Package 'feamiR'

January 19, 2021

Type Package

Title Classification and Feature Selection for microRNA/mRNA Interactions

Version 0.1.0

Maintainer Eleanor Williams <ecw63@cam.ac.uk>

Description Comprises a pipeline for predicting microRNA/mRNA interactions, as detailed in Williams, Calinescu, Mohorianu (2020) <doi:10.1101/2020.12.23.424130>. Its input consists of [a] a messenger RNA (mRNA) dataset (either in fasta format, focused on 3' UTRs or in gtf format; for the latter, the sequences of the 3' UTRs are generated using the genomic coordinates), [b] a microRNA dataset (in fasta format, retrieved from miRBase, <http://www.mirbase.org/>) and [c] an interaction dataset (in csv format, from miRTarBase <http://mirtarbase.cuhk.edu.cn/php/index.php>). To characterise and predict microRNA/mRNA interactions, we use [a] statistical analyses based on Chi-squared and Fisher exact tests and [b] Machine Learning classifiers (decision trees, random forests and support vector machines). To enhance the accuracy of the classifiers we also employ feature selection approaches used in on conjunction with the classifiers. The feature selection approaches include a voting scheme for decision trees, a measure based on Gini index for random forests, forward feature selection and Genetic Algorithms on SVMs. The pipeline also includes a novel approach based on embryonic Genetic Algorithms which combines and optimises the forward feature selection and Genetic Algorithms. All analyses, including the classification and feature selection, are applicable on the microRNA seed features (default), on the full microRNA features and/or flanking features on the mRNA. The sets of features can be combined.

Encoding UTF-8

Depends R (>= 3.1.2)

Imports stringr, randomForest, rpart, rpart.plot, GA, e1071, ggplot2, magrittr, tibble, dplyr, reticulate

Config/reticulate list(packages = list(list(package = ``os''), list(package = ``argparse''), list(package = ``gzip''), list(package = ``pandas''), list(package = ``numpy''), list(package = ``math''), list(package = ``scipy.stats''), list(package = ``matplotlib.pyplot''), list(package = ``seaborn''), list(package = ``statistics''), list(package = ``logging''), list(package = ``Bio'')))

Suggests parallel, doParallel

SystemRequirements Python (>=3.6) sreformat patman

URL https://github.com/Core-Bioinformatics/feamiR

BugReports https://github.com/Core-Bioinformatics/feamiR/issues

LazyData true

RoxygenNote 7.1.1.9000

License GPL-2

NeedsCompilation no

Author Eleanor Williams [aut, cre],
Irina Mohorianu [aut]

Repository CRAN

Date/Publication 2021-01-19 08:30:02 UTC

R topics documented:

decisiontree	*Decision tree Trains a decision on the given training dataset and uses it to predict classification for test dataset. The resulting accuracy, sensitivity and specificity are returned, as well as a tree summary.*

Description

Decision tree Trains a decision on the given training dataset and uses it to predict classification for test dataset. The resulting accuracy, sensitivity and specificity are returned, as well as a tree summary.

Usage

```
decisiontree(data_train, data_test, includeplot = FALSE, showtree = FALSE)
```

Arguments

data_train	Training set: dataframe containing classification column and all other columns features. This is the dataset on which the decision tree model is trained.
data_test	Test set: dataframe containing classification column and all other columns features. This is the dataset on which the decision tree model in tested.
includeplot	Show performance scatter plot (default:FALSE)
showtree	Show trained decision tree graphically (default:FALSE)

Value

List containing performance summary, accessed using training (training accuracy), test (test accuracy), trainsensitivity, testsensitivity, trainspecificity, testspecificity. Also accessed using fit is the trained model produced. This can be used to find the features which appear at each level of the tree.

Examples

```
data_train = data.frame(
      classification=as.factor(c(1,1,0,0,1,1,0,0,1,1)),
      A=c(1,1,1,0,0,0,1,1,1,0),
      B=c(0,1,1,0,1,1,0,1,1,0),
      C=c(0,0,1,0,0,1,0,0,1,0))
data_test = data.frame(
      classification=as.factor(c(1,1,0,0,1,1,1,0)),
      A=c(0,0,0,1,0,0,0,1),
      B=c(1,1,1,0,0,1,1,1),
      C=c(0,0,1,1,0,0,1,1))
decisiontree(data_train,data_test)
```

dtreevoting *Decision tree voting scheme. Implements a feature selection approach based on Decision Trees, using a voting scheme across the top levels on trees trained on multiple subsamples.*

Description

Decision tree voting scheme. Implements a feature selection approach based on Decision Trees, using a voting scheme across the top levels on trees trained on multiple subsamples.

Usage

```
dtreevoting(num_runs = 100, num_levels = 10, file_path = file_path)
```

Arguments

num_runs	Number of subsamples to use for voting scheme (default: 100)
num_levels	Number of levels in each tree to consider. Only the features which appear in the top num_levels levels of the trees (from the root) will be counted
file_path	Where the num_runs subsample files are found (e.g. if sample 10 is at 'subsamples/sample10.csv' then file_path should be 'subsamples/sample'). There must be enough samples to fulfill num_runs runs.

Value

Outputs a dataframe containing (first column) total number of appearances of each feature (each row is a feature). The rest of the columns represent 1 run each and contain the level at which the feature appears.

Examples

```
dtreevoting(
  num_runs=5,
  num_levels=10,
 file_path=paste(system.file('samples/subsamples', package = "feamiR"),'/sample',sep=''))
```

eGA *Embryonic Genetic Algorithm. Feature selection based on Embryonic Genetic Algorithms. It performs feature selection by maintaining an ongoing set of 'good' set of features which are improved run by run. It outputs training and test accuracy, sensitivity and specificity and a list of <=k features.*

Description

Embryonic Genetic Algorithm. Feature selection based on Embryonic Genetic Algorithms. It performs feature selection by maintaining an ongoing set of 'good' set of features which are improved run by run. It outputs training and test accuracy, sensitivity and specificity and a list of <=k features.

Usage

```
eGA(
  k = 30,
  data_train,
  data_test,
  mutprob = 0.05,
  includePlot = FALSE,
  maxnumruns = 50
)
```

Arguments

k	Maximum number of features in the output feature set (default:30)
data_train	Training set: dataframe containing classification column and all other columns features. This is the dataset on which the decision tree model is trained.
data_test	Test set: dataframe containing classification column and all other columns features. This is the dataset on which the decision tree model in tested.
mutprob	Probability that mutation will be performed for each produced feature set from forward feature selection (default:0.05)
includePlot	Show performance scatter plot (default:FALSE)
maxnumruns	Maximum number of iterations after which the feature set will be output, if no other termination conditions have been met (default:50)

Value

List containing (ordered list of) selected features, performance percentages, accessed using training (training accuracy), test (test accuracy), trainsensitivity, testsensitivity, trainspecificity, testspecificity. Also accessed using listofongoing is a list containing the length of the ongoing set at each stage.

Examples

```
data_train = data.frame(
      classification=as.factor(c(1,1,0,0,1,1,0,0,1,1)),
      A=c(1,1,1,0,0,0,1,1,1,0),
      B=c(0,1,1,0,1,1,0,1,1,0),
      C=c(0,0,1,0,0,1,0,0,1,0),
      D=c(0,1,1,0,0,0,1,0,0,0),
      E=c(1,0,1,0,0,1,0,1,1,0))
data_test = data.frame(
      classification=as.factor(c(1,1,0,0,1,1,1,0)),
      A=c(0,0,0,1,0,0,0,1),
```

```
      B=c(1,1,1,0,0,1,1,1),
      C=c(0,0,1,1,0,0,1,1),
      D=c(0,0,1,1,0,1,0,1),
      E=c(0,0,1,0,1,0,1,1))
data = read.csv(paste(system.file('samples/subsamples', package = "feamiR"),'/sample0.csv',sep=''))
data = rbind(head(data,50),tail(data,50))
data$classification = as.factor(data$classification)
ind <- sample(2,nrow(data),replace=TRUE,prob=c(0.8,0.2))
data_train <- data[ind==1,]
data_test <- data[ind==2,]
eGA(k=7,data_train,data_test,maxnumruns=3)
```

`feamiR` *feamiR: Classification and feature selection for microRNA/mRNA interactions*

Description

The feamiR package provides two categories of functions: Dataset preparation functions and analysis functions

Dataset preparation functions

feamiR uses 1 preparation function called preparedataset. There are 2 options for the input mRNA dataset:

1. Reference genome (fasta) and corresponding annotation file (gtf). From these files, the three prime UTR sequences will be extracted for alignment. These paths for these files could be for example a toplevel file and annotation file from Ensembl (e.g. Homo_sapiens.GRCh38.dna.toplevel.fa and Homo_sapiens.GRCh38.100.chr.gtf). Before using this form of input check consistent naming of chromosomes between the two files and that the IDs are consistent with the interactions file you intend to use (e.g. using gene names). These files should be specificed using the fullchromosomes and annotations parameters. If this type of data is supplied you must also specify the number of chromosomes for the species in question (e.g. 23 for Homo sapiens) using the chr parameter.
2. Three prime UTR file (fasta). Again check consistent IDs with interactions file. This file should be specified using the mRNA_3pUTR parameter.

The input miRNA file should be a fasta file containing mature miRNA sequences (e.g. from miRBase). Check the miRNA IDs are consistent with the interaction dataset. From the mature sequences, the seed sequences will be extracted and saved to a separate fasta file.

The mRNA and miRNA datasets will be used for PaTMaN alignment then split into a positive dataset (validated interactions) and negative dataset (non-validated interactions with seed matches). For reformating and PaTMaN alignment both sreformat and patman must be installed and the paths to the executables specificied with sreformatpath and patmanpath. If the commands sreformat and patman work on your system then there is no need to specify the path.

To perform this split an interaction dataset must be supplied. This interaction dataset must contain a 'miRNA' column, 'Target Gene' column. It can also contain an Experiments column detailing

which type of experiment was used to validate the interaction. If this column is supplied some preprocessing should be performed so there are <=10 unique values. If the Experiments column is supplied, statistical analysis is performed on the dataset split by experiment type. Finally a 'Support Type' column may be included with values 'Functional MTI','Functional MTI (Weak)','Non-Functional MTI' and 'Non-Functional MTI (Weak)'. If this column is supplied and there are enough positive entries remaining then they will be filtered for only 'Functional MTI' entries (these entries are more likely to yield good results).

After alignment, first statistical analysis is performed. By default this is only on seed features but if specified using the nonseed_miRNA and flankingmRNA parameters then analysis can be performed on full miRNA features and flanking features. The chi-squared and Fisher exact p-values are saved in csvs and heatmaps created and saved as jpgs. If Experiments column is supplied in interactions dataset then statistical analysis is performed for the dataset split by experiment type.

Finally, the negative set is subsampled to be comparable to the positive set for the ML and feature selection component. Here 100 representative subsamples (checked by chi-squared tests) and created and labelled (1 if positive, 0 if negative) subsamples are saved in a subsamples folder.

By supplying the positive and negative sets using positiveset and negativeset parameters, the process skips straight to the statistical analysis stage but this should only be done with positive and negative sets created by feamiR (although they can be filtered if column names are unchanged)

A prefix for all output files can be supplied using the o parameter.

PLEASE NOTE: To use this function Python (>=3.6) must be installed on your system and the path specified. The following libraries must also be installed on the Python version you specify: os, Bio, gtfparse, pandas, numpy, math, scipy.stats, matplotlib.pyplot, seaborn as sns, statistics, logging.

ML and feature selection functions

Using subsamples created by the preparedataset function, feamiR contains several function for creating miRNA/mRNA classifiers and selecting features which contribute most strongly to the classifiers.

The classifier functions are: decisiontree, randomforest and svm. To select hyperparameters for randomforest and svm, you should use selectsvmkernel and selectrfnumtrees. This functions will produce plots through cross validation from which an appropriate number of trees and kernel can be identified. You should try this on multiple subsamples to check your selection.

Once these hyperparameters are identified, use runallmodels to create and analyse results from Decision Trees, Random Forests and SVMs on all 100 subsamples. The selected hyperparameters using selectsvmkernel and selectrfnumtrees should be input as parameters. The function will output a data.frame of the achieved test and training accuracy, sensitivity and specificity for each model on each subsample. Summary boxplots showing accuracy, sensitivity and specificity for each model will be produced. The function will also output dtreevote containing the features used in the decision trees for each subsample and the level of the tree at which they appear. Finally, the function outputs ongoingginis which contains the Gini index for each feature in the Random Forest for each subsample. The first column of dtreevote contains the number of runs for which each feature was used which can be used for feature selection. The first column of ongoingginis contains the cumulative Gini index for each feature across the 100 runs which can be used for feature selection.

As well as using the Decision Tree voting scheme and Random Forest cumulative Gini index measure, feamiR also has three further feature selection approaches. These are the traditional forwardfeatureselection and geneticalgorithm approaches as well as a novel approach based on embryonic

Genetic Algorithms using the eGA function. It is recommended that a combination of these feature selection appraoches across multiple subsamples and the statistical analysis is used to select discriminative features, for example using summary heatmaps.

`forwardfeatureselection`

Forward Feature Selection. Performs forward feature selection on the given list of features, placing them in order of discriminative power using a given model on the given dataset up to the accuracy plateau.

Description

Forward Feature Selection. Performs forward feature selection on the given list of features, placing them in order of discriminative power using a given model on the given dataset up to the accuracy plateau.

Usage

```
forwardfeatureselection(
  model = feamiR::svmlinear,
  training,
  test,
  featurelist,
  includePlot = FALSE
)
```

Arguments

`model`	The ML models used to classify the data, typically SVM with a given kernel
`training`	Training dataset as a data.frame with classification column and column for each feature.
`test`	Test dataset with matching columns to training.
`featurelist`	List of features to order
`includePlot`	Show number of features vs accuracy line plot (default:FALSE)

Value

Ordered list of most discriminative features when classifying the dataset along with training and test accuracy, sensitivity and specificity

Examples

```
data_train = data.frame(
      classification=as.factor(c(1,1,0,0,1,1,0,0,1,1)),
      A=c(1,1,1,0,0,0,1,1,1,0),
      B=c(0,1,1,0,1,1,0,1,1,0),
      C=c(0,0,1,0,0,1,0,0,1,0),
```

```
      D=c(0,1,1,0,0,0,1,0,0,0),
      E=c(1,0,1,0,0,1,0,1,1,0))
data_test = data.frame(
      classification=as.factor(c(1,1,0,0,1,1,1,0)),
      A=c(0,0,0,1,0,0,0,1),
      B=c(1,1,1,0,0,1,1,1),
      C=c(0,0,1,1,0,0,1,1),
      D=c(0,0,1,1,0,1,0,1),
      E=c(0,0,1,0,1,0,1,1))
listoffeatures = colnames(data_train)[colnames(data_train)!='classification']
forwardfeatureselection(feamiR::svmlinear,data_train,data_test,listoffeatures)
```

`geneticalgorithm` *Standard Genetic Algorithm. Implements a standard genetic algorithm using GA package (ga) with a fitness function specialised for feature selection.*

Description

Standard Genetic Algorithm. Implements a standard genetic algorithm using GA package (ga) with a fitness function specialised for feature selection.

Usage

```
geneticalgorithm(
  model = feamiR::svmlinear,
  k = 30,
  training,
  test,
  parallel = TRUE,
  mutprob = 0.1,
  crossprob = 0.8,
  popsize = 20,
  maxiter = 1000,
  maxiter_withoutimprovement = 300,
  numberpassedon = 3,
  plot = FALSE
)
```

Arguments

`model`	The ML models used to classify the data, typically SVM with a given kernel
`k`	Maximum number of features to be output.
`training`	Training dataset as a data.frame with classification column and column for each feature.
`test`	Test dataset with matching columns to training.
`parallel`	Specifies whether GA should be run sequentially or in parallel (default: TRUE)

mutprob	The probability that an individual undergoes mutation in a particular iteration (default: 0.1)
crossprob	The probability of crossover between pairs of individuals (default: 0.8)
popsize	The size of the solution population (default:20)
maxiter	The maximum number of iterations to run before termination (default: 1000)
maxiter_withoutimprovement	The maximum number of consecutive iterations without improvement to fitness before termination (default: 300)
numberpassedon	The number of best fitness individuals to be passed on to the next generation in each iteration (default: 3)
plot	Specifies whether GA plot should be shown (default: FALSE)

Value

Set (unordered) of <=k features and training and test accuracy, sensitivity and specificity using these features.

Examples

```
data_train = data.frame(
      classification=as.factor(c(1,1,0,0,1,1,0,0,1,1)),
      A=c(1,1,1,0,0,0,1,1,1,0),
      B=c(0,1,1,0,1,1,0,1,1,0),
      C=c(0,0,1,0,0,1,0,0,1,0),
      D=c(0,1,1,0,0,0,1,0,0,0),
      E=c(1,0,1,0,0,1,0,1,1,0))
data_test = data.frame(
      classification=as.factor(c(1,1,0,0,1,1,1,0)),
      A=c(0,0,0,1,0,0,0,1),
      B=c(1,1,1,0,0,1,1,1),
      C=c(0,0,1,1,0,0,1,1),
      D=c(0,0,1,1,0,1,0,1),
      E=c(0,0,1,0,1,0,1,1))
geneticalgorithm(
  feamiR::svmlinear,
  k=2,
  data_train,
  data_test,
  parallel=FALSE,
  maxiter=5,
  maxiter_withoutimprovement=5,
  popsize=10)
```

preparedataset *Dataset preparation This step performs all preparation necessary to perform feamiR analysis, taking a set of mRNAs, a set of miRNAs and an interaction dataset and creating corresponding positive and negative datasets for ML modelling.*

Description

PLEASE NOTE: This analysis is run in Python so python must be installed and location specified if not on PATH. Both sreformat and PaTMaN must also be installed and path specified if not on PATH. Python >= 3.6 is required to use the neccesary packages. The Python component required the following libraries: os, Bio, gtfparse, pandas, numpy, math, scipy.stats, matplotlib.pyplot, seaborn as sns, statistics, logging. Please ensure these are installed for the verison of Python you supply.

Usage

```
preparedataset(
  pythonversion = "python",
  mRNA_3pUTR = "",
  miRNA_full = "",
  interactions = "",
  annotations = "",
  fullchromosomes = "",
  seed = 1,
  nonseed_miRNA = 0,
  flankingmRNA = 0,
  UTR_output = "",
  chr = "",
  o = "feamiR_",
  positiveset = "",
  negativeset = "",
  sreformatpath = "sreformat",
  patmanpath = "patman",
  patmanoutput = "",
  minvalidationentries = 40,
  num_runs = 100,
  check_python = TRUE
)
```

Arguments

pythonversion	File path for installed Python version (default: python)
mRNA_3pUTR	Fasta file of only 3'UTRs, with gene name as name attribute (e.g. Serpinb8)
miRNA_full	Fasta file of full mature miRNA hairpins, with miRNA ID as name attribute (e.g. hsa-miR-576-3p)

`interactions` CSV file containing only validated interactions between miRNA and mRNA (e.g. from miRTarBase). Must have columns miRNA (e.g. hsa-miR-576-3p), Target Gene (e.g. Serpinb8) and optionally Experiments (e.g. qRT-PCR) and/or Support Type (with values Functional MTI, Functional MTI (Weak), Non-Functional MTI, Non-Functional MTI (Weak))

`annotations` GTF file (e.g. from Ensembl) with attributes seqname (chromosome), feature (with 3'UTRs labelled exactly 'three_prime_utr'), transcript_id, gene_id and gene_name matching fullchromosomes and interactions

`fullchromosomes` Fasta file (e.g. top level file from Ensembl) containing full sequence for each chromosome with name as chromosome (e.g. 1, matching seqname from annotations)

`seed` Binary, 1 if full miRNA seed features should be included in statistical analysis. Default: 1.

`nonseed_miRNA` Binary, 1 if full miRNA features should be included in statistical analysis. Seed features are always included. Default: 0.

`flankingmRNA` Binary, 1 if flanking region mRNA features should be included in statistical analysis. Seed features are always included. Default: 0.

`UTR_output` String. File name 3'UTR fasta file should be saved as (when annotations and full chromosomes files are supplied)

`chr` Number of chromosomes for species in question.

`o` Output prefix for any files created and saved.

`positiveset` CSV file containing validated pairs of miRNAs and mRNAs as output by initial stage of analysis. If positiveset and negative set are input, analysis begins at final statistical analysis stage.

`negativeset` CSV file containing non-validated pairs of miRNAs and mRNAs as output by initial stage of analysis. If positiveset and negative set are input, analysis begins at final statistical analysis stage.

`sreformatpath` File path for installed sreformat (default: sreformat)

`patmanpath` File path for installed patman (default: patman)

`patmanoutput` TXT file containing patman output (saved as output_prefix + patman_seed.txt). If supplied, analysis begins at patman output processing stage.

`minvalidationentries` Minimum number of entries for a validation category to be considered separately in statistical analysis (default: 40)

`num_runs` Number of subsamples to create (default: 100)

`check_python` Whether the Python version should be checked (default: TRUE)

Details

The function saves various files (using specified output_prefix) and if you wish to start preparation using one of these pre-output files then these can be specified and preparation will skip to that point (this should only be done with files output by the function).

Value

CSV containing full positive and negative sets. Folder statistical_analysis of heatmaps showing significance of various features under Fisher exact and Chi-squared tests. Seed analysis will always be run, full miRNA and flanking analysis if the respective parameters are set to 1. Folder subsamples containing CSVs for 100 subsamples with positive and negative samples equal for use in classifiers and feature selection.

Examples

```
preparedataset(
   pythonversion=Sys.which('python'),
   positiveset = system.file('samples','test_seed_positive.csv',package='feamiR'),
   negativeset=system.file('samples','test_seed_negative.csv',package='feamiR'),
   o='examples_',
   num_runs=0,
   check_python=FALSE)
```

---

randomforest

Random Forest. Trains a random forest on the training dataset and uses it to predict the classification of the test dataset. The resulting accuracy, sensitivity and specificity are returned, as well as a summary of the importance of features in the dataset.

Description

Random Forest. Trains a random forest on the training dataset and uses it to predict the classification of the test dataset. The resulting accuracy, sensitivity and specificity are returned, as well as a summary of the importance of features in the dataset.

Usage

```
randomforest(data_train, data_test, numoftrees = 10, includeplot = FALSE)
```

Arguments

data_train	Training set: dataframe containing classification column and all other columns features. This is the dataset on which the decision tree model is trained.
data_test	Test set: dataframe containing classification column and all other columns features. This is the dataset on which the decision tree model in tested.
numoftrees	Number of trees used in the random forest (default:10)
includeplot	Show performance scatter plot (default:FALSE)

Value

List containing performance percentages, accessed using training (training accuracy), test (test accuracy), trainsensitivity, testsensitivity, trainspecificity, testspecificity. Also accessed using importance is the vector of Mean Decrease in Gini Index. This can be used to find the features which contribute most to classification.

Examples

```
data_train = data.frame(
      classification=as.factor(c(1,1,0,0,1,1,0,0,1,1)),
      A=c(1,1,1,0,0,0,1,1,1,0),
      B=c(0,1,1,0,1,1,0,1,1,0),
      C=c(0,0,1,0,0,1,0,0,1,0))
data_test = data.frame(
      classification=as.factor(c(1,1,0,0,1,1,1,0)),
      A=c(0,0,0,1,0,0,0,1),
      B=c(1,1,1,0,0,1,1,1),
      C=c(0,0,1,1,0,0,1,1))
randomforest(data_train,data_test,numoftrees=5)
```

`rfgini` *Random Forest cumulative MeanDecreaseGini feature selection. Implements a feature selection approach based on cumulative MeanDecreaseGini using Random Forests trained on multiple subsamples.*

Description

Random Forest cumulative MeanDecreaseGini feature selection. Implements a feature selection approach based on cumulative MeanDecreaseGini using Random Forests trained on multiple subsamples.

Usage

```
rfgini(num_runs = 100, num_trees = 30, file_path = file_path)
```

Arguments

`num_runs`	Number of subsamples to use for voting scheme (default: 100)
`num_trees`	Number of trees for random forest (selected using select_rf_numtrees)
`file_path`	Where the num_runs subsample files are found (e.g. if sample 10 is at 'subsamples/sample10.csv' then file_path should be 'subsamples/sample'). There must be enough samples to fulfill num_runs runs.

Value

The function will output a data.frame with cumulative mean decrease in Gini for each feature in the first columns (each row is a feature) and the rest of the column containing mean decrease in Gini for each of the num_runs runs.

Examples

```
rfgini(
  num_runs=5,
  num_trees=30,
 file_path=paste(system.file('samples/subsamples',package = "feamiR"),'/sample',sep=''))
```

runallmodels	*Run all models. Trains and tests Decision Tree, Random Forest and SVM models on 100 subsamples and provides a summary of the results, to select the best model. The number of trees and kernel chosen by selectsvmkernel and selectrfnumtrees should be used for SVM and Random Forest respectively. We can use this function to inform feature selection, using a Decision Tree voting scheme and a Random Forest measure based on the Gini index.*

Description

Run all models. Trains and tests Decision Tree, Random Forest and SVM models on 100 subsamples and provides a summary of the results, to select the best model. The number of trees and kernel chosen by selectsvmkernel and selectrfnumtrees should be used for SVM and Random Forest respectively. We can use this function to inform feature selection, using a Decision Tree voting scheme and a Random Forest measure based on the Gini index.

Usage

```
runallmodels(
  num_trees = 20,
  kernel = "linear",
  degree = 3,
  poly = 0,
  file_path = file_path,
  num_runs = 100
)
```

Arguments

num_trees	Number of trees for random forest (selected using select_rf_numtrees)
kernel	Kernel for SVM (select using select_svm_kernel)
degree	Degree for SVM kernel (not necessary for linear or sigmoid functions)
poly	1 if polynomial kernel is used, 0 if linear, radial or sigmoid.
file_path	Where the <=num_runs subsample files are found (e.g. if sample 10 is at 'subsamples/sample10.csv' then file_path should be 'subsamples/sample')
num_runs	Number of subsamples to loop over (default: 100)

Value

The function will output a data.frame of the achieved test and training accuracy, sensitivity and specificity for each model on each subsample. Summary boxplots showing accuracy, sensitivity and specificity for each model will be produced. The function will also output dtreevote containing the features used in the decision trees for each subsample and the level of the tree at which they appear. Finally, the function outputs ongoingginis which contains the Gini index for each feature in the Random Forest for each subsample. The first column of dtreevote contains the number of

runs for which each feature was used which can be used for feature selection. The first column of ongoingginis contains the cumulative Gini index for each feature across the 100 runs which can be used for feature selection.

Examples

```
runallmodels(
  num_runs=5,
  num_trees=5,
  kernel='linear',
  poly=0,
 file_path=paste(system.file('samples/subsamples', package = "feamiR"),'/sample',sep=''))
```

`selectrfnumtrees` *Tuning number of trees hyperparameter. Trains random forests with a range of number of trees so the optimal number can be identified (using the resulting plot) with cross validation*

Description

Tuning number of trees hyperparameter. Trains random forests with a range of number of trees so the optimal number can be identified (using the resulting plot) with cross validation

Usage

```
selectrfnumtrees(
  data,
  maxnum = 100,
  title = "",
  showplots = TRUE,
  output_prefix = ""
)
```

Arguments

`data`	Dataset: dataframe containing classification column and all other column features. Both the training and test datasets will be taken from this dataset.
`maxnum`	Maximum number of trees to be considered. All numbers between 1 and maxnum will be considered. Default: 100.
`title`	Title to be used for the resulting boxplot
`showplots`	TRUE if plots should be shown in standard output, FALSE is plots should be saved as jpg files. Default: TRUE.
`output_prefix`	Prefix used for saving plots. If showplots==FALSE then plots are saved here. Otherwise, standard output.

Value

Dataframe containing test and training accuracy, sensitivity and specificity

Examples

```
data = read.csv(paste(system.file('samples/subsamples', package = "feamiR"),'/sample0.csv',sep=''))
data = rbind(head(data,50),tail(data,50))
data$classification = as.factor(data$classification)
data = data[,2:ncol(data)]
selectrfnumtrees(data,5,'RF boxplots')
```

`selectsvmkernel` *Tuning SVM kernel. Trains SVMs with a range of kernels (linear, polynomial degree 2, 3 and 4, radial and sigmoid) using cross validation so the optimal kernel can be chosen (using the resulting plots). If specified (by showplots=FALSE) the plots are saved as jpegs.*

Description

Tuning SVM kernel. Trains SVMs with a range of kernels (linear, polynomial degree 2, 3 and 4, radial and sigmoid) using cross validation so the optimal kernel can be chosen (using the resulting plots). If specified (by showplots=FALSE) the plots are saved as jpegs.

Usage

```
selectsvmkernel(data, title, showplots = TRUE, output_prefix = "")
```

Arguments

`data`	Dataset: dataframe containing classification column and all other column features. Both the training and test datasets will be taken from this dataset.
`title`	Title to be used for the resulting boxplot
`showplots`	TRUE if plots should be shown in standard output, FALSE is plots should be saved as jpg files.
`output_prefix`	Prefix used for saving plots. If showplots==FALSE then plots are saved here. Otherwise, standard output.

Value

Dataframe containing test and training accuracy, sensitivity and specificity

Examples

```
data = read.csv(paste(system.file('samples/subsamples', package = "feamiR"),'/sample0.csv',sep=''))
data = rbind(head(data,50),tail(data,50))
data$classification = as.factor(data$classification)
data = data[,2:ncol(data)]
selectsvmkernel(data,'SVM boxplots')
```

svm *SVM*

Description

SVM

Usage

```
svm(
  data_train,
  data_test,
  kernel = "linear",
  degree = 3,
  poly = 0,
  includeplot = FALSE
)
```

Arguments

data_train	Training set: dataframe containing classification column and all other columns features. This is the dataset on which the decision tree model is trained.
data_test	Test set: dataframe containing classification column and all other columns features. This is the dataset on which the decision tree model in tested.
kernel	Type of kernel to use for SVM model (default:linear)
degree	Degree for kernel used (in polynomial or radial case)
poly	Binary parameter stating whether the chosen kernel is polynomial of degree greater than 1 (default:0)
includeplot	Show performance scatter plot (default:FALSE)

Value

List containing performance percentages, accessed using training (training accuracy), test (test accuracy), trainsensitivity, testsensitivity, trainspecificity, testspecificity.

Examples

```
data_train = data.frame(
      classification=as.factor(c(1,1,0,0,1,1,0,0,1,1)),
      A=c(1,1,1,0,0,0,1,1,1,0),
      B=c(0,1,1,0,1,1,0,1,1,0),
      C=c(0,0,1,0,0,1,0,0,1,0))
data_test = data.frame(
      classification=as.factor(c(1,1,0,0,1,1,1,0)),
      A=c(0,0,0,1,0,0,0,1),
      B=c(1,1,1,0,0,1,1,1),
      C=c(0,0,1,1,0,0,1,1))
```

```
svm(data_train,data_test,kernel='radial',degree=3)
svm(data_train,data_test,kernel='sigmoid')
svm(data_train,data_test,kernel='poly',degree=4,poly=1)
```

---

svmlinear *Linear SVM Implements a linear SVM using the general svm function (for ease of use in feature selection)*

Description

Linear SVM Implements a linear SVM using the general svm function (for ease of use in feature selection)

Usage

```
svmlinear(data_train, data_test, includeplot = FALSE)
```

Arguments

data_train	Training set: dataframe containing classification column and all other columns features. This is the dataset on which the decision tree model is trained.
data_test	Test set: dataframe containing classification column and all other columns features. This is the dataset on which the decision tree model in tested.
includeplot	Show performance scatter plot (default:FALSE)

Value

List containing performance percentages, accessed using training (training accuracy), test (test accuracy), trainsensitivity, testsensitivity, trainspecificity, testspecificity.

Examples

```
data_train = data.frame(
      classification=as.factor(c(1,1,0,0,1,1,0,0,1,1)),
      A=c(1,1,1,0,0,0,1,1,1,0),
      B=c(0,1,1,0,1,1,0,1,1,0),
      C=c(0,0,1,0,0,1,0,0,1,0))
data_test = data.frame(
      classification=as.factor(c(1,1,0,0,1,1,1,0)),
      A=c(0,0,0,1,0,0,0,1),
      B=c(1,1,1,0,0,1,1,1),
      C=c(0,0,1,1,0,0,1,1))
svmlinear(data_train,data_test)
```

svmpolynomial2 *Polynomial degree 2 SVM Implements a polynomial degree 2 SVM using the general svm function (for ease of use in feature selection)*

Description

Polynomial degree 2 SVM Implements a polynomial degree 2 SVM using the general svm function (for ease of use in feature selection)

Usage

```
svmpolynomial2(data_train, data_test, includeplot = FALSE)
```

Arguments

data_train	Training set: dataframe containing classification column and all other columns features. This is the dataset on which the decision tree model is trained.
data_test	Test set: dataframe containing classification column and all other columns features. This is the dataset on which the decision tree model in tested.
includeplot	Show performance scatter plot (default:FALSE)

Value

List containing performance percentages, accessed using training (training accuracy), test (test accuracy), trainsensitivity, testsensitivity, trainspecificity, testspecificity.

Examples

```
data_train = data.frame(
      classification=as.factor(c(1,1,0,0,1,1,0,0,1,1)),
      A=c(1,1,1,0,0,0,1,1,1,0),
      B=c(0,1,1,0,1,1,0,1,1,0),
      C=c(0,0,1,0,0,1,0,0,1,0))
data_test = data.frame(
      classification=as.factor(c(1,1,0,0,1,1,1,0)),
      A=c(0,0,0,1,0,0,0,1),
      B=c(1,1,1,0,0,1,1,1),
      C=c(0,0,1,1,0,0,1,1))
svmpolynomial2(data_train,data_test)
```

svmpolynomial3 *Polynomial degree 3 SVM Implements a polynomial degree 3 SVM using the general svm function (for ease of use in feature selection)*

Description

Polynomial degree 3 SVM Implements a polynomial degree 3 SVM using the general svm function (for ease of use in feature selection)

Usage

```
svmpolynomial3(data_train, data_test, includeplot = FALSE)
```

Arguments

data_train	Training set: dataframe containing classification column and all other columns features. This is the dataset on which the decision tree model is trained.
data_test	Test set: dataframe containing classification column and all other columns features. This is the dataset on which the decision tree model in tested.
includeplot	Show performance scatter plot (default:FALSE)

Value

List containing performance percentages, accessed using training (training accuracy), test (test accuracy), trainsensitivity, testsensitivity, trainspecificity, testspecificity.

Examples

```
data_train = data.frame(
      classification=as.factor(c(1,1,0,0,1,1,0,0,1,1)),
      A=c(1,1,1,0,0,0,1,1,1,0),
      B=c(0,1,1,0,1,1,0,1,1,0),
      C=c(0,0,1,0,0,1,0,0,1,0))
data_test = data.frame(
      classification=as.factor(c(1,1,0,0,1,1,1,0)),
      A=c(0,0,0,1,0,0,0,1),
      B=c(1,1,1,0,0,1,1,1),
      C=c(0,0,1,1,0,0,1,1))
svmpolynomial3(data_train,data_test)
```

svmpolynomial4 *Polynomial degree 4 SVM Implements a polynomial degree 4 SVM using the general svm function (for ease of use in feature selection)*

Description

Polynomial degree 4 SVM Implements a polynomial degree 4 SVM using the general svm function (for ease of use in feature selection)

Usage

```
svmpolynomial4(data_train, data_test, includeplot = FALSE)
```

Arguments

data_train	Training set: dataframe containing classification column and all other columns features. This is the dataset on which the decision tree model is trained.
data_test	Test set: dataframe containing classification column and all other columns features. This is the dataset on which the decision tree model in tested.
includeplot	Show performance scatter plot (default:FALSE)

Value

List containing performance percentages, accessed using training (training accuracy), test (test accuracy), trainsensitivity, testsensitivity, trainspecificity, testspecificity.

Examples

```
data_train = data.frame(
      classification=as.factor(c(1,1,0,0,1,1,0,0,1,1)),
      A=c(1,1,1,0,0,0,1,1,1,0),
      B=c(0,1,1,0,1,1,0,1,1,0),
      C=c(0,0,1,0,0,1,0,0,1,0))
data_test = data.frame(
      classification=as.factor(c(1,1,0,0,1,1,1,0)),
      A=c(0,0,0,1,0,0,0,1),
      B=c(1,1,1,0,0,1,1,1),
      C=c(0,0,1,1,0,0,1,1))
svmpolynomial4(data_train,data_test)
```

svmradial *Radial SVM Implements a radial SVM using the general svm function (for ease of use in feature selection)*

Description

Radial SVM Implements a radial SVM using the general svm function (for ease of use in feature selection)

Usage

```
svmradial(data_train, data_test, includeplot = FALSE)
```

Arguments

data_train	Training set: dataframe containing classification column and all other columns features. This is the dataset on which the decision tree model is trained.
data_test	Test set: dataframe containing classification column and all other columns features. This is the dataset on which the decision tree model in tested.
includeplot	Show performance scatter plot (default:FALSE)

Value

List containing performance percentages, accessed using training (training accuracy), test (test accuracy), trainsensitivity, testsensitivity, trainspecificity, testspecificity.

Examples

```
data_train = data.frame(
      classification=as.factor(c(1,1,0,0,1,1,0,0,1,1)),
      A=c(1,1,1,0,0,0,1,1,1,0),
      B=c(0,1,1,0,1,1,0,1,1,0),
      C=c(0,0,1,0,0,1,0,0,1,0))
data_test = data.frame(
      classification=as.factor(c(1,1,0,0,1,1,1,0)),
      A=c(0,0,0,1,0,0,0,1),
      B=c(1,1,1,0,0,1,1,1),
      C=c(0,0,1,1,0,0,1,1))
svmradial(data_train,data_test)
```

svmsigmoid *Sigmoid SVM Implements a sigmoid SVM using general svm function (for ease of use in feature selection)*

Description

Sigmoid SVM Implements a sigmoid SVM using general svm function (for ease of use in feature selection)

Usage

```
svmsigmoid(data_train, data_test, includeplot = FALSE)
```

Arguments

data_train	Training set: dataframe containing classification column and all other columns features. This is the dataset on which the decision tree model is trained.
data_test	Test set: dataframe containing classification column and all other columns features. This is the dataset on which the decision tree model in tested.
includeplot	Show performance scatter plot (default:FALSE)

Value

List containing performance percentages, accessed using training (training accuracy), test (test accuracy), trainsensitivity, testsensitivity, trainspecificity, testspecificity.

Examples

```
data_train = data.frame(
      classification=as.factor(c(1,1,0,0,1,1,0,0,1,1)),
      A=c(1,1,1,0,0,0,1,1,1,0),
      B=c(0,1,1,0,1,1,0,1,1,0),
      C=c(0,0,1,0,0,1,0,0,1,0))
data_test = data.frame(
      classification=as.factor(c(1,1,0,0,1,1,1,0)),
      A=c(0,0,0,1,0,0,0,1),
      B=c(1,1,1,0,0,1,1,1),
      C=c(0,0,1,1,0,0,1,1))
svmsigmoid(data_train,data_test)
```

Index