE)
```

Arguments

`Data`	Either numeric matrix [1:n, 1:d] with data or one column of data.
`Posteriors`	[1:n, 1:Class] matrix of posteriors.
`Class`	Integer defining which class to look at if numeric matrix is given, for column of data all posteriors are overlayed in line plot.
`CellColorsOrPallette`	Optional, Either a function defining the color palette of a character vector or character vector of length NoBins stating colors.
`Showpoints`	Optional, TRUE, points are displayed.
`NoBins`	Optional, number of bins for class posteriori
`ShowLegend`	Optional, TRUE, show one posterior legend for all pairwise plots

Details

Plotting posteriors in one directions only often does not give any insight. The default option using PlotBayesianDecision2D os often more useful.

Value

`GGobj` ggplot2 object containing 2D visualization of Posteriori.

Author(s)

Michael Thrun

References

[Stier et al., 2026] Stier, Q.,Hoffmann, J. & Thrun, M. C.: Classifying with the Fine Structure of Distributions: Leveraging Distributional Information for Robust and Plausible Naïve Bayes, Machine Learning and Knowledge Extraction (MAKE), Vol. 8(1), 13, doi 10.3390/make8010013, MDPI, 2026.

Examples

```
Data = as.matrix(iris[,1:4])
Cls = as.numeric(iris[,5])

TrainIdx = c(17, 73, 46, 29, 68, 35, 131, 62, 132, 127, 71, 72,
144, 99, 93, 13, 38, 21, 102, 53, 36, 111, 114, 96, 57, 74, 145,
86, 3, 16, 52, 59, 140, 40, 122, 109, 6, 91, 79, 15, 108, 139,
37, 76, 20, 115, 66, 28, 100, 117, 44, 78, 80, 150, 146, 142,
9, 90, 45, 58, 134, 11, 87, 125, 141, 118, 136, 48, 124, 47,
8, 27, 33, 92, 130, 54, 65, 104, 23, 98, 129, 123, 34, 128, 135,
51, 64, 5, 94, 83, 42, 116, 101, 43, 7, 12, 82, 1, 84, 138, 2,
56, 4, 106, 120)

TestIdx = c(60, 10, 75, 70, 81, 18, 97, 95, 67, 22, 55, 143,
88, 24, 105, 26, 119, 31, 107, 63, 41, 61, 32, 147, 89, 14, 121,
19, 113, 49, 126, 112, 25, 77, 137, 103, 50, 30, 149, 110, 39,
69, 148, 85, 133)

TrainX = Data[TrainIdx, ]
TestX  = Data[TestIdx, ]
TrainY = Cls[TrainIdx]
TestY  = Cls[TestIdx]

VPDENB = Train_naiveBayes(Data = TrainX, Cls = TrainY, Plausible = FALSE)
#default option
PlotPosteriors(Data = TrainX, Posteriors = VPDENB$Posteriors, Class = 1)

# alternative option
PlotPosteriors(Data = TrainX[,3], Posteriors = VPDENB$Posteriors)
```

`predict.PDEbayes` *predict.PDEbayes*

Description

Predict a classification with the Pareto Density estimated naive Bayes model [Stier et al., 2026] . (PDENB).

Usage

```
predict.PDEbayes(object, newdata, type = c("class", "response","prob"), ...)
```

Arguments

`object`	Model obtained from training routine in PDEnaiveBayes package.
`newdata`	[1:n,1:d] matrix of test data. It consists of n cases of d-dimensional data points. Every case has d attributes, variables or features.
`type`	Optional parameter.
`...`	`Gaussian`: Optional: Default=TRUE). Assume gaussian distribution. `Plausible`: (Optional: TRUE: uses plausble bayesian theorem, FALSE non-plausible bayesian theorem `Type`: (Optional: default=1, 1 = original PDE, 2 = R native density estimation `Threshold`: Threshold for which the standard deviation cannot be smaller (default =1e-12) `PlotIt`: Optional: Default=FALSE, TRUE: Plots Likelihoods `PlotCutOff`: Optional: Scalar indicating how many features (starting from 1) should be plotted, or a numerical vector specifying the indices of the features to plot. Note: In the second case, avoid selecting too many features, as this may cause the plot to fail `ParetoRadiusPerFeauture`: Optional [1:d] numerical vector for pareto radii computed priorly, see `ParetoRadius` or `{ParetoRadius_fast}` `cl`: Optional: a cluster object, created by parallel, if given and `ParetoRadiusPerFeauture` missing, then `ParetoRadiusPerFeauture` is compputed multicore otherwise single core `Robust`: Optional: Default=FALSE, TRUE: robust estimation of mean and std in case of `Gaussian=TRUE`

Details

The function is implemented in a way so that one can combine training and test data although it is intended to be applied on test data only.

Value

`Cls`	Numeric vector with predicted class associated with newdata.

Author(s)

Michael Thrun

References

[Stier et al., 2026] Stier, Q.,Hoffmann, J. & Thrun, M. C.: Classifying with the Fine Structure of Distributions: Leveraging Distributional Information for Robust and Plausible Naïve Bayes, Machine Learning and Knowledge Extraction (MAKE), Vol. 8(1), 13, doi 10.3390/make8010013, MDPI, 2026.

See Also

`Train_naiveBayes`

Examples

```
if(requireNamespace("FCPS")){
V=FCPS::ClusterChallenge("Hepta",1000)
Data=V$Hepta
Cls=V$Cls
ind=1:length(Cls)
indtrain=sample(ind,800)
indtest=setdiff(ind,indtrain)

model=Train_naiveBayes(Data[indtrain,],Cls[indtrain],Gaussian=FALSE)
ClsTrain=model$ClsTrain
table(Cls[indtrain],ClsTrain)

ClsTest=predict.PDEbayes(object = model, newdata = Data[indtest,])
table(Cls[indtest],ClsTest)
}
```

`Predict_naiveBayes` *Predict_naiveBayes*

Description

Predict classification with naive Bayes model [Stier et al., 2026].

Usage

```
Predict_naiveBayes(Data, Model, ...)
```

Arguments

`Data`	[1:n,1:d] matrix of test data. It consists of n cases of d-dimensional data points. Every case has d attributes, variables or features.
`Model`	Optional, list with elements `Priors`,`c_2List_Train`,`Thetas`, alternative set arguements seperatly

... Priors: Optional, if Model missing, then [1:k] Numeric vector with prior probability for each class.

c_2List_Train: Optional, if Model missing, then c_2List_Train is the output of GetLikelihoods: a list of two three elements of Kernels, Likelihoods per feature and class, optional Thetas or PlausibleCenters depending on parameter setting

Thetas: Optional, if Model missing, then If c_2List_Train is missing, alternatively the parameters mean and standard deviation of the gaussian distributions per class and feaures.

PlotIt: Optional: Default=FALSE, TRUE: Plots Likelihoods

PlotCutOff: Optional: Scalar indicating how many features (starting from 1) should be plotted, or a numerical vector specifying the indices of the features to plot. Note: In the second case, avoid selecting too many features, as this may cause the plot to fail

Details

The function is implemented in a way so that one can combine training and test data although it is intended to be applied on test data only.

Value

Cls [1:n] numerical vector with n numbers defining the classification. It has k unique numbers representing the arbitrary labels of the classification.

Posteriors [1:n, 1:l] Numeric matrices with posterior probabilities.

DataLikelihoodsPerClass
list of length d, each element is a matrix [1:n,1:k] of interpolated class likelihoods per feature d

Author(s)

Michael Thrun

References

[Stier et al., 2026] Stier, Q.,Hoffmann, J. & Thrun, M. C.: Classifying with the Fine Structure of Distributions: Leveraging Distributional Information for Robust and Plausible Naïve Bayes, Machine Learning and Knowledge Extraction (MAKE), Vol. 8(1), 13, doi 10.3390/make8010013, MDPI, 2026.

See Also

Train_naiveBayes

Examples

```
if(requireNamespace("FCPS")){
V=FCPS::ClusterChallenge("Hepta",1000)
Data=V$Hepta
Cls=V$Cls
```

```
ind=1:length(Cls)
indtrain=sample(ind,800)
indtest=setdiff(ind,indtrain)

#PDEbayes
model=Train_naiveBayes(Data[indtrain,],Cls[indtrain],Gaussian=FALSE)
ClsTrain=model$ClsTrain
table(Cls[indtrain],ClsTrain)

res=Predict_naiveBayes(Data[indtest,], Model = model)
table(Cls[indtest],res$ClsTest)
}
```

Train_naiveBayes *Train_naiveBayes*

Description

Trains a Pareto Density estimated naive Bayes model (PDENB) of [Stier et al., 2026].

Usage

```
Train_naiveBayes(Data,Cls,Predict=TRUE,Priors,...)
```

Arguments

`Data`	`[1:n,1:d]` matrix of training data. It consists of `n` cases of `d`-dimensional data points. Every case hasd attributes, variables or features.
`Cls`	`[1:n]` numerical vector with `n` numbers defining the classification. It has `k` unique numbers representing the arbitrary labels of the classification.
`Predict`	Optional, boolean to decide extent of output. In case of TRUE, yields ClsTrain and Posteriors, else it yields only Model and Thetas. Note: Only if Predict is set to TRUE, parameter EvalPlausible can be set true!
`Priors`	Optional, `[1:k]` numerical vector defining the prior probabilities of the `k` classes. If missing, estimated from `Cls`.
`...`	`Gaussian`: Optional: Default=TRUE). Assume gaussian distribution. `Plausible`: (Optional: TRUE: uses plausble bayesian theorem, FALSE non-plausible bayesian theorem. `Type`: (Optional: default=1, 1 = original PDE, 2 = R native density estimation. `Threshold`: Threshold for which the standard deviation cannot be smaller (default =1e-12). `PlotIt`: Optional: Default=FALSE, TRUE: Plots Likelihoods. `PlotCutOff`: Optional: Scalar indicating how many features (starting from 1) should be plotted, or a numerical vector specifying the indices of the features to plot. Note: In the second case, avoid selecting too many features, as this may cause the plot to fail.

ParetoRadiusPerFeauture: Optional [1:d] numerical vector for pareto radii computed priorly, see ParetoRadius or {ParetoRadius_fast}

cl: Optional: a cluster object, created by parallel, if given and ParetoRadiusPerFeauture missing, then ParetoRadiusPerFeauture is compputed multicore otherwise single core

Robust: Optional: Default=FALSE, TRUE: robust estimation of mean and std in case of Gaussian=TRUE.

GlobalPR: Optional: Default=TRUE, FALSE: estimation of pareto radius for each class individually.

Details

Precomputation of ParetoRadiusPerFeauture can be usefull to make cross-validation faster although it should be only done on the training data.

If Plausible is not given, both options are evalauted using shannon information.

c_Kernels_list and ListOfLikelihoods have d elements each storing a matrix [1:m,1:k], usually m!=n. In contrast to DataLikelihoodsPerClass in which by interpolation the matrix are of size [1:n,1:k]

Value

Model
: List of model parameters and results.

c_Kernels_list
: List of matrices, where each matrix represent the kernels of one feature for all classes.

ListOfLikelihoods
: List of matrices, where each matrix represent the likelihood of one feature for all classes.

PDFs_funs
: Nested list of depth 1, where the first index assigns the feature index and the second index assigns the class. The elements are functions for the density estimation for each feature and each class.

ParetoRadiusPerFeauture
: Numeric vector which stores the pareto radius for each feature.

Theta
: Parameters mean and standard deviation of the Gaussian distributions per class and feaures.

Priors
: Numeric vector which stores the prior probability of each class to appear.

PlausibleCenters
: [1:k, 1:f] Numeric matrix which stores the centers for each feature and each class, where the row index assigns features and the column index assigns classes.

ClsTrain
: [1:n] numerical vector with n numbers defining the classification. It has k unique numbers representing the arbitrary labels of the classification.

Posteriors
: [1:n, 1:k] Numeric matrices with posterior probabilities.

Author(s)

Michael Thrun

References

[Stier et al., 2026] Stier, Q.,Hoffmann, J. & Thrun, M. C.: Classifying with the Fine Structure of Distributions: Leveraging Distributional Information for Robust and Plausible Naïve Bayes, Machine Learning and Knowledge Extraction (MAKE), Vol. 8(1), 13, doi 10.3390/make8010013, MDPI, 2026.

See Also

Predict_naiveBayes

Examples

```
if(requireNamespace("FCPS")){
data(Hepta)
Data=Hepta$Data
Cls=Hepta$Cls

#non-parametric
V=Train_naiveBayes(Data,Cls,Gaussian=FALSE)
ClsTrain=V$ClsTrain
table(Cls,ClsTrain)
}
```

Train_naiveBayes_multicore
Train a Multicore Pareto Density Naive Bayes Classifier

Description

Trains a naive Bayes classifier using Gaussian likelihoods or nonparametric Pareto density estimation (PDE), with an optional plausible correction for low-evidence cases [Stier et al., 2026]. Model training can be distributed across a parallel cluster. Shared-memory computation can be enabled to prevent memory duplication among workers [Thrun and Märte, 2026].

Usage

```
Train_naiveBayes_multicore(
  cl = NULL,
  Data,
  Cls,
  Plausible = TRUE,
  Predict = FALSE,
  Priors,
  UseMemshare = FALSE,
  ...
)
```

Arguments

`cl`	A cluster object, typically created with `makeCluster`. If `NULL`, the function runs in a single R process; this mode is intended primarily for debugging.
`Data`	A numeric `n` by `d` matrix containing `n` observations in rows and `d` features in columns. Objects that are not matrices are converted with `as.matrix()`.
`Cls`	A numeric vector of length `n` containing the class label for each observation. The vector must contain at least two distinct finite class labels. If `Cls` is not numeric, the function attempts to convert it using **FCPS**; otherwise it falls back to `as.numeric()`.
`Plausible`	Logical. If `TRUE`, use the plausible naive Bayes extension described by Stier et al. (2026) for low-evidence cases. If `FALSE`, use the standard Bayes decision rule. The plausible adjustment is intended for non-Gaussian likelihood models and requires **FCPS**.
`Predict`	Logical. If `TRUE`, classify the training observations after fitting and return their predicted classes and posterior probabilities. If `FALSE`, only the model is fitted; `ClsTrain` and `Posteriors` are returned as `NULL`.
`Priors`	Optional numeric vector of length `k` containing the prior probabilities of the `k` classes. Names should correspond to the class labels. If missing, priors are estimated from the filtered training labels in `Cls`.
`UseMemshare`	Logical. If `TRUE` and `cl` is not `NULL`, use `memApply` so that memory can be shared within the multicore computation. This can reduce duplication of large training objects among workers and requires the **memshare** package. If `FALSE`, the **parallel** backend is used.
`...`	Additional named arguments passed to the underlying likelihood-estimation and prediction functions. Common arguments include:

`Gaussian` Logical; default `FALSE`. If `TRUE`, estimate a Gaussian likelihood for every feature and class. If `FALSE`, estimate likelihoods nonparametrically.

`Type` Integer selecting the nonparametric density estimator. `1` uses Pareto density estimation, `2` uses `stats::density()`, and `3` uses `stats::density()` with the Pareto radius as its bandwidth. The default is `1`.

`Threshold` Numeric threshold used by the plausible-likelihood adjustment. The default in `Train_naiveBayes()` is `1e-4`.

`SD_Threshold` Positive numeric lower bound used for the standard deviation during Gaussian parameter estimation. The default is `0.001`.

`ParetoRadiusPerFeauture` Optional precomputed Pareto radius. The spelling of this argument is retained for API compatibility. The current multicore wrapper supports a scalar value; a feature-specific vector is not supported. A radius must be estimated using training data only, for example with `ParetoRadius` or `ParetoRadius_fast`.

`Robust` Logical; default `FALSE`. If `TRUE` and `Gaussian = TRUE`, use robust estimates of the class-wise means and standard deviations.

`GlobalPR` Logical; default `TRUE`. If `TRUE`, estimate one Pareto radius per feature. If `FALSE`, estimate a separate Pareto radius for each class and feature.

`PlotIt` Logical; default `FALSE`. If `TRUE`, plot the estimated likelihoods. Plotting is intended for the single-process debugging mode `cl = NULL`.

PlotCutOff A positive integer giving the number of leading features to plot, or an integer vector containing the feature indices to plot. Selecting many features can produce an excessively large plot.

Details

The function trains every column of Data independently and combines the resulting components into one multivariate naive Bayes model. With a non-NULL cluster, the feature-wise fits are evaluated in parallel.

When UseMemshare = FALSE, the standard **parallel** backend is used and worker processes can hold separate copies of the required objects. When UseMemshare = TRUE, **memshare** provides shared-memory objects during the multicore computation [Thrun and Märte, 2026]. The potential memory saving is greatest for large training matrices and multiple workers. Memory sharing has no effect when cl = NULL.

If the number of rows in Data differs from the length of Cls, the inputs are shortened to their common length with a warning. Observations with non-finite class labels are removed before default priors are estimated.

If Data has no column names, feature names of the form X1, X2, and so on are generated. Feature names are retained in the fitted model components.

A precomputed Pareto radius can accelerate repeated fitting, such as during cross-validation. To avoid information leakage, it must be estimated separately within each training fold and never from the corresponding validation data.

Value

A list with the following components:

Model An object of class PDEbayes containing the fitted model. Depending on the selected options, it contains:

- c_Kernels_list A list of length d. Each element is a matrix containing the density kernels for one feature across the k classes.
- ListOfLikelihoods A list of length d. Each element is a matrix containing the estimated likelihood values for one feature across the k classes.
- PDFs_funs A feature-indexed list of class-specific density functions.
- ParetoRadiusPerFeauture For non-Gaussian models, either a numeric vector with one Pareto radius per feature or, when GlobalPR = FALSE, a class-by-feature matrix. For Gaussian models, this component is not used for likelihood estimation.
- Thetas For Gaussian models, a list of length d. Each element is a k by 2 matrix containing the class-wise means and standard deviations. It is NULL for non-Gaussian models.
- Priors A named numeric vector containing the class prior probabilities.
- Plausible Logical value indicating whether plausible likelihood correction was requested.
- PlausibleCenters When plausible correction is used, a d by k numeric matrix of feature-wise class centers; otherwise NULL.

ClsTrain If Predict = TRUE, a numeric vector of length n containing the predicted class labels for the training observations; otherwise NULL.

Posteriors If Predict = TRUE, an n by k matrix of posterior class probabilities; otherwise NULL.

Author(s)

Michael Thrun

References

[Stier et al., 2026] Stier, Q.,Hoffmann, J. & Thrun, M. C.: Classifying with the Fine Structure of Distributions: Leveraging Distributional Information for Robust and Plausible Naïve Bayes, Machine Learning and Knowledge Extraction (MAKE), Vol. 8(1), 13, doi 10.3390/make8010013, MDPI, 2026.

[Thrun and Märte, 2026] Thrun, M.C., Märte, J.: Memshare: Memory Sharing for Multicore Computation in R with an Application to Feature Selection by Mutual Information using PDE, The R Journal, Vol. 17(4), pp. 306 - 322, doi 10.32614/RJ-2025-043, 2026.

See Also

Train_naiveBayes, Predict_naiveBayes, predict.PDEbayes

Examples

```
data(Hepta)
Data <- Hepta$Data
Cls <- Hepta$Cls

## Single-process debugging mode
Vdebug <- Train_naiveBayes_multicore(
  cl = NULL,
  Data = Data,
  Cls = Cls,
  Plausible = FALSE,
  Gaussian = FALSE,
  Predict = TRUE
)

    if(requireNamespace("parallel")){
    cl = parallel::makeCluster(1)#set to number of cores >1
    #each core copies the memory
    V=Train_naiveBayes_multicore(cl=cl,Data=Data,Cls=Cls,
    Predict=TRUE,UseMemshare=FALSE)
    ClsTrain=V$ClsTrain
    table(Cls,ClsTrain)

    #each core shares the memory
    V=Train_naiveBayes_multicore(cl=cl,Data=Data,Cls=Cls,
    Predict=TRUE,UseMemshare=TRUE)
    ClsTrain=V$ClsTrain
    table(Cls,ClsTrain)
    on.exit(parallel::stopCluster(cl))
    }
```

Index