Package 'IPPModel'

March 19, 2018

Type Package

Title Impact pattern plots and feature interaction networks

Version 0.1.0

Date 2018-3-25

Author Xiaohang Zhang

Maintainer Xiaohang Zhang <zhangxiaohang@bupt.edu.cn>

Description This package aims to interpret black box of supervised machine learning models by visualizing the impacts of features on predictical results and feature interaction network.

License GPL-2

Encoding UTF-8

LazyData true

imports rpart (>= 4.1-11), R6 (>= 2.2.0), parallel (>= 3.4.0), xlsx (>= 0.5.7), igraph (>= 1.0.1), akmeans (>= 1.1), cluster (>= 2.0.3), fpc (>= 2.1-10)

RoxygenNote 6.0.1

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

R topics documented:

`BuildTree` *The method 'BuildTree' in IPPModel class*

Description

The method builds tree models for each feature based on the clustering results Clustering.Res. The tree size is determined by the columns 'treeSize' and 'minSplit' in the field ParaTable. The method is executed in parallel using package parallel. Each feature corresponds to one core.

Usage

```
BuildTree()
```

Value

a list of tree rules of each feature. The result is saved in the field TreeRules.

See Also

IPPModel

`CheckInitialization` *The method 'CheckInitialization' in IPPModel class*

Description

The method validates the arguments in the method initialize.

Usage

```
CheckInitialization(XDS, PredType, PredDim, ModelPackage, XB.Size, XB.SamplingMethod)
```

Arguments

XDS	the dataset of input features.
PredType	the type of prediction.
PredDim	indicating which class is predicted.
ModelPackage	the package name of the interpreted machine learning model.
XB.Size	the size of XB.Sample.
XB.SamplingMethod	the sampling method of XB.Sample.

Value

TRUE, if all arguments are validated. FALSE, if at least one argument is not validated.

See Also

IPPModel

CheckParaTable *The method 'CheckParaTable' in IPPModel class*

Description

This method validates the information in ParaTable. For example, if the column 'distMeasure' is 'euclidean' or 'cosine', it is validated.

Usage

```
CheckParaTable()
```

Value

TRUE, if the information in ParaTable are validated; FALSE, otherwise.

See Also

IPPModel

ClusterImpactPlots *The method 'ClusterImpactPlots' in IPPModel class*

Description

Cluster impact curves to generate impact pattern plots.

Usage

```
ClusterImpactPlots()
```

Details

The method clusters the impact curves of each feature based on the predicting results Pred.Res. The parameters of the clustering process are defined in the field ParaTable. The clustering method is defined by the column 'clusteringMethod'; The distance measure is defined by the 'distMeasure'; the column 'centralized' determines if the impact curves are centralized or not before clustering; the 'autoK' tells if the number of clusters is determined automatically by Dunn index or not; the 'numK' determines the maximum number of clusters if autoK = TRUE; otherwise, the 'numK' means the number of clusters. The method is executed in parallel using package parallel. Each feature corresponds to one core.

Value

a list of clustering results of each feature. The result is saved in the field Clustering.Res.

See Also

IPPModel

Clustering.Res *The field 'Clustering.Res' in IPPModel class*

Description

Clustering.Res is the clustering results and is generate by method ClusterImpactPlots. Clustering.Res is a list, each element of which is a data.frame of the clustering results of a feature.

See Also

IPPModel

`ColorList` *The field 'ColorList' in IPPModel class*

Description

ColorList is the list of curve colors used for drawing IPPs. Its default value is c("red", "darkgreen", "blue", "orange", "black", "purple", "pink", "cyan", "cadeblue"). If the default value is not changed and the number of curves in IPPs are greater than the number of colors, 'black' color is used for all of the redundant curves.

See Also

IPPModel

`DrawFIN` *The method 'DrawFIN' in IPPModel class*

Description

The method draws the feature interaction network (FIN) based on the field FIN.Data.

Usage

```
DrawFIN(threshold = 0, lay.out = igraph::layout.auto)
```

Arguments

`threshold`	a numeric, the threshold of link weights. Only the links whose weights are greater than the threshold are shown in FIN.
`lay.out`	the layout defined in the package igraph.

See Also

IPPModel

`DrawIPP` *The method 'DrawIPP' in IPPModel class*

Description

The method draws the impact pattern plots based on the clustering results Clustering.Res.

Usage

```
DrawIPP(centralized=TRUE, nc=4)
```

Arguments

centralized	a boolean, indicating if the impact pattern plots are centralized.
nc	an integer, the number of subplots that are shown in every row.

See Also

IPPModel

ExecuteAll *The method 'ExecuteAll' in IPPModel class*

Description

The method executes the methods SamplingXA, SamplingXB, PredictData, ClusterImpactPlots and BuildTree in sequence.

Usage

```
ExecuteAll()
```

See Also

IPPModel

FIN.Data *The field 'FIN.Data' in IPPModel class*

Description

FIN.Data is a data.frame to describe the feature interaction network, and is generated by the method BuildTree. The row names and column names denote the 'from' nodes and 'to' nodes of the network, respectively. The values in FIN.Data denotes the link weights.

See Also

IPPModel

GenerateParaTable *The method 'GenerateParaTable' in IPPModel class*

Description

This method generates the field ParaTable based on the field X.Data.

Usage

```
GenerateParaTable()
```

Value

a data.frame, which is assigned to the field ParaTable.

Note

The method generates ParaTable based on the data in X.Data and some fixed rules, so the result may be wrong. Please check the information in ParaTable carefully before continuing the other tasks.

See Also

IPPModel

initialize *The method 'initialize' in IPPModel class*

Description

The method initializes some fields and generate the field ParaTable by calling the method GenerateParaTable.

Usage

```
initialize(XDS, PredFun, PredType, PredDim=1, ModelPackage, XB.Size=200,
           XB.SamplingMethod="joint")
```

Arguments

XDS	a data.frame, the dataset of input features. It is assigned to the field X.Data.
PredFun	an object, the prediction function. It is assigned to the field Pred.Fun.
PredType	a string, the type of prediction. It is assigned to the field Pred.Type.
PredDim	an integer, indicating which class is predicted. This field is used only for classification model. It is assigned to the field Pred.Dimension.
ModelPackage	a string, the package name of the interpreted machine learning model, such as "nnet" and "randomforest". It is assigned to the field Model.Package.
XB.Size	an integer, the size of XB.Sample. It is assigned to the field XB.Size.
XB.SamplingMethod	a string, the sampling method of XB.Sample, "joint" or "independent". "joint" means that all features are sampled from X.Data jointly. "independent" means that each feature is sampled independently; then all features are combined randomly. It is assigned to the field XB.SamplingMethod.

See Also

IPPModel

`IPPModel` *Class providing object with methods for drawing IPPs and FIN*

Description

The class provides objects with methods for drawing impact pattern plots (IPPs) and feature interaction network (FIN).

Usage

```
IPPModel
```

Format

`R6Class` object.

Value

Object of `R6Class` with methods for drawing IPPs and FIN.

Fields

`X.Data` a data.frame, the dataset of input features.

`Pred.Fun` an object, the prediction function. It can be any model created by "nnet", "randomforest" and "kernlab" etc.

`Model.Package` a string, the package name of the interpreted machine learning model, such as "nnet" and "randomforest".

`Pred.Type` a string, the type of prediction.

`Pred.Dimension` an integer, indicating which class is predicted. This field is used only for classification model.

`XB.Size` an integer, the size of XB.Sample.

`XB.SamplingMethod` a string, the sampling method of XB.Sample, "joint" or "independent". "joint" means that all features are sampled from X.Data jointly. "independent" means that each feature is sampled independently; then all features are combined randomly.

`ParaTable` a data.frame, the parameter table. It is generated by method GenerateParaTable.

`XA.Sample` a list, the sample of X_A extracted from X.Data. It is generated by method SamplingXA.

`XB.Sample` a list, the sample of X_B extracted from X.Data. It is generated by method SamplingXB.

`Pred.Res` a list, the prediction results of f(X_A,X_B), which is generated by method PredictData.

`Clustering.Res` a list, the clustering results, which is generated by method ClusterImpactPlots.

`TreeRules` a list, the decision tree rules, which is generated by method BuildTree.

`FIN.Data` a data.frame, the feature interaction network, which is generated by method BuildTree.

`ColorList` a list, the curve colors used for drawing IPPs.

`TaskFinishTime` a list, the finishing time of tasks.

Methods

initialize initialize some fields of object and excute the method CheckInitialization and GenerateParaTable.

CheckInitialization validate the initialization information.

GenerateParaTable Generate the parameter table ParaTable.

CheckParaTable validate the information in ParaTable.

SamplingXA sampling XA.Sample from X.Data.

SamplingXB sampling XB.Sample from X.Data.

PredictData predict data using Pred.Fun based on XA.Sample and XB.Sample.

ClusterImpactPlots cluster the impact curves of each feature based on the predicting results Pred.Res.

BuildTree build decision tree based on the clustering results Clustering.Res.

DrawIPP draw the impact pattern plots.

DrawFIN draw the feature interaction network.

WriteToExcel write the results to an excel file.

ExecuteAll execute the methods SamplingXA, SamplingXB, PredictData, ClusterImpactPlots and BuildTree in sequence.

References

Xiaohang Zhang, Ji Zhu, SuBang Choe, Yi Lu and Jing Liu. Exploring black box of supervised learning models: Visualizing the impact of features on prediction. Working paper.

Examples

```
library(IPPModel)
library(igraph)

#------ FIRST EXAMPLE ------
library(nnet)
data("bank")
# build model
bank.NN <- nnet(y ~ ., data = bank, size = 5, maxit = 1000)
# remove the output variable
bank.ds = bank[-17]
# create IPPModel object
IPP.bank = IPPModel$new(XDS=bank.ds, PredFun=bank.NN,
                        ModelPackage="nnet", PredType="raw", PredDim=1,
                        XB.Size=1000, XB.SamplingMethod="joint")
# modify the clustering method to "kmedoids"
IPP.bank$ParaTable$clusteringMethod = "kmedoids"
# execute all tasks
IPP.bank$ExecuteAll()
# draw impact pattern plots (IPP)
IPP.bank$DrawIPP(centralized = TRUE, nc = 4)
# draw feature interaction network (FIN)
IPP.bank$DrawFIN(threshold = 0.2, lay.out = igraph::layout.auto)
# write the results into an excel file
IPP.bank$WriteToExcel("output.xlsx")

#------ SECOND EXAMPLE ------
library(randomForest)
```

```
data("whitewine")
# build model
WW.RF <- randomForest(quality ~ ., data = whitewine, mtry = 4,importance=TRUE, na.action=na.omit)
# remove the output variable
WW.ds = whitewine[-12]
# create IPPModel object
IPP.WW = IPPModel$new(XDS=WW.ds, PredFun=WW.RF,
                      ModelPackage="randomForest", PredType="response", PredDim=1,
                      XB.Size=1000, XB.SamplingMethod="joint")
# set the maximum depth of trees to be 5
IPP.WW$ParaTable$treeDepth = 5
# execute all tasks
IPP.WW$ExecuteAll()
# draw impact pattern plots (IPP)
IPP.WW$DrawIPP(centralized = TRUE, nc = 4)
# draw feature interaction network (FIN)
IPP.WW$DrawFIN(threshold = 0.1, lay.out = igraph::layout.circle)

#------ THIRD EXAMPLE ------
library(kernlab)
data("iris")
iris.SVM <- ksvm(Species ~ ., data = iris,kernel="rbfdot", kpar="automatic",C=0.1, prob.model = TRUE)
# remove the output variable
iris.ds = iris[-5]
# create IPPModel object
IPP.iris = IPPModel$new(XDS=iris.ds, PredFun=iris.SVM,
                        ModelPackage="kernlab", PredType="prob", PredDim=1,
                        XB.Size=200, XB.SamplingMethod="independent")
# execute the tasks step by step
IPP.iris$SamplingXA()  # sampling XA
IPP.iris$SamplingXB()  # sampling XB
IPP.iris$PredictData()  # predict
IPP.iris$ClusterImpactPlots() # clustering impact plots
IPP.iris$BuildTree()  # build tree
# draw impact pattern plots (IPP)
IPP.iris$DrawIPP(centralized = TRUE, nc = 4)
# draw feature interaction network (FIN)
IPP.iris$DrawFIN(threshold = 0.3, lay.out = igraph::layout.auto)
# write the results into an excel file
IPP.iris$WriteToExcel("output.xlsx")
```

`Model.Package` *The field 'Model.Package' in IPPModel class*

Description

Model.Package is the package name of the interpreted machine learning model Pred.Fun, such as "nnet" and "randomforest".

See Also

IPPModel

ParaTable *The field 'ParaTable' in IPPModel class*

Description

The field 'ParaTable' provides some parameter information for the IPPModel class. It is a data.frame whose rowNames are the names of the input features in X.Data. It includs the following columns.

Columns

dataType string, the data types of input features. It should be "interval", "binary", "ordinal" or "nominal". Interval feature is one for which the mean (or average) makes sense, such as person's height. Binary feature has only two possible levels. Gender is an example. Nominal variable has more than two levels, but the values of the levels have no implied order. Colors are examples. Ordinal feature has more than two levels, and the values of the levels have an implied order. Coffee sizes, such as small, medium, and large, are examples.

uniqueValue integer, the numbe of unique values of the input features.

X_A boolean, indicating if the feature serves as a target feature to draw impact pattern plots (IPPs) and involves in the feature interaction network (FIN).

L_A integer, the number of levels sampled from X.Data for the feature.

samplingMethod string, the sampling method of the feature. It should be "equal", "percentile" or "random". "equal" means that all levels are sampled with equal width from the range of the feature. "percentile" means that all levels are sampled based on the percentiles of the feature's distribution. "random" means that all levels are sampled randomly without replacement.

clusteringMethod string, the clustering method. It should be "kmeans" or "kmedoids".

centralized boolean, indicating if the feature's impact plots are centralized before clustering.

distMeasure string, the distance measure used in clustering process. It should be "euclidean" or "cosine".

autoK boolean, indicating if the number of clusters is determined automatically based on the Dunn index.

numK integer, the number of clusters if autoK = FALSE. If autoK = TRUE, numK denotes the maximum number of clusters.

treeDepth integer, the maximum tree depth. It is used to control the size of decision tree.

minSplit integer, the minimum number of observations for tree node splitting. It is used to control the size of decision tree.

Note

The ParaTable can be generated by the method GenerateParaTable based on the field X.Data. However, the user must check the parameter information carefully before continuing the other tasks.

See Also

IPPModel

Pred.Dimension *The field 'Pred.Dimension' in IPPModel class*

Description

Pred.Dimension is an integer, indicating which class is predicted. This field is used only for classification model. For example, the output field is assumed to have three classes, so Pred.Dimension can be 1, 2 or 3. If Pred.Dimension equals 1, then the first class is predicted and analyzed. Note that if Pred.Dimension is larger than 3, then it is reset to 1.

See Also

IPPModel

Pred.Fun *The field 'Pred.Fun' in IPPModel class*

Description

Pred.fun is the prediction function. It can be any model created by "nnet", "randomforest" and "kernlab" etc. Pred.fun must be consistent with the field X.Data.

See Also

IPPModel

Pred.Res *The field 'Pred.Res' in IPPModel class*

Description

Pred.Res is the prediction results of the model defined by Pred.Fun for the combination of XA.Sample and XB.Sample. Pred.Res is generated by the method PredictData. Pred.Res is a list, each element of which is a data.frame of the prediction results of a feature.

See Also

IPPModel

Pred.Type — *The description of the field 'Pred.Type' in IPPModel class*

Description

Pred.Type is the type of prediction. The IPP model only supports interpreting the models with numeircal outputs, so Pred.Type is used to assure that the output is numeric, not class. The value of Pred.Type depends on what package is used. For example, if "nnet" is used to create Pred.Fun, Pred.Type should be "raw" for both classification and regression. If "randomForest" or "kernlab" is used, Pred.Type should be "prob" for classification, or be "response" for regression. If other package is used, you should reference the predict function of that package.

See Also

IPPModel

PredictData — *The method 'PredictData' in IPPModel class*

Description

The method predicts data using Pred.Fun based on XA.Sample and XB.Sample. The method is executed in parallel using package parallel. Each feature corresponds to one core.

Usage

```
PredictData()
```

Value

a list of predicting results of each feature. The result is saved in the field Pred.Res.

See Also

IPPModel

SamplingXA — *The method 'SamplingXA' in IPPModel class*

Description

The method samples X_A from X.Data based on the sample size and the sampling method defined by "L_A" and "samplingMethod" in ParaTable, respectively.

Usage

```
SamplingXA()
```

Value

a list of samples of each feature. The result is saved in the field XA.Sample.

See Also

IPPModel

`SamplingXB` *The method 'SamplingXB' in IPPModel class*

Description

The method samples X_B from X.Data based on the sample size and sampling method defined by the fields XB.Size and XB.SamplingMethod, respectively.

Usage

```
SamplingXB()
```

Value

a data.frame of sample. It is saved in the field XB.Sample.

See Also

IPPModel

`TaskFinishTime` *The field 'TaskFinishTime' in IPPModel class*

Description

TaskFinishTime is a list for recording the finishing time of tasks, including initialization, sampling X_A and X_B, prediction, clustering, obtaining tree rules and building FIN.

See Also

IPPModel

`TreeRules` *The field 'TreeRules' in IPPModel class*

Description

TreeRules are the decision tree rules, which is generated by method BuildTree. It is a list, each element of which is a data.frame of tree rules and the corresponding cluster distribution of a feature.

See Also

IPPModel

WriteToExcel — *The method 'WriteToExcel' in IPPModel class*

Description

The method writes some results into an excel file, including the impact pattern plots, the tree rules and the feature interaction network.

Usage

```
WriteToExcel(excelName)
```

Arguments

excelName	a string, the path and name of the excel file.

See Also

IPPModel

X.Data — *The field 'X.Data' in IPPModel class*

Description

X.Data is a data.frame and is the dataset of input features. X.Data must be consistent with the field Pred.Fun that is the prediction function. Note that the dataset should not include the output feature (y).

See Also

IPPModel

XA.Sample — *The field 'XA.Sample' in IPPModel class*

Description

XA.Sample is the sample of X_A extracted from X.Data. It is generated by method SamplingXA. XA.Sample is a list, each element of which is a vector of the sample values of a feature. The included features in XA.Sample are determined by the column 'X_A' in the field ParaTable. The size of XA.Sample is defined by the column 'L_A' in the field ParaTable. The sampling method of XA.Sample is defined by the column 'samplingMethod' in the field ParaTable.

See Also

IPPModel

`XB.Sample` *The field 'XB.Sample' in IPPModel class*

Description

XB.Sample is the sample of X_B extracted from X.Data. It is generated by method SamplingXB. The size of XB.Sample is defined by the field XB.Size. The sampling method of XB.Sample is defined by the field XB.SamplingMethod. XB.Sample is a data.frame whose columns are the same as X.Data.

See Also

IPPModel

`XB.SamplingMethod` *The field 'XB.SamplingMethod' in IPPModel class*

Description

XB.SamplingMethod is the sampling method of XB.Sample, "joint" or "independent". "joint" means that all features are sampled from X.Data jointly. "independent" means that each feature is sampled independently; then all features are combined randomly. Note that the "joint" method can keep the features' joint distribution unchanged.

See Also

IPPModel

`XB.Size` *The field 'XB.Size' in IPPModel class*

Description

XB.size is the size of XB.Sample. Larger XB.size can bring more robust IPPs and FIN , but higher computational costs.

See Also

IPPModel

Index