LHC Package Documentation

January 21, 2021

`add_rbf_features` *Add n_centroids RBF features to X*

Description

Add n_centroids RBF features to X

Usage

```
add_rbf_features(X, s, n_centroids, Xi = NULL)
```

Arguments

`X`	covariate matrix
`s`	median pairwise distance of points in X
`n_centroids`	number of RBF features to add
`Xi`	matrix of reference points to calculate RBF w.r.t

Value

X augmented covariate matrix

AMS_data *AMS data object class*

Description

This reference class object is used to store the AMS metric of a classification model at different decision thresholds. AMS is a performance measure which includes the sample weightings and is defined by the Higgs Boson Kaggle Competition. A vector of true sample classifications (0 or 1), a vector of estimated probabilities from a model, and a vector of scaled sample weights are needed to initialise.

Fields

`y` A vector of true sample classifications (0 or 1),

`prob` A vector of the samples estimated probabilities from a model

`weights` A vector of scaled sample weights.

`thresholds` A vector of 30 decision thresholds.

`ams` A vector of the AMS metric at each threshold.

`max_ams` maximum ams

`max_thresh` threshold of maximum ams

Methods

`calc_ams()` Calculate the AMS at each thresholds.

`initialize(y, prob, weights)` Provide true sample lables, estimated probabilities, and sample weights. A vector of descision thresholds is initalised.

`plot_ams()` Plot AMS against threshold.

ams_metric *Calculate AMS metric*

Description

$s = sum_{iinB \cup G} w_i$ $b = sum_{iinB \cup G} w_i$; i.e. s is the sum of the weights of successful signal classifications (TP) and b is the sum of the weights of incorrect signal classifications (FP)

Usage

```
ams_metric(s, b)
```

Arguments

`s`	count of true positives
`b`	count of false positives

`avg_median_pairwise_distance` *Calculate average of median pairwise distances for between all adjacent points*

Description

Calculate average of median pairwise distances for between all adjacent points

Usage

```
avg_median_pairwise_distance(X)
```

Arguments

`X`	covariate matrix

Value

s median pairwise distance

`backtrack_linesearch` *backtracking linesearch to find "optimal" step size*

Description

backtracking linesearch to find "optimal" step size

Usage

```
backtrack_linesearch(f, gradf, x, deltax, alpha, beta)
```

Arguments

`f`	function we're minimising
`gradf`	gradient of f
`x`	parameter we're optimising over
`deltax`	newton step
`alpha`	linesearch parameter
`beta`	linesearch update parameter

calc_K *Compute the kernel matrix over the (training) set X*

Description

Compute the kernel matrix over the (training) set X

Usage

```
calc_K(X, ckernel)
```

Arguments

X	covariate matrix (nxd)
ckernel	kernel function with args (x_i, x_j)

Value

K kernel matrix (nxn)

calculate_ams_partition *Calculate the AMS*

Description

Calculate the AMS

Usage

```
calculate_ams_partition(y, y_hat, w, sum_w = NULL)
```

Arguments

y	response vector
y_hat	predicted response vector
w	weights
sum_w	total sum of weights for renormalisation

Value

ams

count_b — *Count the number of false positives need to make sure dims of y, y_hat and w are the same*

Description

Count the number of false positives need to make sure dims of y, y_hat and w are the same

Usage

```
count_b(y, y_hat, w)
```

Arguments

y	response vector
y_hat	predicted response vector
w	weights

count_s — *Count the number of true positives need to make sure dims of y, y_hat and w are the same*

Description

Count the number of true positives need to make sure dims of y, y_hat and w are the same

Usage

```
count_s(y, y_hat, w)
```

Arguments

y	response vector
y_hat	predicted response vector
w	weights

decide *Thresholding function*

Description

Thresholding function

Usage

```
decide(p, thresh = 0.5)
```

Arguments

p	vector of probabilities
thresh	threshold over which we assign output of 1

generate_colours *Function to generate colours that are quite distinct*

Description

Function to generate colours that are quite distinct

Usage

```
generate_colours(ncolours)
```

Arguments

ncolours	number of colours we want

Value

vector of hex colours

`get_const_features` *find colnames for columns that are constant (e.g. all 1, -999, NA etc)*

Description

find colnames for columns that are constant (e.g. all 1, -999, NA etc)

Usage

```
get_const_features(X)
```

Arguments

`X`	matrix of covariates

Value

list of column names

`get_model_idx` *helper function to get the index corresponding to the model built on folds l !=k and for jet number j 1,2,3, ordering columns by fold and then with nesting on j i.e. first six cols are k=1, j=1,2,3; k=2, j=1,2,3; etc*

Description

helper function to get the index corresponding to the model built on folds l !=k and for jet number j 1,2,3, ordering columns by fold and then with nesting on j i.e. first six cols are k=1, j=1,2,3; k=2, j=1,2,3; etc

Usage

```
get_model_idx(j, k, K)
```

Arguments

`j`	this jet group
`k`	this fold
`K`	number of folds

get_rbf_centroids *Get reference points for RBF centroids*

Description

Get reference points for RBF centroids

Usage

```
get_rbf_centroids(X, n_centroids, idx = NULL)
```

Arguments

X	covariate matrix
n_centroids	number of RBF centroids
idx	[Optional] location of RBF centroid reference points

Value

list of Xi (centroid points) and idx (location of them)

get_subset_idx *Select indices of rows in x which correspond to values of labels, which can be multiple elements*

Description

Select indices of rows in x which correspond to values of labels, which can be multiple elements

Usage

```
get_subset_idx(x, labels)
```

Arguments

x	vector of labels
labels	reference labels to compare to e.g. x <- c("t", "b", "t", "v", "b"); labels <- c("t", "b") get_subset_idx(x, labels) = T, T, T, F, T

get_valid_cols *Gets the columns from header that aren't in features_to_rm*

Description

Gets the columns from header that aren't in features_to_rm

Usage

```
get_valid_cols(header, features_to_rm, j)
```

Arguments

header	list of column names e.g. names(X)
features_to_rm	list of feature names we want to remove
j	jet group

Value

index of columns to retain

idx_higgs_mass *Get boolean index for rows with missing/or not missing (depending on G/j) Higgs mass*

Description

Get boolean index for rows with missing/or not missing (depending on G/j) Higgs mass

Usage

```
idx_higgs_mass(X, j, G)
```

Arguments

X	matrix of covariates
j	jet group
G	number of jet groups

Value

vector of bools

idx_jet_cat *Get boolean vector of rows with j=0,1 or 2+*

Description

Get boolean vector of rows with j=0,1 or 2+

Usage

```
idx_jet_cat(nj, j)
```

Arguments

nj	Vector of number of jets for each point
j	jet group

import_data *Import raw LHC data*

Description

This function provides a standard way to load in the LHC dataset for our analysis pipeline. Undefined data (-999s) are replaces with NAs, columns are split into variables, labels and supplementary data.

Usage

```
import_data(filepath = "atlas-higgs-challenge-2014-v2.csv")
```

Arguments

filepath	str location of csv

Value

named list of X, y, w, kaggle_w, kaggle_s, e_id, nj

interior_point_fit *x contains lambda (dual var) as well*

Description

x contains lambda (dual var) as well

Usage

```
interior_point_fit(
  f,
  dualf,
  gradf,
  Hf,
  x,
  m,
  mu = 10,
  eps = 1e-06,
  eps_feas = 1e-06
)
```

Arguments

f	objective function
dualf	dual objective function
gradf	residual vector
Hf	Hessian for residual
x	primal-dual point
m	number of inequality constraints
mu	interior-point step parameter
eps	tolerance for problem
eps_feas	tolerance for feasibility of primal-dual points

invert_angle_sign *Invert Angle Sign*

Description

Uses the sign of the pseudorapidity of the tau particle to modify the sign of the pseudorapidity of the leptons and jets on the basis that the interaction should be invariant to rotations of pi about the beam (z) axis.

$$\eta(\theta) = -\log\tan\frac{\theta}{2}$$

$$\eta(\pi - \theta) = -\eta(\theta)$$

Usage

```
invert_angle_sign(X)
```

`inv_model_idx` *does the inverse procedure to get_model_idx, s.t. if idx <- get_model_idx(j, k, K) then (j,k) <- inv_model_idx(idx) (if R output tuples)*

Description

does the inverse procedure to get_model_idx, s.t. if idx <- get_model_idx(j, k, K) then (j,k) <- inv_model_idx(idx) (if R output tuples)

Usage

```
inv_model_idx(idx, K)
```

Arguments

`idx`	model index
`K`	number of folds

Value

numeric pair of j and k

`kernel_svm` *Kernel SVM*

Description

Fit a kernel support vector machine for binary classification.

Usage

```
kernel_svm(X, y, C, ckernel)
```

Arguments

`X`	An nxd matrix with samples as rows and features as columns.
`y`	A length-n vector of -1s and 1s indicating true sample classes.
`C`	Regularisation parameter.
`ckernel`	Kernel function with hyperparamters set

lin_kernel *Define linear kernel*

Description

Define linear kernel

Usage

```
lin_kernel(x_i, x_j)
```

Arguments

x_i	point in R^d
x_j	point in R^d

logisticf *Calculate logistic function*

Description

Calculate logistic function

Usage

```
logisticf(x)
```

Arguments

x	float

Value

logisticf(x)

logistic_model *Logistic model object class*

Description

This reference class object fits a binary classification model, using the logistic_reg function. The model can be used to predict the classes of new samples. The sample classes must be 0 or 1, and the prediction returns the estimated probabilities that each sample is class 1. A decision threshold should be subsequently.

Fields

`X` An nxd matrix with samples as rows and features as columns.

`y` A length-n vector of 0s and 1s indicating true sample classes.

`coeffs` A length-d vector of model coefficients.

`lambda` regularization parameter

Methods

`initialize(X, y, lambda = 1e-06)` Provide X and y and the coeffs field will be calculated using logistic_reg

`predict(X_test)` Provide a matrix of new samples and a vector of P(y=1) is returned

logistic_reg *Fit a logistic regression model by IRWLS - same thing glm does*

Description

Fit a logistic regression model by IRWLS - same thing glm does

Usage

```
logistic_reg(X, y, lambda = 0)
```

Arguments

`X`	covariate matrix
`y`	response vector
`lambda`	[Optional] L2 regularisation parameter
`r`	[Optional] weight vector

Value

b vector of coefficients

logit *Calculate logit function*

Description

Calculate logit function

Usage

```
logit(p)
```

Arguments

p	float in [0,1]

Value

logit(p)

newton_step *Compute newton step*

Description

Compute newton step

Usage

```
newton_step(grad, H)
```

Arguments

grad	gradient vector
H	Hessian matrix

Value

step to take

pairwise_distance *Calculate distance for each row of X0 and X1*

Description

Calculate distance for each row of X0 and X1

Usage

```
pairwise_distance(X0, X1)
```

Arguments

X0	covariate matrix
X1	covariate matrix

Value

vector of distances

partition_data *Partition data into (random) folds for cross-validation.*

Description

Partition data into (random) folds for cross-validation.

Usage

```
partition_data(n, k, random = FALSE)
```

Arguments

n	number of rows
k	number of folds
random	flag to choose whether to randomly select

Value

ind vector of integers denoting the OOS fold each row belongs to Returns vector of indices denoting the OOS index, i_e. for rows with I_i=1, those are OOS for i=1

permute_matrix *Cyclic permutation of rows of X by r rows*

Description

Cyclic permutation of rows of X by r rows

Usage

```
permute_matrix(X, r = 1)
```

Arguments

X	covariate matrix
r	number of rows to permute (default=1)

plot_amss *define a function to plot multiple ams objects on the same axes*

Description

define a function to plot multiple ams objects on the same axes

Usage

```
plot_amss(amss, title = "AMS data", min.max = TRUE, ...)
```

Arguments

amss	list of ams objects
title	str title to give plot

plot_rocs *Compute receiver operating characteristic (ROC) curve*

Description

Compute receiver operating characteristic (ROC) curve

Usage

```
plot_rocs(rocs, title = "ROC curves")
```

Arguments

rocs	list of roc objects
title	str title to give plot
FP	false positive rate (vector)
TP	true positive rate (vector)
y	response vector
p_hat	model output probabilities plots of parameter values by fold to compare and check consistency of models
b	matrix of coefficients (dxK) define a function to plot multiple roc objects on the same axes

Value

list of false positive and true positive rates at different thresholds Calculate the area under the ROC curve (AUC) as a metric of performance

AUC estimate (scalar) Plot ROC curve for particular model

poly_kernel *Define polynomial kernel*

Description

Define polynomial kernel

Usage

```
poly_kernel(x_i, x_j, b)
```

Arguments

x_i	point in R^d
x_j	point in R^d
b	order of polynomial

Value

scalar of (1+x^Tx)^b

`poly_transform` *Run polynomial transform on columns of X (of order b), removing output columns that are highly correlated*

Description

Run polynomial transform on columns of X (of order b), removing output columns that are highly correlated

Usage

```
poly_transform(X, b = 2)
```

Arguments

`X`	matrix of covariates
`b`	order of polynomial

Value

X augmented matrix of covariates

`rbf_feature` *Compute single RBF feature at some centroid i in idx (or xi in Xi)*

Description

Compute single RBF feature at some centroid i in idx (or xi in Xi)

Usage

```
rbf_feature(X, s, idx = NULL, xi = NULL)
```

Arguments

`X`	covariate matrix
`s`	median pairwise distance of points in X
`idx`	[Optional] location of reference centroid
`xi`	[Optional] reference centroid

`rbf_kernel` *RBF kernel*

Description

RBF kernel

Usage

```
rbf_kernel(x_i, x_j, sigma)
```

Arguments

`x_i`	point in R^d
`x_j`	point in R^d
`sigma`	bandwidth hyperparameter

`reduce_features` *Reduce feature space dimensionality by exploiting redundancy*

Description

Reduce feature space dimensionality by exploiting redundancy

Usage

```
reduce_features(X)
```

Arguments

`X`	matrix of covariates

Value

X augmented matrix of covariates

ROC_curve *ROC curve object class*

Description

This reference class object is used to plot a Receiver Operating Characteristic curve. An ROC curve is a performance measure of a classification model, created by plotting the true positive rate (TPR) against the false positive rate (FPR) as the decision threshold is varied. The object finds suitable thresholds, calculates FPR and TPR at each, and can calculate the Area Under the Curve (AUC). A vector of true sample classifications (0 or 1) and a vector of estimated probabilities from a model are needed to initialise.

Fields

`thresholds` A vector of 30 decision thresholds.

`FP` A vector of the false positive rate at each threshold.

`TP` A vector of the true positive rate at each threshold.

`auc` A numeric that is the area under the ROC curve.

Methods

`calc_auc()` If the AUC has not already be calculated, this calls the calculation.

`initialize(y, prob)` Provide sample labels and probabilites, and the FPR and TPR are calculated at 30 decision thresholds

`plot_curve()` Plot the ROC curve.

scale_dat *define a function to scale features of a matrix with reference to another matrix useful because you can normalise X_train, and apply the same transformation to X_test not designed for data with -999s!*

Description

define a function to scale features of a matrix with reference to another matrix useful because you can normalise X_train, and apply the same transformation to X_test not designed for data with -999s!

Usage

```
scale_dat(X, ref, na.rm = FALSE, add.intercept = TRUE)
```

Arguments

X	matrix of covariates
ref	matrix of covariates from which to calculate mu and sd
na.rm	flag to be compatible with colMeans and sd (to ignore NA)

Value

augmented matrix of covariates, standardized and an intercept column

---

set_features_to_rm *create list of feature names we want to omit based on jet group, or constant values/missing*

Description

create list of feature names we want to omit based on jet group, or constant values/missing

Usage

```
set_features_to_rm(X, G, kI, nj)
```

Arguments

X	matrix of covariates
G	number of jet groups
kI	fold indices (test label)
nj	Vector of number of jets for each point

Value

nested list of column names

---

svm *Soft margin SVM*

Description

Fit a soft margin support vector machine for binary classification.

Usage

```
svm(X, y, C = 1)
```

Arguments

X	An nxd matrix with samples as rows and features as columns.
y	A length-n vector of -1s and 1s indicating true sample classes.
C	Regularisation parameter.

---

trig_kernel *Define trigonometric kernel*

Description

Define trigonometric kernel

Usage

```
trig_kernel(x_i, x_j, b = 0)
```

Arguments

x_i	point in R^d
x_j	point in R^d
b	order of polynomial

---

tuned_kernel *Function factory to partially call kernel function to return the kernel function with it's hyperparameters set*

Description

Function factory to partially call kernel function to return the kernel function with it's hyperparameters set

Usage

```
tuned_kernel(ckernel, ...)
```

Arguments

ckernel	kernel function with args (x_i,x_j,hyper)

Value

function with args (x_i,x_j)

Index