Package ‘explainer’

September 30, 2024

Title Machine Learning Model Explainer

Version 1.0.2

Description It enables detailed interpretation of complex classification and regression models through Shapley analysis including data-driven characterization of subgroups of individuals. Furthermore, it facilitates multi-measure model evaluation, model fairness, and decision curve analysis. Additionally, it offers enhanced visualizations with interactive elements.

License MIT + file LICENSE

Encoding UTF-8

URL `https://persimune.github.io/explainer/`, `https://github.com/PERSIMUNE/explainer`

BugReports `https://github.com/PERSIMUNE/explainer/issues`

RoxygenNote 7.2.1

Imports cvms, data.table, dplyr, egg, ggplot2, ggpmisc, ggpubr, magrittr, plotly, tibble, tidyr, writexl, gridExtra, scales

Suggests cowplot, mlr3, mlr3learners, knitr, broom, iml, forcats, mlr3viz, plotROC, psych, reshape2, remotes, mlbench, ranger, precrec, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Ramtin Zargari Marandi [aut, cre] (<`https://orcid.org/0000-0001-9233-1656`>)

Maintainer Ramtin Zargari Marandi <`ramtin.zargari.marandi@regionh.dk`>

Repository CRAN

Date/Publication 2024-09-30 17:30:02 UTC

Contents

`eCM_plot` *Enhanced Confusion Matrix Plot*

Description

This function generates an enhanced confusion matrix plot using the CVMS package. The plot includes visualizations of sensitivity, specificity, positive predictive value (PPV), and negative predictive value (NPV).

Usage

```
eCM_plot(task, trained_model, splits, add_sums = TRUE, palette = "Green")
```

Arguments

`task`	mlr3 task object specifying the task details
`trained_model`	mlr3 trained learner (model) object obtained after training
`splits`	mlr3 object defining data splits for train and test sets
`add_sums`	logical, indicating whether total numbers should be displayed in the plot (default: TRUE)
`palette`	character, the color palette for the confusion matrix (default: "Green")

Value

A confusion matrix plot visualizing sensitivity, specificity, PPV, and NPV

Examples

```
library("explainer")
seed <- 246
set.seed(seed)

# Load necessary packages
if (!requireNamespace("mlbench", quietly = TRUE)) stop("mlbench not installed.")
if (!requireNamespace("mlr3learners", quietly = TRUE)) stop("mlr3learners not installed.")
if (!requireNamespace("ranger", quietly = TRUE)) stop("ranger not installed.")
# Load BreastCancer dataset
utils::data("BreastCancer", package = "mlbench")
target_col <- "Class"
```

```
positive_class <- "malignant"
mydata <- BreastCancer[, -1]
mydata <- na.omit(mydata)
sex <- sample(
  c("Male", "Female"),
  size = nrow(mydata),
  replace = TRUE
)
mydata$age <- as.numeric(sample(
  seq(18, 60),
  size = nrow(mydata),
  replace = TRUE
))
mydata$sex <- factor(
  sex,
  levels = c("Male", "Female"),
  labels = c(1, 0)
)
maintask <- mlr3::TaskClassif$new(
  id = "my_classification_task",
  backend = mydata,
  target = target_col,
  positive = positive_class
)
splits <- mlr3::partition(maintask)
mylrn <- mlr3::lrn(
  "classif.ranger",
  predict_type = "prob"
)
mylrn$train(maintask, splits$train)
myplot <- eCM_plot(
  task = maintask,
  trained_model = mylrn,
  splits = splits
)
```

`eDecisionCurve` *Decision Curve Plot*

Description

Decision curve analysis is a statistical method used in medical research to evaluate and compare the clinical utility of different diagnostic or predictive models. It assesses the net benefit of a model across a range of decision thresholds, aiding in the selection of the most informative and practical approach for guiding clinical decisions.

Usage

```
eDecisionCurve(task, trained_model, splits, seed = 246)
```

Arguments

`task`	mlr3 task object specifying the task details
`trained_model`	mlr3 trained learner (model) object obtained after training
`splits`	mlr3 object defining data splits for train and test sets
`seed`	numeric, seed for reproducibility (default: 246)

Value

An interactive decision curve plot

Examples

```
library("explainer")
seed <- 246
set.seed(seed)
# Load necessary packages
if (!requireNamespace("mlbench", quietly = TRUE)) stop("mlbench not installed.")
if (!requireNamespace("mlr3learners", quietly = TRUE)) stop("mlr3learners not installed.")
if (!requireNamespace("ranger", quietly = TRUE)) stop("ranger not installed.")
# Load BreastCancer dataset
utils::data("BreastCancer", package = "mlbench")
target_col <- "Class"
positive_class <- "malignant"
mydata <- BreastCancer[, -1]
mydata <- na.omit(mydata)
sex <- sample(
  c("Male", "Female"),
  size = nrow(mydata),
  replace = TRUE
)
mydata$age <- as.numeric(sample(
  seq(18, 60),
  size = nrow(mydata),
  replace = TRUE
))
mydata$sex <- factor(
  sex,
  levels = c("Male", "Female"),
  labels = c(1, 0)
)
maintask <- mlr3::TaskClassif$new(
  id = "my_classification_task",
  backend = mydata,
  target = target_col,
  positive = positive_class
)
splits <- mlr3::partition(maintask)
mylrn <- mlr3::lrn(
  "classif.ranger",
  predict_type = "prob"
)
```

```
mylrn$train(maintask, splits$train)
myplot <- eDecisionCurve(
  task = maintask,
  trained_model = mylrn,
  splits = splits,
  seed = seed
)
```

`eFairness` *Enhanced Fairness Analysis*

Description

This function generates Precision-Recall and ROC curves for sample subgroups, facilitating fairness analysis of a binary classification model.

Usage

```
eFairness(task, trained_model, splits, target_variable, var_levels)
```

Arguments

`task`	mlr3 binary classification task object specifying the task details
`trained_model`	mlr3 trained learner (model) object obtained after training
`splits`	mlr3 object defining data splits for train and test sets
`target_variable`	character, the variable from the dataset used to test the model's performance against
`var_levels`	list, defining the levels for the specified variable

Value

Model performance metrics for user-specified subgroups using Precision-Recall and ROC curves

Examples

```
library("explainer")
seed <- 246
set.seed(seed)
# Load necessary packages
if (!requireNamespace("mlbench", quietly = TRUE)) stop("mlbench not installed.")
if (!requireNamespace("mlr3learners", quietly = TRUE)) stop("mlr3learners not installed.")
if (!requireNamespace("ranger", quietly = TRUE)) stop("ranger not installed.")
# Load BreastCancer dataset
utils::data("BreastCancer", package = "mlbench")
target_col <- "Class"
positive_class <- "malignant"
mydata <- BreastCancer[, -1]
```

```
mydata <- na.omit(mydata)
sex <- sample(
  c("Male", "Female"),
  size = nrow(mydata),
  replace = TRUE
)
mydata$age <- as.numeric(sample(
  seq(18, 60),
  size = nrow(mydata),
  replace = TRUE
))
mydata$sex <- factor(
  sex,
  levels = c("Male", "Female"),
  labels = c(1, 0)
)
maintask <- mlr3::TaskClassif$new(
  id = "my_classification_task",
  backend = mydata,
  target = target_col,
  positive = positive_class
)
splits <- mlr3::partition(maintask)
mylrn <- mlr3::lrn(
  "classif.ranger",
  predict_type = "prob"
)
mylrn$train(maintask, splits$train)
# sex is chosen for fairness analysis
Fairness_results <- eFairness(
  task = maintask,
  trained_model = mylrn,
  splits = splits,
  target_variable = "sex",
  var_levels = c("Male", "Female")
)
```

`ePerformance` *Enhanced Performance Evaluation*

Description

This function generates Precision-Recall and ROC curves, including threshold information for binary classification models.

Usage

```
ePerformance(task, trained_model, splits)
```

Arguments

`task`	mlr3 binary classification task object specifying the task details
`trained_model`	mlr3 trained learner (model) object obtained after training
`splits`	mlr3 object defining data splits for train and test sets

Value

ROC and Precision-Recall curves with threshold information

Examples

```
# Set environment variables for reproducibility
Sys.setenv(LANG = "en") # Change R language to English!
RNGkind("L'Ecuyer-CMRG") # Change to L'Ecuyer-CMRG instead of the default "Mersenne-Twister"

# Load required libraries
library("explainer")

# Set seed for reproducibility
seed <- 246
set.seed(seed)

# Load necessary packages
if (!requireNamespace("mlbench", quietly = TRUE)) stop("mlbench not installed.")
if (!requireNamespace("mlr3learners", quietly = TRUE)) stop("mlr3learners not installed.")
if (!requireNamespace("ranger", quietly = TRUE)) stop("ranger not installed.")
# Load BreastCancer dataset
utils::data("BreastCancer", package = "mlbench")

# Keep the target column as "Class"
target_col <- "Class"

# Change the positive class to "malignant"
positive_class <- "malignant"

# Keep only the predictor variables and outcome
mydata <- BreastCancer[, -1] # 1 is ID

# Remove rows with missing values
mydata <- na.omit(mydata)

# Create a vector of sex categories
sex <- sample(c("Male", "Female"), size = nrow(mydata), replace = TRUE)

# Create a vector of age categories
mydata$age <- as.numeric(sample(seq(18, 60), size = nrow(mydata), replace = TRUE))

# Add a sex column to the mydata data frame (for fairness analysis)
mydata$sex <- factor(sex, levels = c("Male", "Female"), labels = c(1, 0))

# Create a classification task
```

```
maintask <- mlr3::TaskClassif$new(
  id = "my_classification_task",
  backend = mydata,
  target = target_col,
  positive = positive_class
)

# Create a train-test split
set.seed(seed)
splits <- mlr3::partition(maintask)

# Add a learner (machine learning model base)
# Here we use random forest for example (you can use any other available model)
mylrn <- mlr3::lrn("classif.ranger", predict_type = "prob")

# Train the model
mylrn$train(maintask, splits$train)

# Make predictions on new data
mylrn$predict(maintask, splits$test)
ePerformance(task = maintask, trained_model = mylrn, splits = splits)
```

`eROC_plot` *Enhanced ROC and Precision-Recall Plots*

Description

This function generates Precision-Recall and ROC curves for binary classification models.

Usage

```
eROC_plot(task, trained_model, splits)
```

Arguments

`task`	mlr3 binary classification task object specifying the task details
`trained_model`	mlr3 trained learner (model) object obtained after training
`splits`	mlr3 object defining data splits for train and test sets

Value

ROC and Precision-Recall curves

Examples

```
library("explainer")
seed <- 246
set.seed(seed)
# Load necessary packages
```

```
if (!requireNamespace("mlbench", quietly = TRUE)) stop("mlbench not installed.")
if (!requireNamespace("mlr3learners", quietly = TRUE)) stop("mlr3learners not installed.")
if (!requireNamespace("ranger", quietly = TRUE)) stop("ranger not installed.")
# Load BreastCancer dataset
utils::data("BreastCancer", package = "mlbench")
target_col <- "Class"
positive_class <- "malignant"
mydata <- BreastCancer[, -1]
mydata <- na.omit(mydata)
sex <- sample(
  c("Male", "Female"),
  size = nrow(mydata),
  replace = TRUE
)
mydata$age <- as.numeric(sample(
  seq(18, 60),
  size = nrow(mydata),
  replace = TRUE
))
mydata$sex <- factor(
  sex,
  levels = c("Male", "Female"),
  labels = c(1, 0)
)
maintask <- mlr3::TaskClassif$new(
  id = "my_classification_task",
  backend = mydata,
  target = target_col,
  positive = positive_class
)
splits <- mlr3::partition(maintask)
mylrn <- mlr3::lrn(
  "classif.ranger",
  predict_type = "prob"
)
mylrn$train(maintask, splits$train)
myplot <- eROC_plot(
  task = maintask,
  trained_model = mylrn,
  splits = splits
)
```

`eSHAP_plot_reg` *Enhanced SHAP Analysis for Regression Models*

Description

The SHAP plot for regression models is a visualization tool that uses the Shapley value, an approach from cooperative game theory, to compute feature contributions for single predictions. The Shapley value fairly distributes the difference of the instance's prediction and the datasets average prediction among the features. This method is available from the iml package.

Usage

```
eSHAP_plot_reg(
  task,
  trained_model,
  splits,
  sample.size = 30,
  seed = 246,
  subset = 1
)
```

Arguments

`task`	mlr3 regression task object specifying the task details
`trained_model`	mlr3 trained learner (model) object obtained after training
`splits`	mlr3 object defining data splits for train and test sets
`sample.size`	numeric, number of samples to calculate SHAP values (default: 30)
`seed`	numeric, seed for reproducibility (default: 246)
`subset`	numeric, proportion of the test set to use for visualization (default: 1)

Value

A list of two objects:

1. An enhanced SHAP plot with user interactive elements,
2. A matrix of SHAP values

Examples

```
library("explainer")
seed <- 246
set.seed(seed)
# Load necessary packages
if (!requireNamespace("mlbench", quietly = TRUE)) stop("mlbench not installed.")
if (!requireNamespace("mlr3learners", quietly = TRUE)) stop("mlr3learners not installed.")
if (!requireNamespace("ranger", quietly = TRUE)) stop("ranger not installed.")
# Load BreastCancer dataset
utils::data("BreastCancer", package = "mlbench")
mydata <- BreastCancer[, -1]
mydata <- na.omit(mydata)
sex <- sample(c("Male", "Female"), size = nrow(mydata), replace = TRUE)
mydata$age <- sample(seq(18, 60), size = nrow(mydata), replace = TRUE)
mydata$sex <- factor(sex, levels = c("Male", "Female"), labels = c(1, 0))
mydata$Class <- NULL
mydata$Cl.thickness <- as.numeric(mydata$Cl.thickness)
target_col <- "Cl.thickness"
maintask <- mlr3::TaskRegr$new(
  id = "my_regression_task",
  backend = mydata,
  target = target_col
```

```
)
splits <- mlr3::partition(maintask)
mylrn <- mlr3::lrn("regr.ranger", predict_type = "response")
mylrn$train(maintask, splits$train)
reg_model_outputs <- mylrn$predict(maintask, splits$test)
SHAP_output <- eSHAP_plot_reg(
  task = maintask,
  trained_model = mylrn,
  splits = splits,
  sample.size = 2, # also 30 or more
  seed = seed,
  subset = 0.02 # up to 1
)
myplot <- SHAP_output[[1]]
```

`SHAPclust` *SHAP clustering*

Description

SHAP values are used to cluster data samples using the k-means method to identify subgroups of individuals with specific patterns of feature contributions.

Usage

```
SHAPclust(
  task,
  trained_model,
  splits,
  shap_Mean_wide,
  shap_Mean_long,
  num_of_clusters = 4,
  seed = 246,
  subset = 1,
  algorithm = "Hartigan-Wong",
  iter.max = 1000
)
```

Arguments

`task`	an mlr3 task for binary classification
`trained_model`	an mlr3 trained learner object
`splits`	an mlr3 object defining data splits for train and test sets
`shap_Mean_wide`	the data frame of SHAP values in wide format from eSHAP_plot.R
`shap_Mean_long`	the data frame of SHAP values in long format from eSHAP_plot.R

`num_of_clusters`	number of clusters to make based on SHAP values, default: 4
`seed`	an integer for reproducibility, Default to 246
`subset`	what percentage of the instances to use from 0 to 1 where 1 means all
`algorithm`	k-means algorithm character: "Hartigan-Wong", "Lloyd", "Forgy", "MacQueen".
`iter.max`	maximum number of iterations allowed

Value

A list containing four elements:

`shap_plot_onerow`	An interactive plot displaying the SHAP values for each feature, clustered by the specified number of clusters. Each cluster is shown in a facet.
`combined_plot`	A ggplot2 figure combining confusion matrices for each cluster, providing insights into the model's performance within each identified subgroup.
`kmeans_fvals_desc`	A summary table containing statistical descriptions of the clusters based on feature values.
`shap_Mean_wide_kmeans`	A data frame containing clustered SHAP values along with predictions and ground truth information.
`kmeans_info`	Information about the k-means clustering process, including cluster centers and assignment details.

References

Zargari Marandi, R., 2024. ExplaineR: an R package to explain machine learning models. Bioinformatics advances, 4(1), p.vbae049, https://doi.org/10.1093/bioadv/vbae049.

See Also

Other functions to visualize and interpret machine learning models: `eSHAP_plot`.

Examples

```
library("explainer")
seed <- 246
set.seed(seed)
# Load necessary packages
if (!requireNamespace("mlbench", quietly = TRUE)) stop("mlbench not installed.")
if (!requireNamespace("mlr3learners", quietly = TRUE)) stop("mlr3learners not installed.")
if (!requireNamespace("ranger", quietly = TRUE)) stop("ranger not installed.")
# Load BreastCancer dataset
utils::data("BreastCancer", package = "mlbench")
target_col <- "Class"
positive_class <- "malignant"
mydata <- BreastCancer[, -1]
mydata <- na.omit(mydata)
```

```
sex <- sample(
  c("Male", "Female"),
  size = nrow(mydata),
  replace = TRUE
)
mydata$age <- as.numeric(sample(
  seq(18, 60),
  size = nrow(mydata),
  replace = TRUE
))
mydata$sex <- factor(
  sex,
  levels = c("Male", "Female"),
  labels = c(1, 0)
)
maintask <- mlr3::TaskClassif$new(
  id = "my_classification_task",
  backend = mydata,
  target = target_col,
  positive = positive_class
)
splits <- mlr3::partition(maintask)
mylrn <- mlr3::lrn(
  "classif.ranger",
  predict_type = "prob"
)
mylrn$train(maintask, splits$train)
SHAP_output <- eSHAP_plot(
  task = maintask,
  trained_model = mylrn,
  splits = splits,
  sample.size = 2, # also 30 or more
  seed = seed,
  subset = 0.02 # up to 1
)
shap_Mean_wide <- SHAP_output[[2]]
shap_Mean_long <- SHAP_output[[3]]
SHAP_plot_clusters <- SHAPclust(
  task = maintask,
  trained_model = mylrn,
  splits = splits,
  shap_Mean_wide = shap_Mean_wide,
  shap_Mean_long = shap_Mean_long,
  num_of_clusters = 3, # your choice
  seed = seed,
  subset = 0.02, # match with eSHAP_plot
  algorithm = "Hartigan-Wong",
  iter.max = 10
)
```

`ShapFeaturePlot` *SHAP Values versus Feature Values*

Description

SHAP values in association with feature values

Usage

```
ShapFeaturePlot(shap_Mean_long)
```

Arguments

`shap_Mean_long`	the data frame containing SHAP values in long format

Value

an interactive plot of SHAP values in association with feature values

Examples

```
library("explainer")
seed <- 246
set.seed(seed)
# Load necessary packages
if (!requireNamespace("mlbench", quietly = TRUE)) stop("mlbench not installed.")
if (!requireNamespace("mlr3learners", quietly = TRUE)) stop("mlr3learners not installed.")
if (!requireNamespace("ranger", quietly = TRUE)) stop("ranger not installed.")
# Load BreastCancer dataset
utils::data("BreastCancer", package = "mlbench")
target_col <- "Class"
positive_class <- "malignant"
mydata <- BreastCancer[, -1]
mydata <- na.omit(mydata)
sex <- sample(
  c("Male", "Female"),
  size = nrow(mydata),
  replace = TRUE
)
mydata$age <- as.numeric(sample(
  seq(18, 60),
  size = nrow(mydata),
  replace = TRUE
))
mydata$sex <- factor(
  sex,
  levels = c("Male", "Female"),
  labels = c(1, 0)
)
maintask <- mlr3::TaskClassif$new(
```

```
  id = "my_classification_task",
  backend = mydata,
  target = target_col,
  positive = positive_class
)
splits <- mlr3::partition(maintask)
mylrn <- mlr3::lrn(
  "classif.ranger",
  predict_type = "prob"
)
mylrn$train(maintask, splits$train)
SHAP_output <- eSHAP_plot(
  task = maintask,
  trained_model = mylrn,
  splits = splits,
  sample.size = 2, # also 30 or more
  seed = seed,
  subset = 0.02 # up to 1
)
shap_Mean_long <- SHAP_output[[3]]
myplot <- ShapFeaturePlot(shap_Mean_long)
```

`ShapPartialPlot` *SHAP Partial Plot*

Description

Generates an interactive partial dependence plot based on SHAP values, visualizing the marginal effect of one or two features on the predicted outcome of a machine learning model.

Usage

```
ShapPartialPlot(shap_Mean_long)
```

Arguments

`shap_Mean_long`	data frame containing SHAP values in long format

Value

an interactive partial dependence plot

Examples

```
library("explainer")
seed <- 246
set.seed(seed)
# Load necessary packages
if (!requireNamespace("mlbench", quietly = TRUE)) stop("mlbench not installed.")
```

```
if (!requireNamespace("mlr3learners", quietly = TRUE)) stop("mlr3learners not installed.")
if (!requireNamespace("ranger", quietly = TRUE)) stop("ranger not installed.")
# Load BreastCancer dataset
utils::data("BreastCancer", package = "mlbench")
target_col <- "Class"
positive_class <- "malignant"
mydata <- BreastCancer[, -1]
mydata <- na.omit(mydata)
sex <- sample(
  c("Male", "Female"),
  size = nrow(mydata),
  replace = TRUE
)
mydata$age <- as.numeric(sample(
  seq(18, 60),
  size = nrow(mydata),
  replace = TRUE
))
mydata$sex <- factor(
  sex,
  levels = c("Male", "Female"),
  labels = c(1, 0)
)
maintask <- mlr3::TaskClassif$new(
  id = "my_classification_task",
  backend = mydata,
  target = target_col,
  positive = positive_class
)
splits <- mlr3::partition(maintask)
mylrn <- mlr3::lrn(
  "classif.ranger",
  predict_type = "prob"
)
mylrn$train(maintask, splits$train)
SHAP_output <- eSHAP_plot(
  task = maintask,
  trained_model = mylrn,
  splits = splits,
  sample.size = 2, # also 30 or more
  seed = seed,
  subset = 0.02 # up to 1
)
shap_Mean_long <- SHAP_output[[3]]
myplot <- ShapPartialPlot(shap_Mean_long)
```

Index