Package 'Aphrodite'

July 31, 2015

Type Package

Title Automated PHenotype Routine for Observational Definition Identification Training and Evaluation (APHRODITE) - Phenotype building tool using Fuzzy labels

Version 1.0

Date 2015-06-09

Author Juan M. Banda [aut, cre],
Marc A. Suchard [aut],
Martijn J. Schuemie [aut]

Maintainer Juan M. Banda <jmbanda@stanford.edu>

Description Aphrodite uses noisy class labels to create silver standard training corpora to construct phenotype models in conjunction with expert knowledge codified in existing ontologies and a comprehensive representation of the patient clinical record to learn phenotype models.

License Apache License 2.0 | file LICENSE

Depends R (>= 3.1.0)

Imports DatabaseConnector,
SqlRender,
plyr,
caret,
pROC,
devtools,
knitr,
testthat

Suggests ggplot2

VignetteBuilder knitr

R topics documented:

`buildFeatureVector` *This function builds a feature vector using raw patient data*

Description

This function builds a feature vector using raw patient data. Returns a patient feature vector (divided by feature sets).

Usage

```
buildFeatureVector(flags, casesS, controlsS)
```

Arguments

`flags`	The R dataframe that contains all feature/model flags specified in settings.R.
`casesS`	Dataframe containing the raw patient data.
`controlsS`	(OPTIONAL) Dataframe containing the raw patient data.

Details

This function flattens the patient feature data (per feature set) into a feature vector that will be used as input for caret. This function can optionally flat two sources of patient data (cases and controls)

Value

An object containing the flattened feature vectors for all given feature sets. Of form: list(observations = FV_ob, visits = FV_v, labs = FV_lab, drugexposures = FV_de)

Examples

```
## Not run:

 fv_all<-buildFeatureVector(flag, dataFcases,dataFcontrols)

 #OR

 fv_cases<-buildFeatureVector(flag, dataFcases)


## End(Not run)
```

`buildKeywordList` *This function generates keyword and ignore lists based on the expansion of concepts.*

Description

Given any given concept_id or string of text this function generates keyword and ignore lists based on the expansion of concepts (looking at their synonyms).

Usage

```
buildKeywordList(connection, aphroditeConceptName, schema, dbms)
```

Arguments

`connection`	The connection to the database server.
`aphroditeConceptName`	The string of text / concept name to use.
`schema`	The database schema being used.
`dbms`	The target DBMS for SQL to be rendered in.

Details

Takes the aphroditeConceptName looks for synonyms and builds a list of related concepts using the vocabulary hierarchies

Value

A list with two elements: a list of positive keywords found (keywordlist_ALL), and a list of ignore keywords (ignorelist_ALL)

Examples

```
## Not run:

wordLists <- buildKeywordList(conn, aphrodite_concept_name, cdmSchema, dbms)


## End(Not run)
```

`buildModel` *This function builds a model for the specified feature vector using cases and controls for a certain outcomeName*

Description

This function builds a model for the specified feature vector using cases and controls for a certain outcomeName. Returns a caret trained model.

Usage

```
buildModel(flags, pp_total, outcomeNameS, saveFolder)
```

Arguments

`flags`	The R dataframe that contains all feature/model flags specified in settings.R.
`outcomeNameS`	String description of the outcome for which the model is being built
`featureVector`	Flattened feature vector returned by combineFeatureVectors function, with labeled cases and controls. Assumed to have one column named "Class_labels" and one named "pid"

Details

This function builds a model for the specified outcomeName. The model is specified in the flags dataframe (currently only supports LASSO).

Value

An transferable caret Model object

Examples

```
## Not run:

 model_predictors <- buildModel(flag, fv_all, predictorsNames, outcomeName, saveFolder)


## End(Not run)
```

`combineFeatureVectors` *This function combines all of the desired feature types into one single feature vector*

Description

This function combines all of the desired feature types into one single feature vector. This feature vector is ready to be used for training

Usage

```
combineFeatureVectors(flags, cases_pids, controls_pids, featureVector,
  outcomeNameS)
```

Arguments

`flags`	The R dataframe that contains all feature/model flags specified in settings.R.
`cases_pids`	List of patient_id's considered cases (for labeling purposes)
`controls_pids`	List of patient_id's considered controls (for labeling purposes)
`featureVector`	List of flattened feature vectors returned by buildFeatureVector function.
`outcomeNameS`	String description of the outcome for which the model is being built [Not actually needed]

Details

This function builds a feature vector by concatenating all of the available datasets. If binary features are specified in the settings, this conversion is made. The cases_pids and control_pids are patient_id's used for the labeling of the testing and training sets.

Value

fv_all - The combined feature vector (n patients x n features). The columns are: pid column, predictorNames, outcomeName

Examples

```
## Not run:

 fv_full_data <- combineFeatureVectors(flag, cases, controls, fv_all, outcomeName)


## End(Not run)
```

`conceptDecoder` *This function returns the concept terms corresponding to an input set of concept IDs.*

Description

This function returns the concept terms corresponding to an input set of concept IDs.

Usage

```
conceptDecoder(connection, schema, dbms, model, numFeats)
```

Arguments

`connection`	The connection to the database server.
`schema`	The database schema being used
`dbms`	The target DBMS for SQL to be rendered in.
`model`	The model object; will be used to extract top-ranking features
`numFeats`	The number of features you'd like returned

Details

This function returns the concept terms corresponding to an input set of concept IDs. Use case: to investigate highly-ranked features from classification model

Value

A list of concept terms and concept ids, corresponding to the IDs of interest

Examples

```
## Not run:

 high_ranking_concepts <- conceptDecoder(connection, schema, dbms, model, 20)


## End(Not run)
```

`convertFeatVecPortion` *This function builds a feature vector for a specific subset of features*

Description

This function builds a feature matrix for a specific subset of features, e.g. labs/visits/observations/drug exposures. Returns a feature matrix with all features from all patients included.

Usage

```
convertFeatVecPortion(featuresType, key, labIndic = 0)
```

Arguments

`featuresType`	A set of patient data in the form of a list of data frames. Each data frame contains a pid to label the patient, the names of the features that the patient had present, and the frequency counts of these features in his/her record
`key`	String descriptor of type of feature (e.g. "obs:" or "visit:"). This will be used to label the feature
`labIndic=0`	Whether this is for a lab feature. If so, must be converted from factor to numeric. Default is 0=no conversion required; 1=conversion required.

Details

This function takes a list of patient data frames as input. Each patient's data frame contains the features that this patient has present in his/her record. This function flattens this information into the combined feature matrix, with all features (of a certain type - e.g. labs or visits) from all patients included. Clearly, many patients will not have data for many features; their feature counts for any feature that was not present in their record will be set as 0.

Value

An data frame of (pts) x (features of input type)

Examples

```
## Not run:

 FV_converted<-convertFeatVecPortion(featuresType, 'obs:')

 #OR

 FV_converted<-convertFeatVecPortion(featuresType, 'labs:', labIndic=1)


## End(Not run)
```

`executeSQL` *This function executes one single SQL statement*

Description

This function renders, translates and executes one single SQL statement that produces a result

Usage

```
executeSQL(connection, schema, query, targetDBMS)
```

Arguments

`connection`	The connection to the database server.
`schema`	The database schema being used.
`query`	The SQL statement to retrieve the data.
`targetDBMS`	The target DBMS for SQL to be rendered in.

Details

Renders, translates, and executes a single SQL statement that is expeting to produce and result.

Value

An object containing the data.

Examples

```
## Not run:

  library("SqlRender")
  library("DatabaseConnector")
  library("Aphordite")
  connectionDetails <- createConnectionDetails(dbms="mysql", server="localhost",
         user="root", password="blah" ,schema="cdm_v5")
  conn <- connect(connectionDetails)

  concept_of_interest <- executeSQL(connection, schema, paste("SELECT concept_id,
         concept_name FROM @cdmSchema.concept WHERE lower(concept_name) =
         lower('myocardial infarction') AND standard_concept = 'S' AND
```

```
          invalid_reason IS NULL AND domain_id = 'Condition';" ,sep = ""),dbms)

  dbDisconnect(conn)

## End(Not run)
```

`f_score_calc` *This function creates a summary metric for model training*

Description

This function creates a new summary metric for model training, specific for unbalanced classes. Inputs as specified in caret.

Usage

```
f_score_calc(data, lev = levels(data$obs), model = NULL)
```

Arguments

`data`	A dataframe of the held-out example cases, with columns for 'obs', 'pred', 'T', 'F'. 'T' and 'F' have the probabilities of each of these classes
`lev`	Outcome factor levels for model
`model`	Character string of model used

Details

This function returns the F-score for model training optimization. Beta is currently set at 2 - TODO: should make this edit-able in future version.

Value

f_score

Examples

```
## Not run:

 f_score <- f_score_calc(data, lev, model)


## End(Not run)
```

getdPatientCohort — *This function builds a patient cohort (and controls) based on a concept list*

Description

This function will build a patient cohort with its respective controls using an inclusion concept_id list as well as an exclussion concept_id list. The user specifies the number of both cases and controls for his cohort.

Usage

```
getdPatientCohort(connection, dbms, includeConceptlist, excludeConceptlist,
  schema, cohortSize, controlSize)
```

Arguments

`connection`	The connection to the database server.
`dbms`	The target DBMS for SQL to be rendered in.
`includeConceptlist`	The list of concept_id's used to build the cohort.
`excludeConceptlist`	The list of concept_id's used as exclusion criteria for the cohort.
`schema`	The database schema being used.
`cohortSize`	The number of desired patients to appear in the cohort.
`controlSize`	The number of desired patients to be in the control group.

Details

This function takes the lists of include and exclude concept_ids and finds all patients that satisfy this characteristics from the Observation and Condition_occurrence tables in CDM V5.

Value

A list of dataframes containing both cases and control patient_id's.

Examples

```
## Not run:

casesANDcontrolspatient_ids_df<- getdPatientCohort(conn, dbms,
         as.character(keywordList_FF$V3), as.character(ignoreList_FF$V3),
         cdmSchema,nCases,nControls)
if (nCases > nrow(casesANDcontrolspatient_ids_df[[1]])) {
     message("Not enough patients to get the number of cases specified")
     stop
} else {
   if (nCases > nrow(casesANDcontrolspatient_ids_df[[2]])) {
       message("Not enough patients to get the number of controls specified")
       stop
   }
}

## End(Not run)
```

`getNormalizationTerm` *This function returns the normalizing factor, based upon the input settings*

Description

This function returns the normalizing factor, based upon the input settings

Usage

```
getNormalizationTerm(dates, flags, defaultTime = 1)
```

Arguments

`dates`	The dates of all visits recorded in the record
`flags`	The R dataframe that contains all feature/model flags specified in settings.R. - specifies which sort of normalization to perform
`defaultTime`	Value by which to normalize patients who only have a single visit, so cannot say what the follow-up time is (=0 –> undefined). Set default time as 1; as if spreading single observation over an entire year or 1 month (depending on settings)

Details

Depending upon the input settings, will return the normalizing term for the feature values. Some normalization settings depend upon the specific feature type; these are addressed within the individual feature types. This is a helper function for getPatientData

Value

The value by which to divide term counts

Examples

```
## Not run:

 timeDiff <- getNormalizationTerm(dates, flags)


## End(Not run)
```

`getPatientData` *This function fetches all the patient data (generic)*

Description

This function fetches all the patient data (generic). Returns raw patient data.

Usage

```
getPatientData(connection, dbms, patient_ids, keywords, flags, schema)
```

Arguments

`connection`	The connection to the database server.
`dbms`	The target DBMS for SQL to be rendered in.
`patient_ids`	The list of case patient id's to extract data from - NOT a data.frame.
`keywords`	The list of concept_id's that are NOT wanted to be used as features
`flags`	The R dataframe that contains all feature/model flags specified in settings.R.
`schema`	The database schema being used.

Details

Based on the groups of feature sets determined in the flags variable, this function will fetch patient data. The function returns all patient information

Value

An object containing the raw feature sets for the patient data.

Examples

```
## Not run:

 dataFcontrols <- getPatientData(conn, dbms, controls, flag , cdmSchema)


## End(Not run)
```

`getPatientDataCases` *This function fetches all the patient data (non-generic) designed to work when building a model*

Description

This function fetches all the patient data (non-generic) designed to work when building a model. Returns raw patient data.

Usage

```
getPatientDataCases(connection, dbms, patient_ids, keywords, ignores, flags,
  schema)
```

Arguments

`connection`	The connection to the database server.
`dbms`	The target DBMS for SQL to be rendered in.
`patient_ids`	The list of case patient id's to extract data from.
`keywords`	The list of concept_id's used to build the cohort.
`ignores`	The list of concept_id's ignored when building the cohort.
`flags`	The R dataframe that contains all feature/model flags specified in settings.R.
`schema`	The database schema being used.

Details

Based on the groups of feature sets determined in the flags variable, this function will fetch patient data. The function determines the first mention of the keywords and selects that date to start the data extraction of the remaining patient information

Value

An object containing the raw feature sets for the patient data.

Examples

```
## Not run:

 dataFcases <-getPatientDataCases(conn, dbms, cases, as.character(keywordList_FF$V3),
          flag , cdmSchema)


## End(Not run)
```

`manipulateSqlPull` *This function performs the manipulation of the sql extract data; should be generic for any of the feature types*

Description

This function performs the manipulation of the sql extract data; should be generic for any of the feature types

Usage

```
manipulateSqlPull(tmp_fv, flags, timeDiff)
```

Arguments

`tmp_fv`	Pull from sql query. Should have a column for date and concept_id
`flags`	Flags set in settings - specifies which normalization is needed
`timeDiff`	Value to use for normalization

Details

This is just a helper function that reduces the repeats of code for the manipulation of the sql extract data, so that it is put in the desired format for compiling all patient features together. This function: gets the counts of codes on a given visit (so multiple codes/terms/drugs/etc are not all counted); normalizes based on the normalization setting; returns a data frame with counts of codes

Value

An object containing the re-formatted patient data: ptID x (num concept IDs) - filled with counts, deduplicated by visit

Examples

```
## Not run:

 test1 <- manipulateSqlPull(tmp_fv, flags, timeDiff)


## End(Not run)
```

`plotFeatWeightings` *This function plots the feature importance weightings*

Description

This function plots the feature importance weightings

Usage

```
plotFeatWeightings(plotSaveFile, weightingsDF)
```

Arguments

`plotSaveFile`	The name of the file to save
`weightingsDF`	Data frame of the weightings with their labels

Details

This function returned predicted classes for the input patient list. Use case: evaluate trained model on a set of gold-standard patients.

Value

(none)

Examples

```
## Not run:

 plotFeatWeightings(plotSaveFile, weightingsDF)


## End(Not run)
```

Index