

Package: maize (via r-universe)

October 14, 2026

Title Specialty Kernels for SVMs

Version 0.0.2.95

Description Bindings for SVMs and special kernels via kernlab, e1701, ebmc, and mildsvm extending the 'parsnip' package. Specifically related to kernels for support vector machines not available in parsnip. Package also includes interfaces for novel SVM preprocessors (recipes) and postprocessors (probably).

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports applicable, parsnip, probably, recipes, rlang, tibble, vctrs

Remotes frankiethull/qrsvm, tidymodels/workflows, tidymodels/tailor

Depends R (>= 4.4.1.0)

LazyData true

Suggests knitr, quarto, rmarkdown, qrsvm, quantregForest, ebmc, mildsvm, modeltime, timetk, workflows, kernlab, ggplot2, dplyr, tailor, tidyr

Config/Needs/website brand.yml

VignetteBuilder quarto, knitr

URL <https://frankiethull.github.io/maize/>

Config/pak/sysreqs cmake make libicu-dev libuv1-dev

Repository <https://frankiethull.r-universe.dev>

Date/Publication 2026-10-14 19:35:24 UTC

RemoteUrl <https://github.com/frankiethull/maize>

RemoteRef HEAD

RemoteSha e50678f6b4137f792af6157be59a37a8f6f29d0d

Contents

ada_boost_svm_rbf	2
apd_svm_novel_detection	3
arima_svm_laplace	4
autoplot.kcor_df	5
bag_svm_rbf	6
cal_estimate_svm	6
corn_data	8
int_conformal_quantile_svm	8
kcca_correlate	9
kqr_laplace	10
lssvm_laplace	10
misvm_orova_rbf	11
rus_boost_svm_rbf	11
rvm_laplace	12
score.apd_svm_novel_detection	12
step_kfa_laplace	13
step_kfm_nystrom	15
step_kha_laplace	16
step_kha_tanh	18
step_kpca_laplace	19
step_kpca_tanh	20
svm_anova_rbf	22
svm_bessel	22
svm_cauchy	23
svm_cossim	24
svm_fourier	24
svm_laplace	25
svm_sorensen	26
svm_spline	26
svm_string	27
svm_tanh	27
svm_tanimoto	28
svm_tstudent	29
svm_wavelet	29
Index	31

ada_boost_svm_rbf

AdaBoost SVM with Radial Basis Function Kernel

Description

AdaBoost SVM with Radial Basis Function Kernel

Usage

```
ada_boost_svm_rbf(  
  mode = "unknown",  
  engine = "ebmc",  
  num_learners = NULL,  
  imb_ratio = NULL  
)
```

Arguments

mode	classification
engine	ebmc's adam2 which uses e1701's svm
num_learners	how many weak learners should be ensembled via boosting
imb_ratio	major-minor class imbalance ratio

apd_svm_novel_detection

Fit an SVM novelty detection model to estimate an applicability domain.

Description

apd_svm_novel_detection() fits an 'one-svc' novelty detection model.

Usage

```
apd_svm_novel_detection(x, ...)  
  
## Default S3 method:  
apd_svm_novel_detection(x, ...)  
  
## S3 method for class 'data.frame'  
apd_svm_novel_detection(x, ...)  
  
## S3 method for class 'matrix'  
apd_svm_novel_detection(x, ...)  
  
## S3 method for class 'formula'  
apd_svm_novel_detection(formula, data, ...)  
  
## S3 method for class 'recipe'  
apd_svm_novel_detection(x, data, ...)
```

Arguments

x	Depending on the context: • A data frame of predictors. • A matrix of predictors. • A recipe specifying a set of preprocessing steps created from <code>recipes::recipe()</code>.
...	Options to pass to <code>kernlab::ksvm()</code> . Options should not include data.
formula	A formula specifying the predictor terms on the right-hand side. No outcome should be specified.
data	When a recipe or formula is used, data is specified as: • A data frame containing the predictors.

Value

A `apd_svm_novel_detection` object.

arima_svm_laplace	<i>General Interface for "Support Vector Machine" ARIMA Regression Models</i>
-------------------	---

Description

`arima_svm_laplace()` is a way to generate a *specification* of a time series model that uses SVMs to improve modeling errors (residuals) on Exogenous Regressors. It works with both "automated" ARIMA (`auto.arima`) and standard ARIMA (`arima`). The main algorithms are:

- Auto ARIMA + SVM Errors (`engine = auto_arima_svm_laplace`, default)
- ARIMA + SVM Errors (`engine = arima_svm_laplace`)

Usage

```
arima_svm_laplace(
  mode = "regression",
  seasonal_period = NULL,
  non_seasonal_ar = NULL,
  non_seasonal_differences = NULL,
  non_seasonal_ma = NULL,
  seasonal_ar = NULL,
  seasonal_differences = NULL,
  seasonal_ma = NULL,
  cost = NULL,
  margin = NULL,
  laplace_sigma = NULL
)
```

Arguments

mode	A single character string for the type of model. The only possible value for this model is "regression".
seasonal_period	A seasonal frequency. Uses "auto" by default. A character phrase of "auto" or time-based phrase of "2 weeks" can be used if a date or date-time variable is provided. See Fit Details below.
non_seasonal_ar	The order of the non-seasonal auto-regressive (AR) terms. Often denoted "p" in pdq-notation.
non_seasonal_differences	The order of integration for non-seasonal differencing. Often denoted "d" in pdq-notation.
non_seasonal_ma	The order of the non-seasonal moving average (MA) terms. Often denoted "q" in pdq-notation.
seasonal_ar	The order of the seasonal auto-regressive (SAR) terms. Often denoted "P" in PDQ-notation.
seasonal_differences	The order of integration for seasonal differencing. Often denoted "D" in PDQ-notation.
seasonal_ma	The order of the seasonal moving average (SMA) terms. Often denoted "Q" in PDQ-notation.
cost	A positive number for the cost of predicting a sample within or on the wrong side of the margin
margin	A positive number for the epsilon in the SVM insensitive loss function (regression only)
laplace_sigma	sigma parameter for laplacian
sample_size	number for the number (or proportion) of data that is exposed to the fitting routine.

autoplot.kcor_df	<i>visualize component x-y feature space</i>
------------------	--

Description

visualize component x-y feature space

Usage

```
## S3 method for class 'kcor_df'
autoplot(x)
```

Arguments

x a kcor_df object

bag_svm_rbf

RUS Bagged SVM with Radial Basis Function Kernel

Description

RUS Bagged SVM with Radial Basis Function Kernel

Usage

```
bag_svm_rbf(
  mode = "unknown",
  engine = "ebmc",
  num_learners = NULL,
  imb_ratio = NULL
)
```

Arguments

mode	classification
engine	ebmc's ub which uses e1701's svm
num_learners	how many weak learners should be ensembled via bagging
imb_ratio	major-minor class imbalance ratio

cal_estimate_svm

Uses a support vector regression model to calibrate numeric predictions

Description

Uses a support vector regression model to calibrate numeric predictions

Usage

```
cal_estimate_svm(
  .data,
  truth = NULL,
  estimate = dplyr::matches("^pred$"),
  smooth = TRUE,
  parameters = NULL,
  ...,
  .by = NULL
)

## S3 method for class 'data.frame'
cal_estimate_svm(
```

```

    .data,
    truth = NULL,
    estimate = dplyr::matches("^pred$"),
    smooth = TRUE,
    parameters = NULL,
    ...,
    .by = NULL
  )

## S3 method for class 'tune_results'
cal_estimate_svm(
  .data,
  truth = NULL,
  estimate = dplyr::matches("^pred$"),
  smooth = TRUE,
  parameters = NULL,
  ...
)

## S3 method for class 'grouped_df'
cal_estimate_svm(
  .data,
  truth = NULL,
  estimate = NULL,
  smooth = TRUE,
  parameters = NULL,
  ...
)

```

Arguments

<code>.data</code>	An ungrouped data.frame object, or <code>tune_results</code> object, that contains a prediction column.
<code>truth</code>	The column identifier for the observed outcome data (that is numeric). This should be an unquoted column name.
<code>estimate</code>	Column identifier for the predicted values
<code>smooth</code>	Applies to the svm models. It switches between a <code>polydotTRUE</code> , and <code>vanilladotFALSE</code> .
<code>parameters</code>	(Optional) An optional tibble of tuning parameter values that can be used to filter the predicted values before processing. Applies only to <code>tune_results</code> objects.
<code>...</code>	Additional arguments passed to the models or routines used to calculate the new predictions.

Details

This function uses existing modeling functions from other packages to create the calibration:

- `kernlab::ksvm()` with a "vanilladot" is used when `smooth` is set to `FALSE`

- `kernlab::ksvm()` with a "polydot" is used when `smooth` is set to `TRUE`

These methods estimate the relationship in the unmodified predicted values and then remove that trend when `cal_apply()` is invoked.

See Also

<https://www.tidymodels.org/learn/models/calibration/>,

corn_data	<i>Synthetic Corn Dataset for Corny Example</i>
-----------	---

Description

Synthetic Corn Dataset for Corny Example

Details

Asked Claude Sonnet for a corn data given the README story problem

Value

corn_data a tibble

Source

claude-3-5-sonnet-20240620

Examples

```
data(corn_data)
str(corn_data)
```

int_conformal_quantile_svm	<i>Prediction intervals via conformal inference and QRSVM</i>
----------------------------	---

Description

To compute quantiles, this function uses Quantile SVM instead of probably's "int_conformal_quantile" QRF or classic quantile regression.

Usage

```
int_conformal_quantile_svm(object, ...)

## S3 method for class 'workflow'
int_conformal_quantile_svm(object, train_data, cal_data, level = 0.95, ...)
```

Arguments

object	A fitted <code>workflows::workflow()</code> object.
...	Options to pass to <code>qrsvm::qrsvm()</code>
train_data, cal_data	Data frames with the <i>predictor and outcome data</i> . <code>train_data</code> should be the same data used to produce object and <code>cal_data</code> is used to produce predictions (and residuals). If the workflow used a recipe, these should be the data that were inputs to the recipe (and not the product of a recipe).
level	The confidence level for the intervals.

Details

based on the initial probably implementation with slight modification.

origin: https://github.com/tidymodels/probably/blob/HEAD/R/conformal_infer_quantile.R

for more information, visit: <https://probably.tidymodels.org>

Value

An object of class "int_conformal_quantile" containing the information to create intervals (which includes object). The `predict()` method is used to produce the intervals.

kcca_correlate	<i>Kernel Canonical Correlation Analysis</i>
----------------	--

Description

Computes the canonical correlation analysis in feature space. Kernel Canonical Correlation Analysis (KCCA) is a non-linear extension of CCA. Given two random variables (or datasets), KCCA aims at extracting the information which is shared by the two random variables (or datasets). more information found at [kernlab::kcca\(\)](#)

Usage

```
kcca_correlate(x, y = NULL, kernel = "rbfdot", gamma = 0.1, num_comp = 10, ...)
```

Arguments

x	variable or dataframe
y	variable or dataframe
kernel	a kernel to use
gamma	regularization parameter
num_comp	number of components
...	pass through args for kcca function

Value

A kernel canonical correlation analysis data frame kcor_df

kqr_laplace	<i>Laplacian Kernel Quantile Regression</i>
-------------	---

Description

laplacian kernel

Usage

```
kqr_laplace(  
  mode = "unknown",  
  engine = "kernlab",  
  cost = NULL,  
  tau = NULL,  
  laplace_sigma = NULL  
)
```

Arguments

- | | |
|---------------|---|
| mode | regression |
| engine | kernlab kqr |
| cost | A positive number for the cost of predicting a sample within or on the wrong side of the margin |
| tau | a quantile for the loss function |
| laplace_sigma | sigma parameter for laplacian |

lssvm_laplace	<i>Laplacian Least Squares Support Vector Machine</i>
---------------	---

Description

laplacian kernel

Usage

```
lssvm_laplace(mode = "unknown", engine = "kernlab", laplace_sigma = NULL)
```

Arguments

- | | |
|---------------|-------------------------------|
| mode | classification |
| engine | kernlab lssvm |
| laplace_sigma | sigma parameter for laplacian |

misvm_orova_rbf	<i>Multiple Instance SVMs for Ordinal Outcome Data, One-Vs-All, with Radial Basis Function Kernel</i>
-----------------	---

Description

Multiple Instance SVMs for Ordinal Outcome Data, One-Vs-All, with Radial Basis Function Kernel

Usage

```
misvm_orova_rbf(
  mode = "unknown",
  engine = "ebmc",
  cost = NULL,
  rbf_sigma = NULL
)
```

Arguments

mode	classification
engine	mildsvm::misvm_orova() which uses e1701's svm
cost	A positive number for the cost of predicting a sample within or on the wrong side of the margin
rbf_sigma	A positive number for radial basis function.

rus_boost_svm_rbf	<i>RUSBoost SVM with Radial Basis Function Kernel</i>
-------------------	---

Description

RUSBoost SVM with Radial Basis Function Kernel

Usage

```
rus_boost_svm_rbf(
  mode = "unknown",
  engine = "ebmc",
  num_learners = NULL,
  imb_ratio = NULL
)
```

Arguments

mode	classification
engine	ebmc's rus which uses e1701's svm
num_learners	how many weak learners should be ensembled via boosting
imb_ratio	major-minor class imbalance ratio

rvm_laplace	<i>Laplacian Relevance Vector Machine (Experimental RVM)</i>
-------------	--

Description

laplacian kernel

Usage

```
rvm_laplace(
  mode = "unknown",
  engine = "kernlab",
  alpha = NULL,
  var = NULL,
  laplace_sigma = NULL
)
```

Arguments

mode	regression only for RVM
engine	kernlab rvm
alpha	(alpha) The initial alpha value or vector. Can be either a vector of length equal to the number of data points or a single number.
var	(var) the initial noise variance
laplace_sigma	sigma parameter for laplacian

score.apd_svm_novel_detection	<i>Predict from a apd_svm_novel_detection</i>
-------------------------------	---

Description

Predict from a apd_svm_novel_detection

Usage

```
## S3 method for class 'apd_svm_novel_detection'
score(object, new_data, type = "numeric", ...)
```

Arguments

object	A apd_svm_novel_detection object.
new_data	A data frame or matrix of new samples.
type	A single character. The type of predictions to generate. Valid options are: • "numeric" for numeric predictions.
...	Not used, but required for extensibility.

Details

About the score

Value

A tibble of predictions. The number of rows in the tibble is guaranteed to be the same as the number of rows in new_data. The score column is the raw prediction from `kernlab::predict()` while score_pctl compares this value to the reference distribution of the score created by predicting the training set. A value of *X* means that *X* percent of the training data have scores less than the predicted value.

See Also

[apd_svm_novel_detection\(\)](#)

step_kfa_laplace	<i>Laplacian function kernel KFA signal extraction</i>
------------------	--

Description

step_kfa_laplace() creates a *specification* of a recipe step that will convert numeric data into one or more kernel components using a laplace kernel. similar to KPCA, but instead of extracting eigenvectors of the dataset in feature space, it approximates the eigenvectors by selecting patterns which are good basis vectors for the dataset.

Usage

```
step_kfa_laplace(
  recipe,
  ...,
  role = "predictor",
  trained = FALSE,
  num_comp = 5,
  res = NULL,
  columns = NULL,
  sigma = 0.2,
  prefix = "kFA",
  keep_original_cols = FALSE,
```

```

    skip = FALSE,
    id = rand_id("kfa_laplace")
  )

```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose variables for this step. See selections() for more details.
role	For model terms created by this step, what analysis role should they be assigned? By default, the new columns created by this step from the original variables will be used as <i>predictors</i> in a model.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
num_comp	The number of components to retain as new predictors. If num_comp is greater than the number of columns or the number of possible components, a smaller value will be used. If num_comp = 0 is set then no transformation is done and selected variables will stay unchanged, regardless of the value of keep_original_cols.
res	An S4 kernlab::kfa() object is stored here once this preprocessing step has been trained by prep() .
columns	A character string of the selected variable names. This field is a placeholder and will be populated once prep() is used.
sigma	A numeric value for the laplace function parameter.
prefix	A character string for the prefix of the resulting new variables. See notes below.
keep_original_cols	A logical to keep the original variables in the output. Defaults to FALSE.
skip	A logical. Should the step be skipped when the recipe is baked by bake() ? While all operations are baked when prep() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations.
id	A character string that is unique to this step to identify it.

See Also

Other multivariate transformation steps: [step_kfm_nystrom\(\)](#), [step_kha_laplace\(\)](#), [step_kha_tanh\(\)](#), [step_kpca_laplace\(\)](#), [step_kpca_tanh\(\)](#)

step_kfm_nystrom	<i>Nystrom kernel feature map approximation (RBF)</i>
------------------	---

Description

step_kfm_nystrom() creates a *specification* of a recipe step that will convert numeric data into a feature approximation. nystrom approximates the 'radial' kernel approximation.

Usage

```
step_kfm_nystrom(
  recipe,
  ...,
  role = "predictor",
  trained = FALSE,
  res = NULL,
  columns = NULL,
  sigma = 0.2,
  m = NULL,
  r = NULL,
  sampling = "random",
  prefix = "kFM",
  keep_original_cols = FALSE,
  skip = FALSE,
  id = rand_id("kfm_nystrom")
)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose variables for this step. See selections() for more details.
role	For model terms created by this step, what analysis role should they be assigned? By default, the new columns created by this step from the original variables will be used as <i>predictors</i> in a model.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
res	An <code>mildsvm::kfm_nystrom()</code> object is stored here once this preprocessing step has been trained by prep() .
columns	A character string of the selected variable names. This field is a placeholder and will be populated once prep() is used.
sigma	A numeric value for the nystrom function parameter.
m	The number rows from df to sample in fitting. defaults to nrow of data
r	The rank of matrix approximation to use. Must be less than or equal to m, the default.

prefix	A character string for the prefix of the resulting new variables. See notes below.
keep_original_cols	A logical to keep the original variables in the output. Defaults to FALSE.
skip	A logical. Should the step be skipped when the recipe is baked by <code>bake()</code> ? While all operations are baked when <code>prep()</code> is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using <code>skip = TRUE</code> as it may affect the computations for subsequent operations.
id	A character string that is unique to this step to identify it.

See Also

Other multivariate transformation steps: `step_kfa_laplace()`, `step_kha_laplace()`, `step_kha_tanh()`, `step_kpca_laplace()`, `step_kpca_tanh()`

step_kha_laplace	<i>Laplacian function kernel PCA signal extraction via Hebbian Algorithm</i>
------------------	--

Description

`step_kha_laplace()` creates a *specification* of a recipe step that will convert numeric data into one or more principal components using a laplace kernel basis expansion.

Usage

```
step_kha_laplace(
  recipe,
  ...,
  role = "predictor",
  trained = FALSE,
  num_comp = 5,
  res = NULL,
  columns = NULL,
  sigma = 0.2,
  learn_rate = 0.005,
  threshold = 1e-04,
  stop_iter = 100,
  prefix = "kha",
  keep_original_cols = FALSE,
  skip = FALSE,
  id = rand_id("kha_laplace")
)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose variables for this step. See selections() for more details.
role	For model terms created by this step, what analysis role should they be assigned? By default, the new columns created by this step from the original variables will be used as <i>predictors</i> in a model.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
num_comp	The number of components to retain as new predictors. If num_comp is greater than the number of columns or the number of possible components, a smaller value will be used. If num_comp = 0 is set then no transformation is done and selected variables will stay unchanged, regardless of the value of keep_original_cols.
res	An S4 kernlab::kha() object is stored here once this preprocessing step has been trained by prep() .
columns	A character string of the selected variable names. This field is a placeholder and will be populated once prep() is used.
sigma	A numeric value for the laplace function parameter.
learn_rate	hebbian learning rate
threshold	the smallest value of the convergence step
stop_iter	maximum number of iterations
prefix	A character string for the prefix of the resulting new variables. See notes below.
keep_original_cols	A logical to keep the original variables in the output. Defaults to FALSE.
skip	A logical. Should the step be skipped when the recipe is baked by bake() ? While all operations are baked when prep() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations.
id	A character string that is unique to this step to identify it.

See Also

Other multivariate transformation steps: [step_kfa_laplace\(\)](#), [step_kfm_nystrom\(\)](#), [step_kha_tanh\(\)](#), [step_kpca_laplace\(\)](#), [step_kpca_tanh\(\)](#)

step_kha_tanh	<i>tanh function kernel PCA signal extraction via Hebbian Algorithm</i>
---------------	---

Description

step_kha_tanh() creates a *specification* of a recipe step that will convert numeric data into one or more principal components using a tanh kernel basis expansion.

Usage

```
step_kha_tanh(
  recipe,
  ...,
  role = "predictor",
  trained = FALSE,
  num_comp = 5,
  res = NULL,
  columns = NULL,
  scale_factor = 0.2,
  learn_rate = 0.005,
  threshold = 1e-06,
  stop_iter = 100,
  prefix = "kha",
  keep_original_cols = FALSE,
  skip = FALSE,
  id = rand_id("kha_tanh")
)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose variables for this step. See selections() for more details.
role	For model terms created by this step, what analysis role should they be assigned? By default, the new columns created by this step from the original variables will be used as <i>predictors</i> in a model.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
num_comp	The number of components to retain as new predictors. If num_comp is greater than the number of columns or the number of possible components, a smaller value will be used. If num_comp = 0 is set then no transformation is done and selected variables will stay unchanged, regardless of the value of keep_original_cols.
res	An S4 kernlab::kha() object is stored here once this preprocessing step has been trained by prep() .
columns	A character string of the selected variable names. This field is a placeholder and will be populated once prep() is used.

scale_factor	A numeric value for the tanh function parameter.
learn_rate	hebbian learning rate
threshold	the smallest value of the convergence step
stop_iter	maximum number of iterations
prefix	A character string for the prefix of the resulting new variables. See notes below.
keep_original_cols	A logical to keep the original variables in the output. Defaults to FALSE.
skip	A logical. Should the step be skipped when the recipe is baked by <code>bake()</code> ? While all operations are baked when <code>prep()</code> is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using <code>skip = TRUE</code> as it may affect the computations for subsequent operations.
id	A character string that is unique to this step to identify it.

See Also

Other multivariate transformation steps: [step_kfa_laplace\(\)](#), [step_kfm_nystrom\(\)](#), [step_kha_laplace\(\)](#), [step_kpca_laplace\(\)](#), [step_kpca_tanh\(\)](#)

step_kpca_laplace	<i>Laplacian function kernel PCA signal extraction</i>
-------------------	--

Description

`step_kpca_laplace()` creates a *specification* of a recipe step that will convert numeric data into one or more principal components using a laplace kernel

Usage

```
step_kpca_laplace(
  recipe,
  ...,
  role = "predictor",
  trained = FALSE,
  num_comp = 5,
  res = NULL,
  columns = NULL,
  sigma = 0.2,
  prefix = "kPC",
  keep_original_cols = FALSE,
  skip = FALSE,
  id = rand_id("kpca_laplace")
)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose variables for this step. See selections() for more details.
role	For model terms created by this step, what analysis role should they be assigned? By default, the new columns created by this step from the original variables will be used as <i>predictors</i> in a model.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
num_comp	The number of components to retain as new predictors. If num_comp is greater than the number of columns or the number of possible components, a smaller value will be used. If num_comp = 0 is set then no transformation is done and selected variables will stay unchanged, regardless of the value of keep_original_cols.
res	An S4 kernlab::kpca() object is stored here once this preprocessing step has been trained by prep() .
columns	A character string of the selected variable names. This field is a placeholder and will be populated once prep() is used.
sigma	A numeric value for the laplace function parameter.
prefix	A character string for the prefix of the resulting new variables. See notes below.
keep_original_cols	A logical to keep the original variables in the output. Defaults to FALSE.
skip	A logical. Should the step be skipped when the recipe is baked by bake() ? While all operations are baked when prep() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations.
id	A character string that is unique to this step to identify it.

See Also

Other multivariate transformation steps: [step_kfa_laplace\(\)](#), [step_kfm_nystrom\(\)](#), [step_kha_laplace\(\)](#), [step_kha_tanh\(\)](#), [step_kpca_tanh\(\)](#)

step_kpca_tanh

Laplacian function kernel PCA signal extraction

Description

step_kpca_tanh() creates a *specification* of a recipe step that will convert numeric data into one or more principal components using a tanh kernel basis expansion.

Usage

```
step_kpca_tanh(
  recipe,
  ...,
  role = "predictor",
  trained = FALSE,
  num_comp = 5,
  res = NULL,
  columns = NULL,
  scale_factor = 0.2,
  offset = 0,
  prefix = "kPC",
  keep_original_cols = FALSE,
  skip = FALSE,
  id = rand_id("kpca_tanh")
)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose variables for this step. See selections() for more details.
role	For model terms created by this step, what analysis role should they be assigned? By default, the new columns created by this step from the original variables will be used as <i>predictors</i> in a model.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
num_comp	The number of components to retain as new predictors. If num_comp is greater than the number of columns or the number of possible components, a smaller value will be used. If num_comp = 0 is set then no transformation is done and selected variables will stay unchanged, regardless of the value of keep_original_cols.
res	An S4 <code>kernlab::kpca()</code> object is stored here once this preprocessing step has been trained by prep() .
columns	A character string of the selected variable names. This field is a placeholder and will be populated once prep() is used.
scale_factor	A numeric value for the tanh function parameter.
prefix	A character string for the prefix of the resulting new variables. See notes below.
keep_original_cols	A logical to keep the original variables in the output. Defaults to FALSE.
skip	A logical. Should the step be skipped when the recipe is baked by bake() ? While all operations are baked when prep() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations.
id	A character string that is unique to this step to identify it.

See Also

Other multivariate transformation steps: [step_kfa_laplace\(\)](#), [step_kfm_nystrom\(\)](#), [step_kha_laplace\(\)](#), [step_kha_tanh\(\)](#), [step_kpca_laplace\(\)](#)

svm_anova_rbf	<i>ANOVA RBF Support Vector Machine</i>
---------------	---

Description

anova rbf for support vector machines

Usage

```
svm_anova_rbf(  
  mode = "unknown",  
  engine = "kernlab",  
  cost = NULL,  
  anova_rbf_sigma = NULL,  
  degree = NULL,  
  margin = NULL  
)
```

Arguments

mode	regression or classification
engine	kernlab ksvm
cost	A positive number for the cost of predicting a sample within or on the wrong side of the margin
anova_rbf_sigma	sigma parameter for anova rbf
degree	degree parameter for anova rbf
margin	A positive number for the epsilon in the SVM insensitive loss function (regression only)

svm_bessel	<i>Bessel Support Vector Machine</i>
------------	--------------------------------------

Description

bessel kernel for support vector machines

Usage

```
svm_bessel(
  mode = "unknown",
  engine = "kernlab",
  cost = NULL,
  bessel_sigma = NULL,
  degree = NULL,
  order = NULL,
  margin = NULL
)
```

Arguments

mode	regression or classification
engine	kernlab ksvm
cost	A positive number for the cost of predicting a sample within or on the wrong side of the margin
bessel_sigma	sigma parameter for bessel
degree	degree parameter for bessel
order	order parameter for bessel
margin	A positive number for the epsilon in the SVM insensitive loss function (regression only)

svm_cauchy

*Cauchy Support Vector Machine***Description**

cauchy kernel for support vector machines

Usage

```
svm_cauchy(
  mode = "unknown",
  engine = "kernlab",
  cost = NULL,
  margin = NULL,
  sigma = NULL
)
```

Arguments

mode	regression or classification
engine	kernlab ksvm
cost	A positive number for the cost of predicting a sample within or on the wrong side of the margin
margin	A positive number for the epsilon in the SVM insensitive loss function (regression only)
sigma	a sigma parameter for cauchy kernels

svm_cossim	<i>Cosine Similarity Support Vector Machine</i>
------------	---

Description

cossim kernel for support vector machines

Usage

```
svm_cossim(mode = "unknown", engine = "kernlab", cost = NULL, margin = NULL)
```

Arguments

mode	regression or classification
engine	kernlab ksvm
cost	A positive number for the cost of predicting a sample within or on the wrong side of the margin
margin	A positive number for the epsilon in the SVM insensitive loss function (regression only)

svm_fourier	<i>Fourier Support Vector Machine</i>
-------------	---------------------------------------

Description

fourier kernel for support vector machines

Usage

```
svm_fourier(
  mode = "unknown",
  engine = "kernlab",
  cost = NULL,
  margin = NULL,
  sigma = NULL
)
```

Arguments

mode	regression or classification
engine	kernlab ksvm
cost	A positive number for the cost of predicting a sample within or on the wrong side of the margin
margin	A positive number for the epsilon in the SVM insensitive loss function (regression only)
sigma	a sigma parameter for fourier kernels

`svm_laplace`*Laplacian Support Vector Machine*

Description

laplacian kernel for support vector machines

Usage

```
svm_laplace(  
  mode = "unknown",  
  engine = "kernlab",  
  cost = NULL,  
  margin = NULL,  
  laplace_sigma = NULL  
)
```

Arguments

mode	regression or classification
engine	kernlab ksvm
cost	A positive number for the cost of predicting a sample within or on the wrong side of the margin
margin	A positive number for the epsilon in the SVM insensitive loss function (regression only)
laplace_sigma	sigma parameter for laplacian

`svm_sorensen`*Sorensen Support Vector Machine*

Description

sorensen kernel for support vector machines which is used as a graph kernel for chemical informatics

Usage

```
svm_sorensen(mode = "unknown", engine = "kernlab", cost = NULL, margin = NULL)
```

Arguments

mode	regression or classification
engine	kernlab ksvm
cost	A positive number for the cost of predicting a sample within or on the wrong side of the margin
margin	A positive number for the epsilon in the SVM insensitive loss function (regression only)

`svm_spline`*Spline Support Vector Machine*

Description

spline kernel for support vector machines

Usage

```
svm_spline(mode = "unknown", engine = "kernlab", cost = NULL, margin = NULL)
```

Arguments

mode	regression or classification
engine	kernlab ksvm
cost	A positive number for the cost of predicting a sample within or on the wrong side of the margin
margin	A positive number for the epsilon in the SVM insensitive loss function (regression only)

svm_string	<i>String Support Vector Machine</i>
------------	--------------------------------------

Description

stringdot for support vector machines

Usage

```
svm_string(  
  x,  
  y,  
  mode = "unknown",  
  engine = "kernlab",  
  cost = NULL,  
  length = NULL,  
  lambda = NULL,  
  normalized = TRUE,  
  margin = NULL  
)
```

Arguments

mode	regression or classification
engine	kernlab ksvm
cost	A positive number for the cost of predicting a sample within or on the wrong side of the margin
length	The length of the substrings considered
lambda	The decay factor
normalized	normalize string kernel values, (default = TRUE)
margin	A positive number for the epsilon in the SVM insensitive loss function (regression only)

svm_tanh	<i>Hyperbolic Tangent Support Vector Machine</i>
----------	--

Description

tanh kernel for support vector machines

Usage

```
svm_tanh(  
  mode = "unknown",  
  engine = "kernlab",  
  cost = NULL,  
  scale_factor = NULL,  
  margin = NULL  
)
```

Arguments

mode	regression or classification
engine	kernlab ksvm
cost	A positive number for the cost of predicting a sample within or on the wrong side of the margin
scale_factor	scale parameter for tanh
margin	A positive number for the epsilon in the SVM insensitive loss function (regression only)

svm_tanimoto

Tanimoto Support Vector Machine

Description

tanimoto kernel for support vector machines which is used as a graph kernel for chemical informatics

Usage

```
svm_tanimoto(mode = "unknown", engine = "kernlab", cost = NULL, margin = NULL)
```

Arguments

mode	regression or classification
engine	kernlab ksvm
cost	A positive number for the cost of predicting a sample within or on the wrong side of the margin
margin	A positive number for the epsilon in the SVM insensitive loss function (regression only)

`svm_tstudent`*T-Student Support Vector Machine*

Description

t-student kernel for support vector machines

Usage

```
svm_tstudent(  
  mode = "unknown",  
  engine = "kernlab",  
  cost = NULL,  
  margin = NULL,  
  degree = NULL  
)
```

Arguments

mode	regression or classification
engine	kernlab ksvm
cost	A positive number for the cost of predicting a sample within or on the wrong side of the margin
margin	A positive number for the epsilon in the SVM insensitive loss function (regression only)
degree	a degree parameter for tstudent kernels

`svm_wavelet`*Wavelet Support Vector Machine*

Description

wavelet kernel for support vector machines

Usage

```
svm_wavelet(  
  mode = "unknown",  
  engine = "kernlab",  
  cost = NULL,  
  margin = NULL,  
  sigma = NULL,  
  a = 1,  
  c = NULL,  
  h = NULL  
)
```

Arguments

mode	regression or classification
engine	kernlab ksvm
cost	A positive number for the cost of predicting a sample within or on the wrong side of the margin
margin	A positive number for the epsilon in the SVM insensitive loss function (regression only)
sigma	sigma parameter for svm wavelet kernel
a	scale adjustment parameter for wavelet kernels (temp name)
c	dist adjustment parameter for wavelet kernels can be NULL (temp name)
h	wavelet function for wavelet kernel, default wavelet if NULL (temp name)

Index

- * **datasets**
 - corn_data, 8
- * **multivariate transformation steps**
 - step_kfa_laplace, 13
 - step_kfm_nystrom, 15
 - step_kha_laplace, 16
 - step_kha_tanh, 18
 - step_kpca_laplace, 19
 - step_kpca_tanh, 20
- ada_boost_svm_rbf, 2
- apd_svm_novel_detection, 3
- apd_svm_novel_detection(), 13
- arima_svm_laplace, 4
- autoplot.kcor_df, 5
- bag_svm_rbf, 6
- bake(), 14, 16, 17, 19–21
- cal_apply(), 8
- cal_estimate_svm, 6
- corn_data, 8
- int_conformal_quantile_svm, 8
- kcca_correlate, 9
- kernlab::kcca(), 9
- kernlab::kfa(), 14
- kernlab::kha(), 17, 18
- kernlab::kpca(), 20, 21
- kernlab::ksvm(), 4, 7, 8
- kernlab::predict(), 13
- kqr_laplace, 10
- lssvm_laplace, 10
- mildsvm::kfm_nystrom(), 15
- misvm_orova_rbf, 11
- prep(), 14–21
- qrsvm::qrsvm(), 9
- recipes::recipe(), 4
- rus_boost_svm_rbf, 11
- rvm_laplace, 12
- score.apd_svm_novel_detection, 12
- selections(), 14, 15, 17, 18, 20, 21
- step_kfa_laplace, 13, 16, 17, 19, 20, 22
- step_kfm_nystrom, 14, 15, 17, 19, 20, 22
- step_kha_laplace, 14, 16, 16, 19, 20, 22
- step_kha_tanh, 14, 16, 17, 18, 20, 22
- step_kpca_laplace, 14, 16, 17, 19, 19, 22
- step_kpca_tanh, 14, 16, 17, 19, 20, 20
- svm_anova_rbf, 22
- svm_bessel, 22
- svm_cauchy, 23
- svm_cossim, 24
- svm_fourier, 24
- svm_laplace, 25
- svm_sorensen, 26
- svm_spline, 26
- svm_string, 27
- svm_tanh, 27
- svm_tanimoto, 28
- svm_tstudent, 29
- svm_wavelet, 29
- workflows::workflow(), 9