

# Package ‘neighbr’

February 23, 2017

**Title** Classification, Regression, Clustering with K Nearest Neighbors

**Version** 1.0

**Description** Classification, regression, and clustering with k nearest neighbors algorithm. Implements several distance and similarity measures, covering continuous and logical features. Outputs ranked neighbors. Most features of this package are directly based on the PMML specification for KNN.

**Depends** R (>= 3.3.0)

**License** GPL (>= 2.1)

**Encoding** UTF-8

**LazyData** true

**Suggests** testthat, knitr, rmarkdown, mlbench

**RoxygenNote** 6.0.0

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Dmitriy Bolotov [aut, cre],  
Zementis Inc [cph]

**Maintainer** Dmitriy Bolotov <dmitriy.bolotov@softwareag.com>

**Repository** CRAN

**Date/Publication** 2017-02-23 08:31:25

## R topics documented:

distance . . . . .	2
knn . . . . .	3
neighbr . . . . .	5
similarity . . . . .	6
<b>Index</b>	<b>8</b>

---

distance	<i>Calculate the distance between two vectors.</i>
----------	--

---

### Description

Calculate the distance between two vectors.

### Usage

```
distance(x, y, measure)
```

### Arguments

x, y	Numeric vectors.
measure	Distance measure ("euclidean" or "squared_euclidean").

### Details

Distance measures in this package are based on those defined in the [PMML specification](#). Distances are calculated using the following equations:

Euclidean:  $(\sum((x_i - y_i)^2))^{0.5}$

Squared euclidean:  $\sum((x_i - y_i)^2)$

The input vectors must be of the same length.

### Value

The distance between x and y.

### See Also

[similarity](#), [PMML comparison measures](#)

### Examples

```
distance(c(-0.5, 1), c(0.4, 1.6), "euclidean")
distance(c(-0.5, 1), c(0.4, 1.6), "squared_euclidean")
```

---

knn *Classification, regression, and clustering with k nearest neighbors.*

---

### Description

Classification, regression, and clustering with k nearest neighbors.

### Usage

```
knn(train_set, test_set, k = 3, categorical_target = NULL,  
    continuous_target = NULL, comparison_measure,  
    categorical_scoring_method = "majority_vote",  
    continuous_scoring_method = "average", return_ranked_neighbors = 0,  
    id = NULL)
```

### Arguments

train_set	Data frame containing the training instances, with features and any targets and IDs.
test_set	Data frame containing the test instances, with feature columns only.
k	Number of nearest neighbors.
categorical_target	Categorical target variable.
continuous_target	Continuous target variable.
comparison_measure	Distance or similarity measure.
categorical_scoring_method	Categorical scoring method.
continuous_scoring_method	Continuous scoring method.
return_ranked_neighbors	Number of ranked neighbors to return. A 0 indicates no ranked neighbors. Must not exceed k.
id	Column containing unique identifiers for each row in the training set. Only used when return_ranked_neighbors > 0.

### Details

The algorithm can score data with continuous or logical features.

The algorithm can predict either a continuous or categorical target, or both (but no more than one of each), as well as return the closest neighbors ranked by distance or similarity. If no continuous or categorical target is provided, return\_ranked\_neighbors must be non-zero, and ranked neighbors will be returned.

There is no predict method for knn. The scored test set is returned as part of the neighbor object. The data to be scored must be passed in with the training data to knn().

Supported distance measures (used with continuous features): euclidean, squared\_euclidean.

Supported similarity measures (used with logical features): simple\_matching, jaccard, tanimoto.

Currently, only one type of categorical\_scoring\_method and continuous\_scoring\_method are supported (majority vote and average, respectively).

Logical features must consist of 0,1 or TRUE,FALSE values.

Categorical non-logical features must be transformed before being used.

The categorical target does not have to be of factor class, but is assumed to be not continuous.

The distance and similarity measures in this package are based on those defined in the [PMML specification](#).

Several of the elements in the returned list are only used when converting the knn model to PMML (for example, function\_name).

For more details and examples, see the vignette by running the following:

```
vignette("neighbor-help")
```

## Value

An object of class neighbor, which is a list of the following:

call	The original call to knn.
k	Number of nearest neighbors.
categorical_target	Categorical target variable.
continuous_target	Continuous target variable.
comparison_measure	Distance or similarity measure.
categorical_scoring_method	Categorical scoring method.
continuous_scoring_method	Continuous scoring method.
return_ranked_neighbors	Number of ranked neighbors to return.
id	ID variable.
features	List of feature names.
function_name	Function name, used when generating PMML. One of "classification", "regression", "clustering", or "mixed".
categorical_levels	Levels of the categorical target.
num_train_rows	Number of training instances.
num_test_rows	Number of test instances.
train_set	Data frame with training instances.
test_set_scores	Data frame with scores for the test set.

**See Also**

[similarity](#), [distance](#), [PMML KNN specification](#)

**Examples**

```
# continuous features with continuous target, categorical target,  
# and neighbor ranking  
  
data(iris)  
  
# add an ID column to the data for neighbor ranking  
iris$ID <- c(1:150)  
  
# train set contains all predicted variables, features, and ID column  
train_set <- iris[1:145,]  
  
# omit predicted variables or ID column from test set  
test_set <- iris[146:150,-c(4,5,6)]  
  
fit <- knn(train_set=train_set,test_set=test_set,  
           k=5,  
           categorical_target="Species",  
           continuous_target= "Petal.Width",  
           comparison_measure="euclidean",  
           return_ranked_neighbors=3,  
           id="ID")
```

---

neighbr

*neighbr: A package for computing nearest neighbors.*

---

**Description**

Classification, regression, and clustering with k nearest neighbors.

**Neighbr functions**

The package contains a function for running the nearest neighbors algorithm, as well as functions to directly compute distances between two vectors.

**More information**

[knn](#) documents the main knn function.

[distance](#) and [similarity](#) provide details on standalone measures.

For more details and examples, see the vignette by running the following:

```
vignette("neighbr-help")
```

---

 similarity

---

*Calculate the similarity between two vectors of logicals.*


---

### Description

Calculate the similarity between two vectors of logicals.

### Usage

```
similarity(x, y, measure)
```

### Arguments

x, y	Logical or numeric vectors.
measure	Similarity measure ("simple_matching", "jaccard", or "tanimoto")

### Details

Input vectors must consist of logical or numeric elements TRUE,FALSE or 0,1 (not factors). Similarity measures in this package are based on those defined in the [PMML specification](#). Similarity ranges from 0 (no similarity) to 1 (identical).

For logical vectors x and y, we define the following:

a11 = number of times where x<sub>i</sub>=1 and y<sub>i</sub>=1

a10 = number of times where x<sub>i</sub>=1 and y<sub>i</sub>=0

a01 = number of times where x<sub>i</sub>=0 and y<sub>i</sub>=1

a00 = number of times where x<sub>i</sub>=0 and y<sub>i</sub>=0

Similarities are calculated using the following formulas:

Simple matching:  $(a_{11} + a_{00}) / (a_{11} + a_{10} + a_{01} + a_{00})$

Jaccard:  $(a_{11}) / (a_{11} + a_{10} + a_{01})$

Tanimoto:  $(a_{11} + a_{00}) / (a_{11} + 2 * (a_{10} + a_{01}) + a_{00})$

### Value

The similarity between x and y.

### See Also

[distance](#), [PMML comparison measures](#)

**Examples**

```
similarity(c(0,1,1),c(0,0,1),"simple_matching")  
similarity(c(0,1,1),c(0,0,1),"jaccard")  
similarity(as.logical(c(0,1,1)),as.logical(c(0,0,1)),"tanimoto")
```

# Index

distance, [2](#), [5](#), [6](#)

knn, [3](#), [5](#)

neighbr, [5](#)

neighbr-package (neighbr), [5](#)

similarity, [2](#), [5](#), [6](#)