# Package 'basemodels'

May 7, 2026

**Type** Package

**Title** Baseline Models for Classification and Regression

**Version** 1.1.0

**Description** Providing equivalent functions for the dummy classifier and regressor used in 'Python' 'scikit-learn' library. Our goal is to allow R users to easily identify baseline performance for their classification and regression problems. Our baseline models use no predictors, and are useful in cases of class imbalance, multiclass classification, and when users want to quickly identify how much improvement their statistical and machine learning models are over several baseline models. We use a ``better'' default (proportional guessing) for the dummy classifier than the 'Python' implementation (``prior'', which is the most frequent class in the training set). The functions in the package can be used on their own, or introduce methods named 'dummy_regressor' or 'dummy_classifier' that can be used within the caret package pipeline.

**License** MIT + file LICENSE

**URL** https://github.com/Ying-Ju/basemodels

**Imports** stats

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**Suggests** caret, knitr, rmarkdown

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Ying-Ju Chen [aut, cre] (ORCID: <https://orcid.org/0000-0002-6444-6859>), Fadel M. Megahed [aut] (ORCID: <https://orcid.org/0000-0003-2194-5110>), L. Allison Jones-Farmer [aut] (ORCID: <https://orcid.org/0000-0002-1529-1133>), Steven E. Rigdon [aut] (ORCID: <https://orcid.org/0000-0001-7668-0899>)

**Maintainer** Ying-Ju Chen <ychen4@udayton.edu>

**Repository** CRAN

**Date/Publication** 2023-08-09 04:10:03 UTC

# Contents



---

`dummyClassifier` *a method used for the train function in caret*

---

## Description

a method used for the train function in caret

## Usage

```
dummyClassifier
```

## Format

An object of class `list` of length 13.

## Examples

```
# Split the data into training and testing sets
set.seed(2023)
index <- sample(1:nrow(iris), nrow(iris) * 0.8)
train_data <- iris[index,]
test_data <- iris[-index,]

ctrl1 <- caret::trainControl(method = "none")
# Train a dummy classifier with caret
dummy_model <- caret::train(Species ~ ., data = train_data,
                            method = dummyClassifier,
                            strategy = "stratified",
                            trControl = ctrl1)

# Make predictions using the trained dummy classifier
pred_vec <- predict(dummy_model, test_data)

# Evaluate the performance of the dummy classifier
conf_matrix <- caret::confusionMatrix(pred_vec, test_data$Species)
print(conf_matrix)
```

## dummyRegressor *a method used for the train function in caret*

### Description

a method used for the train function in caret

### Usage

```
dummyRegressor
```

### Format

An object of class `list` of length 13.

### Examples

```
# Split the data into training and testing sets
set.seed(2023)
index <- sample(1:nrow(iris), nrow(iris) * 0.8)
train_data <- iris[index,]
test_data <- iris[-index,]

ctrl1 <- caret::trainControl(method = "none")
# Train a dummy regressor with caret
reg_model <- caret::train(Sepal.Length ~ ., data = train_data,
                                method = dummyRegressor,
                                strategy = "median",
                                trControl = ctrl1)
y_hat <- predict(reg_model, test_data)
# Find mean squared error
mean((test_data$Sepal.Length-y_hat)^2)
```

## dummy_classifier *dummy classifier for a categorical variable.*

### Description

dummy classifier for a categorical variable.

### Usage

```
dummy_classifier(
  y,
  strategy = "proportional",
  constant = NULL,
  random_state = NULL
)
```

### Arguments

| | |
|---|---|
| y | a categorical vector, containing the outcomes of interest |
| strategy | a strategy from "constant", "most_frequent", "proportional", "uniform", or "stratified". |
| constant | a constant value for the constant strategy. |
| random_state | a random seed. |

### Value

a list

### Examples

```
# Split the data into training and testing sets
set.seed(2023)
index <- sample(1:nrow(iris), nrow(iris) * 0.8)
train_data <- iris[index,]
test_data <- iris[-index,]
dummy_model <- dummy_classifier(train_data$Species, strategy = "proportional", random_state = 2024)
dummy_model
```

---

## dummy_classifier_caret

*dummy classifier for a categorical variable, used with the train function in caret.*

---

### Description

dummy classifier for a categorical variable, used with the train function in caret.

### Usage

```
dummy_classifier_caret(
  strategy = "proportional",
  constant = NULL,
  random_state = NULL
)
```

### Arguments

| | |
|---|---|
| strategy | a strategy from "constant", "most_frequent", "proportional", "uniform", or "stratified". |
| constant | a constant value for the constant strategy. |
| random_state | a random seed. |

### Value

a list

dummy_regressor *dummy regressor for a numerical variable.*

## Description

dummy regressor for a numerical variable.

## Usage

```
dummy_regressor(y, strategy = "mean", quantile = NULL, constant = NULL)
```

## Arguments

| | |
|---|---|
| y | a numerical vector. |
| strategy | a strategy from "constant", "mean", "median", or "quantile". |
| quantile | used when using the quantile strategy. It is a value between 0 and 1. |
| constant | used when using the constant strategy. It is a numeric value. |

## Value

a list containing information of the model.

## Examples

```
# Split the data into training and testing sets
set.seed(2023)
index <- sample(1:nrow(iris), nrow(iris) * 0.8)
train_data <- iris[index,]
test_data <- iris[-index,]
reg_model <- dummy_regressor(train_data$Sepal.Length, strategy = "median")
reg_model
```

dummy_regressor_caret *dummy regressor for a numerical variable, used in the train function in caret.*

## Description

dummy regressor for a numerical variable, used in the train function in caret.

## Usage

```
dummy_regressor_caret(strategy = "mean", quantile = NULL, constant = NULL)
```

### Arguments

| | |
|---|---|
| strategy | a strategy from "constant", "mean", "median", or "quantile". |
| quantile | used when using the quantile strategy. It is a value between 0 and 1. |
| constant | used when using the constant strategy. It is a numeric value. |

### Value

a list containing information of the model.

---

## predict_dummy_classifier

*dummy classifier predictor*

---

### Description

dummy classifier predictor

### Usage

```
predict_dummy_classifier(object, X)
```

### Arguments

| | |
|---|---|
| object | a list created using dummy classifier. |
| X | a data frame. |

### Value

predicted values for the response variable.

### Examples

```
# Split the data into training and testing sets
set.seed(2023)
index <- sample(1:nrow(iris), nrow(iris) * 0.8)
train_data <- iris[index,]
test_data <- iris[-index,]
dummy_model <- dummy_classifier(train_data$Species, strategy = "proportional", random_state = 2024)

# Make predictions using the trained dummy classifier
pred_vec <- predict_dummy_classifier(dummy_model, test_data)

# Evaluate the performance of the dummy classifier
conf_matrix <- caret::confusionMatrix(pred_vec, test_data$Species)
print(conf_matrix)
```

predict_dummy_regressor *dummy regressor predictor*

## Description

dummy regressor predictor

## Usage

```
predict_dummy_regressor(object, X)
```

## Arguments

| | |
|---|---|
| object | a list from the dummy_regressor function |
| X | a data frame |

## Value

the predicted values

## Examples

```
#' # Split the data into training and testing sets
set.seed(2023)
index <- sample(1:nrow(iris), nrow(iris) * 0.8)
train_data <- iris[index,]
test_data <- iris[-index,]

# Make predictions using the trained dummy regressor
reg_model <- dummy_regressor(train_data$Sepal.Length, strategy = "median")
y_hat <- predict_dummy_regressor(reg_model, test_data)
# Find mean squared error
mean((test_data$Sepal.Length-y_hat)^2)
```

predict_proba *probabilities for predicting classes*

## Description

probabilities for predicting classes

## Usage

```
predict_proba(model, X)
```

## Arguments

| | |
|---|---|
| `model` | a list from dummy classifier. |
| `X` | a data frame. |

## Value

a probability matrix.

# Index