Package ‘DriveML’

June 14, 2021

Type Package

Title Self-Drive Machine Learning Projects

Version 0.1.3

Maintainer Dayanand Ubrangala <daya6489@gmail.com>

Depends R (>= 3.3.0)

Imports sampling(>= 2.8), rmarkdown, SmartEDA, data.table(>= 1.10.4-3), caTools, ParamHelpers(>= 1.12), mlr(>= 2.15.0), ggplot2(>= 2.2.1), iml


License GPL-3 | file LICENSE

Suggests testthat, knitr, ranger, glmnet, randomForest, rpart, xgboost, stats, graphics, tidyr, MASS

Encoding UTF-8

BugReports https://github.com/daya6489/DriveML/issues

LazyData true

Repository CRAN

RoxygenNote 7.1.1

VignetteBuilder knitr
autoDataprep

Automatic data preparation for ML algorithms

Description

Final data preparation before ML algorithms. Function provides final data set and highlights of the data preparation

Usage

autoDataprep(
  data,
  target = NULL,  # Target variable
  missimpute = "default",  # Method for imputing missing values
  auto_mar = FALSE,  # Whether to automatically impute MAR
  mar_object = NULL,  # Object containing MAR values
  dummyvar = TRUE,  # Whether to create dummy variables for factor variables
  char_var_limit = 12,  # Limit for character variables
  aucv = 0.02,  # AUC cut-off for variable selection
  corr = 0.99,  # Correlation threshold for variable selection
  outlier_flag = FALSE,  # Whether to detect outliers
  interaction_var = FALSE,  # Whether to create interaction variables
  frequent_var = FALSE  # Whether to select frequent variables
)

Index

autoDataprep Automatic data preparation for ML algorithms

2

autoMAR

5

autoMLmodel

6

autoMLReport

8

autoPDP

9

generateFeature

10

heart

12

heart.model

13

misspattern

13

predictAutoMAR

14

predictDataprep

15

2
uid = NULL,
onlykeep = NULL,
drop = NULL,
verbose = FALSE
)

Arguments

data [data.frame | Required] dataframe or data.table
target [integer | Required] dependent variable (binary or multiclass)
missimpute [text | Optional] missing value impuation using mlr misimpute function. Please refer to the "details" section to know more
auto_mar [character | Optional] identify any missing variable which are completely missing at random or not (default FALSE). If TRUE this will call autoMAR()
mar_object [character | Optional] object created from autoMAR function
dummyvar [logical | Optional] categorical feature engineering i.e. one hot encoding (default is TRUE)
char_var_limit [integer | Optional] default limit is 12 for a dummy variable preparation. e.g. if gender variable has two different value "M" and "F", then gender has 2 levels
aucv [integer | Optional] cut off value for AUC based variable selection
corr [integer | Optional] cut off value for correlation based variable selection
outlier_flag [logical | Optional] to add outlier features (default is FALSE)
interaction_var [logical | Optional] bulk interactions transformer for numerical features
frequent_var [logical | Optional] frequent transformer for categorical features
uid [character | Optional] unique identifier column if any to keep in the final data set
onlykeep [character | Optional] only consider selected variables for data preparation
drop [character | Optional] exclude variables from the dataset
verbose [logical | Optional] display executions steps on console(default is FALSE)

Details

Missing imputation using impute function from MLR
MLR package have a appropriate way to impute missing value using multiple methods. #'

* mean value for integer variable
* median value for numeric variable
* mode value for character or factor variable

optional: You might be interested to impute missing variable using ML method. List of algorithms will be handle missing variables in MLR package listLearners("classif", check.packages = TRUE, properties = "missings")[c("class", "package")]

Feature engineering
autoDataprep

- missing not completely at random variable using autoMAR function
- date transformer like year, month, quarter, week
- frequent transformer counts each categorical value in the dataset
- interaction transformer using multiplication
- one hot dummy coding for categorical value
- outlier flag and capping variable for numerical value

Feature reduction

- zero variance using nearZeroVar caret function
- pearson's correlation value
- auc with target variable

Value

list output contains below objects

complete_data complete dataset including new derived features based on the functional understanding of the dataset
master_data filtered dataset based on the input parameters
final_var_list list of master variables
auc_var list of auc variables
cor_var list of correlation variables
overall_var all variables in the dataset
zerovariance variables with zero variance in the dataset

See Also

impute

Examples

#Auto data prep
traindata <- autoDataprep(heart, target = "target_var", missimpute = "default",
dummyvar = TRUE, aucv = 0.02, corr = 0.98, outlier_flag = TRUE,
interaction_var = TRUE, frequent_var = TRUE)
train <- traindata$master
autoMAR Function to identify and generate the Missing at Random features (MAR)

Description

This function will automatically identify the missing patterns and flag the variables if they are not missing at random based on the AUC method

Usage

```r
autoMAR(
  data,
  aucv = 0.9,
  strataname = NULL,
  stratasize = NULL,
  mar_method = "glm"
)
```

Arguments

data [data.frame | Required] dataframe or data.table

aucv [integer | Optional] auc cut-off value for the not missing at random variable selection

strataname [text | Optional] vector of stratification variables

stratasize [integer | Optional] vector of stratum sample sizes (in the order in which the strata are given in the input dataset).

mar_method [text | Optional] missing at random classification method ("glm", "rf"). Default GLM is used (GLM runs faster for high dimensional data)

Value

list output including missing variable summary and number of MAR flag variables

Examples

```r
# create missing at random features
marobj <- autoMAR(heart, aucv = 0.9, strataname = NULL, stratasize = NULL, mar_method = "glm")
```
Description

Automated training, tuning and validation of machine learning models. Models are tuned, resampled and validated on an experimental dataset and trained on the full dataset and validated/tested on external datasets. Classification models tune the probability threshold automatically and returns the results. Each model contains information on performance, model object and evaluation plots.

Usage

```r
autoMLmodel(
  train,
  test = NULL,
  score = NULL,
  target = NULL,
  testSplit = 0.2,
  tuneIters = 10,
  tuneType = "random",
  models = "all",
  perMetric = "auc",
  varImp = 10,
  liftGroup = 50,
  maxObs = 10000,
  uid = NULL,
  pdp = FALSE,
  positive = 1,
  htmlreport = FALSE,
  seed = 1991,
  verbose = FALSE
)
```

Arguments

- **train** [data.frame | Required] training set
- **test** [data.frame | Optional] optional testing set to validate models on. If none is provided, one will be created internally. Default of NULL
- **score** [data.frame | Optional] optional score the models on best trained model based on AUC. If none is provided, scorelist will be null. Default of NULL
- **target** [integer | Required] if a target is provided classification or regression models will be trained, if left as NULL unsupervised models will be trained. Default of NULL
- **testSplit** [numeric | Optional] percentage of data to allocate to the test set. Stratified sampling is done. Default of 0.1
tuneIters [integer | Optional] number of tuning iterations to search for optimal hyper parameters. Default of 10

**tuneType** [character | Optional] tune method applied, list of options are:
- "random" - random search hyperparameter tuning
- "frace" - frace uses iterated f-racing algorithm for the best solution from irace package

**models** [character | Optional] which models to train. Default option is all. Please find below the names for each of the methods
- randomForest - random forests using the randomForest package
- ranger - random forests using the ranger package
- xgboost - gradient boosting using xgboost
- rpart - decision tree classification using rpart
- glmnet - regularised regression from glmnet
- logreg - logistic regression from stats

**perMetric** [character | Optional] model validation metric. Default is "auc"
- auc - area under the curve; mlr::auc
- accuracy - accuracy; mlr::acc
- balancedAccuracy - balanced accuracy; mlr::bac
- brier - brier score; mlr::brier
- f1 - F1 measure; mlr::f1
- meanPrecRecall - geometric mean of precision and recall; mlr::gpr
- logloss - logarithmic loss; mlr:logloss

**varImp** [integer | Optional] number of important features to plot
**liftGroup** [integer | Optional] lift value to validate the test model performance
**maxObs** [numeric | Optional] number of observations in the experiment training dataset on which models are trained, tuned and resampled. Default of 40,000. If the training dataset has less than 40k observations then all the observations will be used

**uid** [character | Optional] unique variables to keep in test output data
**pdp** [logical | Optional] partial dependence plot for important variables
**positive** [character | Optional] positive class for the target variable
**htmlreport** [logical | Optional] to view the model outcome in html format
**seed** [integer | Optional] random number seed for reproducible results
**verbose** [logical | Optional] display executions steps on console. Default is FALSE

**Details**

all the models trained using mlr train function, all of the functionality in mlr package can be applied to the autoMLmodel outcome

**autoMLmodel** provides below the information of the various machine learning classification models
- trainedModels - model level list output contains trained model object, hyper parameters, tuned data, test data, performance and Model plots
- results - summary of all trained model result like AUC, Precision, Recall, F1 score
- modelexp - model gain chart
- predicted_score - predicted score
- datasummary - summary of the input data

Value

List output contains trained models and results

See Also

mlr train caret train makeLearner tuneParams

Examples

# Run only Logistic regression model
mymodel <- autoMLmodel(train = heart, test = NULL, target = 'target_var',
testSplit = 0.2, tuneIters = 10, tuneType = "random", models = "logreg",
varImp = 10, liftGroup = 50, maxObs = 4000, uid = NULL, seed = 1991)

autoMLReport

Display autoMLmodel output in HTML format using Rmarkdown

Description

This function will generate R markdown report for DriveML model object

Usage

autoMLReport(mlobject, mldata = NULL, op_file = NULL, op_dir = NULL)

Arguments

mlobject [autoMLmodel Object | Required] autoMLmodel function output
mldata [autoDataprep Object | Optional] autoDataprep function output
op_file [character | Required] output file name (.html)
op_dir [character | Optional] output path. Default path is the current working directory

Details

Using this function we can easily present the model outcome in standard HTML format without writing Rmarkdown scripts

Value

HTML R Markdown output
Examples

```r
## Creating HTML report
autoMLReport(heart.model, mldata = NULL, op_file = "sample.html", op_dir = tempdir())
```

---

**Description**

Partial dependence plots (PDPs) help you to visualize the relationship between a subset of the features and the response while accounting for the average effect of the other predictors in the model. They are particularly effective with black box models like random forests, gradient boosting, etc.

**Usage**

```r
autoPDP(
  train,  # data.frame | Required]
  trainedModel,  # model object | Required]
  target,  # character | Optional]
  feature,  # character | Optional]
  sample = 0.5,  # numeric | Optional]
  modelname,  # character | Optional]
  seed = 1991  # integer | Optional]
)
```

**Arguments**

- **train**: [data.frame | Required] training sample used to train ML model
- **trainedModel**: [model object | Required] the object holding the machine learning model and the data
- **target**: [character | Optional] target variable name. Specify target variable if model object is other than MLR or driveML
- **feature**: [character | Optional] the feature name for which to compute the effects
- **sample**: [numeric | Optional] percentage of sample to be considered for training set for faster computation. Default of 0.5
- **modelname**: [character | Optional] specify which model to be plotted
- **seed**: [integer | Optional] random seed number. Default is 121

**Value**

List object containing a plot for each feature listed.
generateFeature

Automated column transformer

Description

This function automatically scans through each variable and generate features based on the type listed in the "details"

Usage

generateFeature(data, varlist, type = "Frequent", method = NULL)

Arguments

data [data.frame | Required] dataframe or data.table
varlist [text | Required] variable list to generate the additional features
type [text | Required] variable transformation with type - 'Dummy', 'Outlier', 'Frequent' or 'Interaction'
method [text | Required] input for variabe transformation for type = 'Frequent' then the method should be 'Frequency' or 'Percent'. Please refer to the "details" section to know more
generateFeature

Details

This function is for generating features based on different transformation methods such as interaction, outliers, Dummy coding, etc.

Interaction type

- multiply - multiplication
- add - addition
- subtract - substraction
- divide - division

Frequency type

- Frequency - frequency
- Percent - percentage

Outlier type

- Flag - flag outlier values like 1 or 0
- Capping - impute outlier value by 95th or 5th percentile value

Date type

- Year
- Month
- Quarter
- Week

Value

generated transformed features

Examples

# Generate interaction features
generateFeature(heart, varlist = c("cp", "chol", "trestbps"), type = "Interaction", method = "add")
generateFeature(heart, varlist = c("cp", "chol", "trestbps"), type = "Interaction", method = "multiply")

# Generate frequency features
generateFeature(heart, varlist = c("cp", "thal"), type = "Frequent", method = "Percent")
generateFeature(heart, varlist = c("cp", "thal"), type = "Frequent", method = "Frequency")
Description

This database contains 76 attributes, but all published experiments refer to using a subset of 14 of them. In particular, the Cleveland database is the only one that has been used by ML researchers to this date. The "goal" field refers to the presence of heart disease in the patient. It is integer valued from 0 (no presence) to 4.

Usage

heart

Format

A data frame with 303 rows and 14 variables:

- age  integer Age
- sex  integer Sex
- cp  integer chest pain type (4 values)
- trestbps  integer resting blood pressure
- chol  integer serum cholestoral in mg/dl
- fbs  integer fasting blood sugar > 120 mg/dl
- restecg  integer resting electrocardiographic results (values 0,1,2)
- thalach  integer maximum heart rate achieved
- exang  integer exercise induced angina
- oldpeak  double oldpeak = ST depression induced by exercise relative to rest
- slope  integer the slope of the peak exercise ST segment
- ca  integer number of major vessels (0-3) colored by flouroscopy
- thal  integer thal: 3 = normal; 6 = fixed defect; 7 = reversible defect
- target_var  integer the presence of heart disease in the patient. It is integer valued from 0 (no presence) to 4

Value

sample data

Examples

```r
## Load heart data
data(heart)
```
heart.model

Heart Classification Drive ML Model.

Description
Contains the task ('heart.model').

Usage
heart.model

Format
An object of class autoMLmodel of length 6.

Value
heart data driveML sample model output

Examples
## Sample model object
modelobj <- heart.model

misspattern

Missing pattern analysis for missing data

Description
This function is used to summarise the missing variable, missing pattern identification and classifying the columns based on the pattern of missing values.

Usage
misspattern(data, mfeature, drop = 0.99, print = FALSE)

Arguments
data [data.frame | Required] data set with missing values
mfeature [character | Required] only missing variable name
drop [numeric | optional] drop variable percentage. Example, if drop = 0.9, function will automatically drop 90per missing columns from the data set
print [character | optional] defualt print is FALSE
Value

final variable list, summary of missing data analysis

Examples

## Sample iris data
mdata <- iris
mobject <- misspattern(mdata, mfeature = c("Sepal.Length", "Petal.Length"), drop = 0.99, print = F)

predictAutoMAR(x, data, mar_var = NULL)

Arguments

x [autoMAR object | Required] autoMAR object for which prediction is desired
data [data.frame | Required] prediction data set to prepare the autoMAR outcomes
mar_var [character list | Optional] list of predefined mar variables

Value

flagged variables for missing not completely at random

Examples

## Missing at random features
train <- heart[1 : 199, ]
test <- heart[200 : 300, ]
marobj <- autoMAR(train, aucv = 0.9, strataname = NULL, stratasize = NULL, mar_method = "glm")

## print summary in console
testobj <- predictAutoMAR(marobj, test)
predictDataprep  
Extract predictions and generate columns from autoDataprep objects

Description
this function can be used for autoDataprep objects to generate the same for validation

Usage
predictDataprep(x, data)

Arguments
x  [autoDataprep object | Required] autoDataprep object for which prediction is desired
data  [data.frame | Required] prediction data set to prepare the MAR columns

Value
master data set same as train data set

Examples
## Sample train data set
train <- heart[1:200,]

## Predict same features for test set
predictDataprep(traindata, test)
Index

* datasets
  heart, 12

* mlmodel
  heart.model, 13

autoDataprep, 2
autoMAR, 5
autoMLmodel, 6
autoMLReport, 8
autoPDP, 9

caret train, 8
FeatureEffects, 10
generateFeature, 10

heart, 12
heart.model, 13

impute, 4

makeLearner, 8
misspattern, 13
mlr train, 8

partial, 10
plotPartialDependence, 10
predictAutoMAR, 14
predictDataprep, 15

tuneParams, 8