Package 'ConfusionTableR'

May 7, 2026

Type Package

Title Confusion Matrix Toolset

Version 1.0.4

Maintainer Gary Hutson <hutsons-hacks@outlook.com>

Description Takes the outputs of a 'caret' confusion matrix and allows for the quick conversion of these list items to lists.
The intended usage is to allow the tool to work with the outputs of machine learning classification models.
This tool works with classification problems for binary and multi-classification problems and allows for the record level conversion of the confusion matrix outputs.
This is useful, as it allows quick conversion of these objects for storage in database systems and to track ML model performance over time.
Traditionally, this approach has been used for highlighting model representation and feature slippage.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.1.2

Imports dplyr, tidyr, magrittr, caret, purrr, furrr

Suggests knitr, rmarkdown, e1071, randomForest, scales, mlbench, FeatureTerminatoR

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Collate 'MultiFramer.R' 'SingleFramer.R' 'binaryVisualiseR.R' 'dummycoder.R' 'globals.R'

Language en-US

Author Gary Hutson [aut, cre] (ORCID: <https://orcid.org/0000-0003-3534-6143>)

Date/Publication 2021-12-01 16:30:01 UTC

Contents

`binary_class_cm` *Binary Confusion Matrix data frame*

Description

a confusion matrix object for binary classification machine learning problems.

Usage

```
binary_class_cm(train_labels, truth_labels, ...)
```

Arguments

`train_labels`	the classification labels from the training set
`truth_labels`	the testing set ground truth labels for comparison
`...`	function forwarding for additional ‘caret‘ confusion matrix parameters to be passed such as mode="everything" and positive="class label"

Value

A list containing the outputs highlighted hereunder:

- **"confusion_matrix"** a confusion matrix list item with all the associated confusion matrix statistics
- **"record_level_cm"** a row by row data.frame version of the above output, to allow for storage in databases and row by row for tracking ML model performance
- **"cm_tbl"** a confusion matrix raw table of the values in the matrix
- **"last_run"**datetime object storing when the function was run

Examples

```
library(dplyr)
library(ConfusionTableR)
library(caret)
library(tidyr)
library(mlbench)

# Load in the data
data("BreastCancer", package = "mlbench")
```

```
breast <- BreastCancer[complete.cases(BreastCancer), ] #Create a copy
breast <- breast[, -1]
breast <- breast[1:100,]
breast$Class <- factor(breast$Class) # Create as factor
for(i in 1:9) {
 breast[, i] <- as.numeric(as.character(breast[, i]))
}

#Perform train / test split on the data
train_split_idx <- caret::createDataPartition(breast$Class, p = 0.75, list = FALSE)
train <- breast[train_split_idx, ]
test <- breast[-train_split_idx, ]
rf_fit <- caret::train(Class ~ ., data=train, method="rf")
#Make predictions to expose class labels
preds <- predict(rf_fit, newdata=test, type="raw")
predicted <- cbind(data.frame(class_preds=preds), test)

#ConfusionTableR to produce record level output
cm <- ConfusionTableR::binary_class_cm(predicted$class_preds,predicted$Class)
# Other modes here are mode="prec_recall", mode="sens_spec" and mode="everything"
# Record level output
cm$record_level_cm #Primed for storage in a database table
# List confusion matrix
cm$confusion_matrix
```

`binary_visualiseR` *Binary Visualiser - A Binary Confusion Matrix Visual*

Description

a confusion matrix object for binary classification machine learning problems. Returns a plot to visualise the important statistics derived from a confusion matrix, see: https://machinelearningmastery.com/confusion-matrix-machine-learning/.

Usage

```
binary_visualiseR(
  train_labels,
  truth_labels,
  class_label1 = "Class Negative",
  class_label2 = "Class Positive",
  quadrant_col1 = "#3F97D0",
  quadrant_col2 = "#F7AD50",
  custom_title = "Confusion matrix",
  info_box_title = "Confusion matrix statistics",
  text_col = "black",
  round_dig = 2,
  cm_stat_size = 1.4,
  cm_stat_lbl_size = 1.5,
```

```
  ...
)
```

Arguments

train_labels	the classification labels from the training set
truth_labels	the testing set ground truth labels for comparison
class_label1	classification label 1 i.e. readmission into hospital
class_label2	classification label 2 i.e. not a readmission into hospital
quadrant_col1	colour of the first quadrant - specified as hexadecimal
quadrant_col2	colour of the second quadrant - specified as hexadecimal
custom_title	title of the confusion matrix plot
info_box_title	title of the confusion matrix statistics box
text_col	the colour of the text
round_dig	rounding options
cm_stat_size	the cex size of the statistics box label
cm_stat_lbl_size	the cex size of the label in the statistics box
...	function forwarding to the confusion matrix object to pass additional args, such as positive = "Class label"

Value

returns a visual of a Confusion Matrix output

Examples

```
library(dplyr)
library(ConfusionTableR)
library(caret)
library(tidyr)
library(mlbench)


# Load in the data
data("BreastCancer", package = "mlbench")
breast <- BreastCancer[complete.cases(BreastCancer), ] #Create a copy
breast <- breast[, -1]
breast <- breast[1:100,]
breast$Class <- factor(breast$Class) # Create as factor
for(i in 1:9) {
 breast[, i] <- as.numeric(as.character(breast[, i]))
}

#Perform train / test split on the data
train_split_idx <- caret::createDataPartition(breast$Class, p = 0.75, list = FALSE)
train <- breast[train_split_idx, ]
```

```
test <- breast[-train_split_idx, ]
rf_fit <- caret::train(Class ~ ., data=train, method="rf")
#Make predictions to expose class labels
preds <- predict(rf_fit, newdata=test, type="raw")
predicted <- cbind(data.frame(class_preds=preds), test)
# Create the visual
ConfusionTableR::binary_visualiseR(predicted$class_preds, predicted$Class)
```

---

dummy_encoder *Dummy Encoder function to encode multiple columns at once*

Description

This function has been designed to encode multiple columns at once and allows the user to specify whether to drop the reference columns or retain them in the data

Usage

```
dummy_encoder(df, columns, map_fn = furrr::future_map, remove_original = TRUE)
```

Arguments

df	- data.frame object to pass to the function
columns	- vector of columns to be encoded for dummy encoding
map_fn	- choice of mapping function purrr:map or furr::future_map accepted
remove_original	- remove the variables that the dummy encodings are based off

Value

A tibble containing the dummy encodings

Examples

```
## Not run:
#Use the NHSR stranded dataset
df <- NHSRdatasets::stranded_data
#Create a function to select categorical variables
sep_categorical <- function(df){
 cats <- df %>%
   dplyr::select_if(is.character)
 return(cats)
}
cats <- sep_categorical(df) %>%
 dplyr::select(-c(admit_date))
#Dummy encoding
columns_vector <- c(names(cats))
dummy_encodings <- dummy_encoder(cats, columns_vector)
```

```
glimpse(dummy_encodings)

## End(Not run)
```

`multi_class_cm` *Multiple Confusion Matrix data frame*

Description

a confusion matrix object for multiple outcome classification machine learning problems.

Usage

```
multi_class_cm(train_labels, truth_labels, ...)
```

Arguments

`train_labels`	the classification labels from the training set
`truth_labels`	the testing set ground truth labels for comparison
`...`	function forwarding for passing mode and other parameters to 'caret' confusionMatrix

Value

A list containing the outputs highlighted hereunder:

- **"confusion_matrix"** a confusion matrix list item with all the associated confusion matrix statistics
- **"record_level_cm"** a row by row data.frame version of the above output, to allow for storage in databases and row by row for tracking ML model performance
- **"cm_tbl"** a confusion matrix raw table of the values in the matrix
- **"last_run"**datetime object storing when the function was run

Examples

```
# Get the IRIS data as this is a famous multi-classification problem
library(caret)
library(ConfusionTableR)
library(randomForest)
df <- iris
df <- na.omit(df)
table(iris$Species)
# Create a training / test split
train_split_idx <- caret::createDataPartition(df$Species, p = 0.75, list = FALSE)
# Here we define a split index and we are now going to use a multiclass ML model to fit the data
train <- df[train_split_idx, ]
test <- df[-train_split_idx, ]
# Fit a random forest model on the data
```

```
rf_model <- caret::train(Species ~ .,data = df,method = "rf", metric = "Accuracy")
# Predict the values on the test hold out set
rf_class <- predict(rf_model, newdata = test, type = "raw")
predictions <- cbind(data.frame(train_preds=rf_class, test$Species))
# Use ConfusionTableR to create a row level output
cm <- ConfusionTableR::multi_class_cm(predictions$train_preds, predictions$test.Species)
# Create the row level output
cm_rl <- cm$record_level_cm
print(cm_rl)
#Expose the original confusion matrix list
cm_orig <- cm$confusion_matrix
print(cm_orig)
```

Index