Package 'localICE'

July 22, 2025

Type Package

Title Local Individual Conditional Expectation

Version 0.1.1

Maintainer Martin Walter <mf-walter@web.de>

Description Local Individual Conditional Expectation ('localICE') is a local explanation approach from the field of eXplainable Artificial Intelligence (XAI). localICE is a model-agnostic XAI approach which provides three-dimensional local explanations for particular data instances. The approach is proposed in the master thesis of Martin Walter as an extension to ICE (see Reference). The three dimensions are the two features at the horizontal and vertical axes as well as the target represented by different colors. The approach is applicable for classification and regression problems to explain interactions of two features towards the target. For classification models, the number of classes can be more than two and each class is added as a different color to the plot. The given instance is added to the plot as two dotted lines according to the feature values. The localICE-package can explain features of type factor and numeric of any machine learning model. Automatically supported machine learning packages are 'mlr', 'randomForest', 'caret' or all other with an S3 predict function. For further model types from other libraries, a predict function has to be provided as an argument in order to get access to the model. Reference to the ICE approach: Alex Goldstein, Adam Kapelner, Justin Bleich, Emil Pitkin (2013) <doi:10.48550/arXiv.1309.6392>.

URL https://github.com/viadee/localICE

BugReports https://github.com/viadee/localICE/issues

License BSD_3_clause + file LICENSE

Encoding UTF-8

LazyData true

Imports ggplot2, checkmate

Suggests covr, h2o, mlbench, randomForest, stats, testthat, utils

RoxygenNote 7.0.2

NeedsCompilation no

Author Martin Walter [aut, cre]

Repository CRAN

Date/Publication 2020-02-07 23:20:08 UTC

Contents

`localICE` *Local Individual Conditional Expectation (localICE)*

Description

Local Individual Conditional Expectation (localICE) is a local explanation approach from the field of eXplainable Artificial Intelligence (XAI). It is proposed in the master thesis of the author of this package as an extension to ICE and is a three-dimensional local explanation for particular data instances. The three dimensions are the two features at the horizontal and vertical axes as well as the target represented by different colors. The approach is applicable for classification and regression problems to explain interactions of two features towards the target. The plot for discrete targets looks similar to plots of cluster algorithms like k-means, where different clusters represent different predictions. The given `instance` is added to the plot as two dotted lines according to the feature values. The `localICE`-package can explain features of type `factor` and `numeric`.

Usage

```
localICE(
  instance,
  data,
  feature_1,
  feature_2,
  target,
  model,
  predict.fun = NULL,
  regression = TRUE,
  step_1 = 1,
  step_2 = 1
  )
```

Arguments

`instance`	instance is a row of `data` that has to be explained by means of localICE.
`data`	a data frame containing all predictors and a representative distribution of data instances (rows). The data set can be the test data from model creation and does not have to contain predictions or true labels. The data set is needed to get the data distribution of `feature_1` and `feature_2` that should be explained for the given `instance`. The data distribution is then used to perturb the values of the two given features.
`feature_1`	a feature of interest as `character`
`feature_2`	an other feature of interest as `character`

target	the name of the target as character. It is required to name the legend of the plot.
model	a machine learning model as object.
predict.fun	a prediction function if model is not of type randomForest, mlr or caret. An exemplary function for the machine learning library h2o is shown below in the "Examples" section
regression	if the model is not a regression problem but a classification problem, then set regression = FALSE.
step_1	set how accurate the explanation according to feature_1 should be. Step is only required if feature_1 is numeric. The greater the step, the faster the computation and the less accurate the explanation for feature_1. The step has to be smaller than max(data[,feature_1]) - min(data[,feature_1]) and greater than 0. For integer features, the step should also be an integer to avoid biased model predictions.
step_2	same as step_1 but for feature_2

Details

The computation time of localICE is strongly dependent to the distribution of feature_1, feature_2 and the steps step_1 and step_2 for numerical features.

Value

The function localICE returns a ggplot2 object that can be modified with further ggplot2 functions.

References

Goldstein, Alex; Kapelner, Adam; Bleich, Justin; Pitkin, Emil (2013): "Peeking Inside the Black Box: Visualizing Statistical Learning With Plots of Individual Conditional Expectation". In: Journal of Computational and Graphical Statistics 24.1 (2013), pp. 44-65. doi: 10.1080/10618600.2014.907095. url: https://doi.org/10.1080/10618600.2014.907095

Examples

```
# Regression example:
if(require("randomForest")){
  rf = randomForest(Sepal.Length ~., data = iris, ntree = 20)

  explanation = localICE(
    instance = iris[1, ],
    data = iris,
    feature_1 = "Species",
    feature_2 = "Sepal.Width",
    target = "Sepal.Length",
    model = rf,
    regression = TRUE,
    step_2 = 0.1
  )
```

```
  plot(explanation)
}

# Classification example:
if(require("randomForest") && require("mlbench")){
  data("PimaIndiansDiabetes")
  rf = randomForest(diabetes ~., data = PimaIndiansDiabetes, ntree = 20)

  explanation = localICE(
    instance = PimaIndiansDiabetes[8, ],
    data = PimaIndiansDiabetes,
    feature_1 = "age",
    feature_2 = "glucose",
    target = "diabetes",
    model = rf,
    regression = FALSE,
    step_1 = 5,
    step_2 = 5
  )
  plot(explanation)
}
# An example of how to use predict.fun to use any machine learning library,
# in this case the library h2o (please see GitHub for the complete h2o example):
predict.fun = function(model, newdata){
  prediction = h2o.predict(model, as.h2o(newdata))
  prediction = as.data.frame(prediction)
  prediction = prediction$prediction
  return(prediction)
}
```

Index