# Package 'planningML'

November 8, 2022

**Title** A Sample Size Calculator for Machine Learning Applications in Healthcare

**Version** 1.0.0

**Description** Advances in automated document classification has led to identifying massive numbers of clinical concepts from handwritten clinical notes. These high dimensional clinical concepts can serve as highly informative predictors in building classification algorithms for identifying patients with different clinical conditions, commonly referred to as patient phenotyping. However, from a planning perspective, it is critical to ensure that enough data is available for the phenotyping algorithm to obtain a desired classification performance. This challenge in sample size planning is further exacerbated by the high dimension of the feature space and the inherent imbalance of the response class. Currently available sample size planning methods can be categorized into: (i) model-based approaches that predict the sample size required for achieving a desired accuracy using a linear machine learning classifier and (ii) learning curve-based approaches (Figueroa et al. (2012) <doi:10.1186/1472-6947-12-8>) that fit an inverse power law curve to pilot data to extrapolate performance. We develop model-based approaches for imbalanced data with correlated features, deriving sample size formulas for performance metrics that are sensitive to class imbalance such as Area Under the receiver operating characteristic Curve (AUC) and Matthews Correlation Coefficient (MCC). This is done using a two-step approach where we first perform feature selection using the innovated High Criticism thresholding method (Hall and Jin (2010) <doi:10.1214/09-AOS764>), then determine the sample size by optimizing the two performance metrics. Further, we develop software in the form of an R package named 'planningML' and an 'R' 'Shiny' app to facilitate the convenient implementation of the developed model-based approaches and learning curve approaches for imbalanced data. We apply our methods to the problem of phenotyping rare outcomes using the MIMIC-III electronic health record database. We show that our developed methods which relate training data size and performance on AUC and MCC, can predict the true or observed performance from linear ML classifiers such as LASSO and SVM at different training data sizes. Therefore, in high-dimensional classification analysis with imbalanced data and correlated features, our approach can efficiently and accurately determine the sample size needed for machine-learning based classification.

**Imports** tidyverse, caret, lubridate, Matrix, MESS, dplyr, pROC, stats

**Depends** R (>= 3.5.0)

**License** GPL-2

**Encoding** UTF-8

**RoxygenNote** 7.2.1

**Suggests** knitr,rmarkdown

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Xinying Fang [aut, cre],
Satabdi Saha [aut],
Jaejoon Song [aut],
Sai Dharmarajan [aut]

**Maintainer** Xinying Fang <fxy950225@gmail.com>

**Repository** CRAN

**Date/Publication** 2022-11-08 10:20:02 UTC

# R topics documented:



---

`featureselection` *Feature selection*

---

## Description

This function selects important features from the dataset

## Usage

```
featureselection(x = NULL, y = NULL, method = "iHCT")
```

## Arguments

| | |
|---|---|
| `x` | a matrix of predictor variables |
| `y` | a vector of binary outcome |
| `method` | feature selection method, default is iHCT |

## Value

`featureselection()` returns selected features and other outcomes needed for sample size determination.

## Examples

```
## load data
pilot.data = readRDS(system.file("extdata", "pilotdata_sub.rds", package = "planningML"))
x = pilot.data[,-ncol(pilot.data)]
y = pilot.data$DEPRESSION

## select important features
features = featureselection(x = x, y = y)
summary(features)
```

---

`fit_learningcurve` *Generate descriptive summary for objects returned by functions in EHRsampling*

---

## Description

Generate descriptive summary for objects returned by functions in EHRsampling.

## Usage

```
fit_learningcurve(df, testX, target = NULL)
```

## Arguments

| | |
|---|---|
| `df` | data for learning curve fitting; first column is sample size, second column is AUC measurement. |
| `testX` | test data for prediction |
| `target` | target MCC/AUC that you want to achieve |

## Value

`fit_learningcurve()` returns the estimated power law model for the learning curve.

---

`learningcurve_data` *Generate descriptive summary for objects returned by functions in EHRsampling*

---

## Description

Generate descriptive summary for objects returned by functions in EHRsampling.

## Usage

```
learningcurve_data(
  x,
  y,
  method = "log",
  metric = "MCC",
  batchsize = 60,
  class.prob,
  pct.train = 0.8,
  nfold = 5,
  nrepeat = 10
)
```

## Arguments

| | |
|---|---|
| x | a matrix of predictor variables |
| y | a vector of binary outcome, encoded as a factor and denoted by 1 for events and 0 for non-events |
| method | training method to get performance measurements. Available options are "log" (logistic regression, default), "regul.log" (regularized logistic regression), "svm" (support vector machine), "rf" (random forest) and "lda" (linear discriminant analysis) |
| metric | default = "MCC". The target performance estimation metric that you want to optimize. Other choice can be "AUC". |
| batchsize | sample size for each training batch |
| class.prob | probability of the event |
| pct.train | the percentage of data that goes to training. Default is 0.8 |
| nfold | number of folds in cross validation |
| nrepeat | number of repeats for cross validation |

## Value

`learningcurve_data()` returns a data frame of sample size and the corresponding performance measurements.

---

# plot.planningML *Plot sample size dependent AUC or MCC based on number of selected features*

---

## Description

Plot the output returned by samplesize function

### Usage

```
## S3 method for class 'planningML'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | the output returned by the samplesize function |
| ... | ignored arguments |

### Value

plot() returns a scatterplot of sample size dependent performance measurement metrics (AUC or MCC) based on number of selected features

---

## samplesize *Sample size determination*

---

### Description

This function determine the optimal sample size based on the performance evaluation metric and number of selected features.

### Usage

```
samplesize(
  features = NULL,
  method = "HCT",
  m = NULL,
  effectsize = NULL,
  class.prob = NULL,
  totalnum_features = NULL,
  threshold = 0.1,
  metric = "MCC",
  target = NULL
)
```

### Arguments

| | |
|---|---|
| features | feature selection results from the featureselection function in the package. |
| method | default is HCT method, sample size dependent performance metric based on HCT method (HCT) or DS method (DS). |
| m | the number of features involved in the sample size determination. Default is NULL, which means the number of features are determined by the featureselection results based on the iHCT method. Otherwise, users can select the number based on their needs. The self-defined m should be smaller than the optimal number of features determined by the featureselection function. |

| | |
|---|---|
| effectsize | common effect size the the m features. NULL means the effect size is directly calculated from the data. Users can also provide the effect sizes based on historical data. |
| class.prob | probability of the event |
| totalnum_features | total number of features |
| threshold | default = 0.1. Threshold needed to determine the sample size. |
| metric | default = "MCC". The target performance estimation metric that you want to optimize. Other choices can be AUC. |
| target | target MCC/AUC that you want to achieve |

## Value

samplesize() returns sample size needed to achieve corresponding performance measurements.

## Examples

```
# first compute the results of featureselection function:
## load data
pilot.data = readRDS(system.file("extdata", "pilotdata_sub.rds", package = "planningML"))
x = pilot.data[,-ncol(pilot.data)]
y = pilot.data$DEPRESSION

## select important features
features = featureselection(x = x, y = y)

## determine sample size
if (identical(features$index, integer(0))){
   output = samplesize(features=features,
                  method="HCT", m=c(5,10,length(features$features)),
                   effectsize=NULL, class.prob = NULL, totalnum_features = NULL,
                   threshold=0.1, metric="MCC", target = NULL)
   output
   summary(output)
  plot(output) # Plot sample size dependent AUC or MCC based on number of selected features
}
```

---

summary.planningML *Generate descriptive summary for objects returned by functions in EHRsampling*

---

## Description

Generate descriptive summary for objects returned by functions in EHRsampling.

## Usage

```
## S3 method for class 'planningML'
summary(object, ...)
```

## Arguments

| | |
|---|---|
| `object` | the object returned by other functions. |
| `...` | ignored arguments |

## Value

`summary()` prints the objects returned by other functions.

# Index