

# Package ‘civic.icarm’

June 22, 2026

**Title** Interpretable Civic-Accountable and Responsible Machine Learning

**Version** 0.3.0

**Description** A general-purpose framework for Interpretable Civic-Accountable and Responsible Machine Learning (ICARM). Works with any clean tabular data and automatically detects whether a task is binary classification, multi-class classification, or regression from the target variable type. Provides a single unified entry point `civic_fit()` alongside tidy interfaces for global and local model explanations, group-level fairness auditing, probability calibration, multi-model comparison, threshold analysis, and reproducible audit trails. Designed to support the DataCitizen-Pro research agenda at Ludwigsburg University of Education: developing data literacy, statistical reasoning, and democratic judgment formation in civic and political teacher education.

References: Biecek (2018) <[doi:10.18637/jss.v085.i04](https://doi.org/10.18637/jss.v085.i04)>,

Kuhn (2008) <[doi:10.18637/jss.v028.i05](https://doi.org/10.18637/jss.v028.i05)>,

Awe (2025) <<https://github.com/Olawaleawe/civic.icarm>>.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Language** en-GB

**Depends** R (>= 4.1.0)

**Imports** stats, utils, rpart, ggplot2, dplyr, tidyr, tibble, purrr,  
rlang, jsonlite, digest

**Suggests** DALEX, glmnet, mgcv, pROC, nnet, testthat, covr, gridExtra

**Config/testthat/edition** 3

**LazyData** true

**RoxygenNote** 7.3.3

**NeedsCompilation** no

**Author** Olushina Olawale Awe [aut, cre],  
Ludwigsburg University of Education [fnd]

**Maintainer** Olushina Olawale Awe <[olawaleawe@gmail.com](mailto:olawaleawe@gmail.com)>

**Repository** CRAN

**Date/Publication** 2026-06-22 13:10:11 UTC

## Contents

civic_audit . . . . .	2
civic_calibrate . . . . .	3
civic_compare . . . . .	4
civic_dashboard . . . . .	5
civic_education . . . . .	6
civic_equalized_odds_curve . . . . .	6
civic_equity_summary . . . . .	7
civic_explain . . . . .	7
civic_explain_local . . . . .	8
civic_fairness . . . . .	8
civic_fit . . . . .	9
civic_german_credit . . . . .	11
civic_metrics . . . . .	12
civic_plots . . . . .	13
civic_plot_calibration . . . . .	13
civic_plot_comparison . . . . .	14
civic_plot_confusion . . . . .	14
civic_plot_fairness . . . . .	15
civic_plot_importance . . . . .	16
civic_plot_roc_groups . . . . .	16
civic_plot_thresholds . . . . .	17
civic_racism_survey . . . . .	17
civic_scorecard . . . . .	19
civic_split . . . . .	20
civic_thresholds . . . . .	21
civic_voting . . . . .	22
predict.civic_model . . . . .	22
print.civic_model . . . . .	23
summary.civic_model . . . . .	23
<b>Index</b>	<b>25</b>

---

civic\_audit

*Generate a reproducible audit trail*

---

### Description

Produces a structured JSON audit record for any ‘civic\_model’: provenance metadata (data hash, seed, analyst, timestamp), performance metrics, equity summary, and DataCitizen-Pro annotations.

**Usage**

```
civic_audit(
  object,
  metrics = NULL,
  fairness = NULL,
  notes = NULL,
  analyst = NULL,
  path = NULL
)
```

**Arguments**

object	A 'civic_model'.
metrics	Named numeric vector from [civic_metrics()] (optional).
fairness	A 'civic_fairness' from [civic_fairness()] (optional).
notes	Character analyst notes (optional).
analyst	Character analyst name (optional).
path	File path to write the JSON (optional).

**Value**

Invisibly, the JSON character string.

**Examples**

```
m <- civic_fit(voted ~ age + education, civic_voting)
trail <- civic_audit(m, analyst = "O. O. Awe", notes = "Baseline")
cat(trail)
```

---

civic_calibrate	<i>Assess probability calibration (binary classification)</i>
-----------------	---

---

**Description**

Evaluates whether predicted probabilities are well-calibrated: a model predicting 0.7 should be correct ~70 Returns Brier score and Expected Calibration Error (ECE).

**Usage**

```
civic_calibrate(object, data, outcome, positive = NULL, n_bins = 10L)
```

**Arguments**

object	A 'civic_model' (binary classification only).
data	A data frame for evaluation.
outcome	Character. Outcome column name.
positive	Character. Positive class level.
n_bins	Integer. Number of probability bins (default '10').

**Value**

An object of class 'civic\_calibration' (a list) with: 'bins' (tibble), 'brier\_score', 'ece', 'positive', 'outcome', 'model'.

**Examples**

```
m <- civic_fit(voted ~ age + education, civic_voting)
cal <- civic_calibrate(m, civic_voting, "voted", "yes")
print(cal)
civic_plot_calibration(cal)
```

---

civic\_compare

---

*Compare multiple civic\_models on a shared test set*


---

**Description**

Evaluates a named list of 'civic\_model' objects on a common test set and returns a tidy comparison table of performance, fairness, and interpretability. All models must have the same task type.

**Usage**

```
civic_compare(
  models,
  test_data,
  outcome,
  protected = NULL,
  positive = NULL,
  threshold = 0.5
)
```

**Arguments**

models	A <b>named</b> list of 'civic_model' objects, e.g. 'list(CART = m1, Logistic = m2)'.
test_data	A data frame used for all evaluations.
outcome	Character. Outcome column name.
protected	Character. Protected attribute column (optional). Pass 'NULL' to skip fairness metrics.
positive	Positive class level (binary only).
threshold	Decision threshold (binary only, default '0.5').

**Value**

A tibble of class 'civic\_comparison'.

**Examples**

```

splits <- civic_split(iris, stratify = "Species")
m1 <- civic_fit(Species ~ ., splits$train, model = "cart")
m2 <- civic_fit(Species ~ ., splits$train, model = "multinomial")
cmp <- civic_compare(list(CART = m1, Multinomial = m2),
                      splits$test, outcome = "Species")
civic_plot_comparison(cmp)

```

---

civic\_dashboard

*Generate a 4-panel civic accountability dashboard*


---

**Description**

Produces a single 2x2 summary dashboard combining feature importance, fairness audit, confusion matrix or actual vs predicted plot, and performance metrics bar chart. Requires gridExtra.

**Usage**

```

civic_dashboard(object, test_data, outcome, protected = NULL,
                positive = NULL, title = NULL, analyst = NULL)

```

**Arguments**

object	A civic_model from civic_fit().
test_data	A data frame for evaluation.
outcome	Character. Outcome column name.
protected	Character. Protected attribute column (optional).
positive	Character. Positive class for binary (optional).
title	Character. Overall dashboard title (optional).
analyst	Character. Analyst name for caption (optional).

**Value**

A grid object printed to the graphics device. Requires gridExtra.

**Examples**

```

## Not run:
data(civic_voting)
sp <- civic_split(civic_voting, stratify = "voted")
m <- civic_fit(voted ~ age + education,
               sp$train, model = "logistic", positive = "yes")
civic_dashboard(m, sp$test, outcome = "voted",
                protected = "gender", positive = "yes")

## End(Not run)

```

---

civic_education	<i>civic_education dataset</i>
-----------------	--------------------------------

---

**Description**

Synthetic civic education outcomes data.

**Usage**

```
civic_education
```

**Format**

A tibble with 800 rows and 9 variables.

**Source**

Synthetic data generated by civic.icarm team.

---

civic_equalized_odds_curve	<i>Compute equalized odds curves across thresholds (binary only)</i>
----------------------------	--

---

**Description**

Compute equalized odds curves across thresholds (binary only)

**Usage**

```
civic_equalized_odds_curve(
  object,
  data,
  outcome,
  protected,
  positive = NULL,
  thresholds = seq(0.05, 0.95, 0.05)
)
```

**Arguments**

object	A 'civic_model' (binary classification).
data	A data frame.
outcome	Character outcome column name.
protected	Character protected attribute column name.
positive	Positive class level.
thresholds	Numeric vector of thresholds.

**Value**

A tibble with columns: 'threshold', 'group', 'tpr', 'fpr', 'tnr'.

---

civic\_equity\_summary    *Summarise fairness into scalar equity indicators*

---

**Description**

Summarise fairness into scalar equity indicators

**Usage**

```
civic_equity_summary(fairness)
```

**Arguments**

fairness            A 'civic\_fairness' from [civic\_fairness()].

**Value**

A named list of scalar equity indicators.

---

civic\_explain            *Generate global model explanations*

---

**Description**

Creates a civic\_explainer containing feature importance and optionally a DALEX explainer for PDP/ICE and local explanations. Works for all task types: binary, multiclass, and regression.

**Usage**

```
civic_explain(object, data = NULL, label = NULL)
```

**Arguments**

object            A civic\_model from civic\_fit().  
data              Optional data frame for DALEX explainer.  
label             Optional character label for the DALEX explainer.

**Value**

An object of class civic\_explainer.

**Examples**

```
m <- civic_fit(Species ~ ., iris)
ex <- civic_explain(m)
print(ex)
```

---

civic\_explain\_local     *Generate local instance-level explanations*

---

**Description**

Explains why the model made a specific prediction for one or more individual observations. Uses DALEX break-down if available, falls back to coefficient contributions for GLM and LM models.

**Usage**

```
civic_explain_local(explainer, newdata, n_features = 10L)
```

**Arguments**

explainer	A civic_explainer from civic_explain().
newdata	A data frame of observations to explain.
n_features	Integer. Maximum features to show. Default 10.

**Value**

A list of tibbles, one per row of newdata.

**Examples**

```
m <- civic_fit(Species ~ ., iris)
ex <- civic_explain(m)
civic_explain_local(ex, iris[1, ])
```

---

civic\_fairness     *Compute group-level fairness metrics*

---

**Description**

Evaluates a 'civic\_model' across levels of a protected attribute, computing standard algorithmic fairness metrics. Works for binary classification, multi-class classification, and regression.

**\*\*Binary classification metrics (per group):\*\*** 'n', 'acc', 'tpr', 'tnr', 'fpr', 'fnr', 'ppv', 'rate\_pos', 'mean\_prob', 'acc\_gap', 'tpr\_gap', 'fpr\_gap', 'dp\_ratio' (disparate impact), 'eo\_gap' (equalized odds gap).

**\*\*Multi-class metrics (per group):\*\*** 'n', 'acc', 'balanced\_acc', 'acc\_gap'.

**\*\*Regression metrics (per group):\*\*** 'n', 'mae', 'rmse', 'mae\_gap', 'rmse\_gap'.

**Usage**

```
civic_fairness(
  object,
  data,
  outcome,
  protected,
  positive = NULL,
  threshold = 0.5
)
```

**Arguments**

object	A ‘civic_model’ from [civic_fit()].
data	A ‘data.frame’ containing features, outcome, and protected column.
outcome	Character. Name of the outcome/target column.
protected	Character. Name of the protected attribute column (e.g., “gender”, “ethnicity”, “age_group”).
positive	Character. Positive class for binary classification. Defaults to ‘object\$positive’.
threshold	Decision threshold for binary classification (default ‘0.5’).

**Value**

A tibble of class ‘civic\_fairness’ with one row per group.

**Examples**

```
# Binary classification
m <- civic_fit(voted ~ age + education, civic_voting)
civic_fairness(m, civic_voting, outcome = "voted",
              protected = "gender", positive = "yes")

# Regression
m2 <- civic_fit(mpg ~ cyl + wt + hp, mtcars)
mtcars$gear_grp <- factor(mtcars$gear)
civic_fairness(m2, mtcars, outcome = "mpg", protected = "gear_grp")

# Any data – works with iris too
m3 <- civic_fit(Sepal.Length ~ Sepal.Width + Petal.Length, iris)
civic_fairness(m3, iris, outcome = "Sepal.Length", protected = "Species")
```

## Description

Single unified entry point for all civic.icarm modelling. Automatically detects the prediction task from your target variable — you do not need to choose between classification and regression upfront.

**\*\*Task auto-detection rules:\*\*** | Target type | Task | Default model | |—|—|—| | 'factor' / 'character', 2 levels | Binary classification | "cart" | | 'factor' / 'character', 3+ levels | Multi-class classification | "cart" | | 'numeric' / 'integer' | Regression | "cart" |

**\*\*Supported models:\*\***

**\*Binary classification:** - "cart" — Classification tree (rpart). Fully inspectable. - "logistic" — Logistic regression (stats::glm). Coefficient-interpretable. - "logistic\_l1" — L1-penalised logistic (glmnet). Requires 'glmnet'.

**\*Multi-class classification:** - "cart" — Classification tree (rpart). Handles any number of classes. - "multinomial" — Multinomial logistic regression (nnet). Requires 'nnet'.

**\*Regression:** - "cart" — Regression tree (rpart). - "linear" — Ordinary least squares (stats::lm). - "gam" — Generalised Additive Model (mgcv). Requires 'mgcv'.

## Usage

```
civic_fit(
  formula,
  data,
  task = "auto",
  model = "auto",
  seed = 2025L,
  cart_control = NULL,
  positive = NULL,
  ...
)
```

## Arguments

formula	A model formula. Use '.' for all columns: 'target ~ .' or 'target ~ x1 + x2 + x3'.
data	A 'data.frame' or 'tibble' of training data.
task	One of "auto" (default), "binary", "multiclass", or "regression". Use "auto" to let the package detect the task.
model	Character. Model type. Use "auto" to let the package pick a sensible default, or specify one explicitly (see above).
seed	Integer. Random seed recorded for reproducibility (default 2025).
cart_control	A [rpart::rpart.control()] list for tuning CART trees. Default: 'cp = 0.01', 'minsplit = 20'.
positive	Character. For binary classification: which factor level is the "positive" class. If 'NULL', uses the first factor level.
...	Additional arguments passed to the underlying model fitter.

**Value**

An S3 object of class ‘civic\_model’ containing:

- ‘fit’ The underlying fitted model object.
- ‘task’ Detected/specified task: “binary”, “multiclass”, or “regression”.
- ‘model’ Model type string.
- ‘formula’ The model formula used.
- ‘outcome’ Name of the target/outcome variable.
- ‘levels’ Factor levels (classification only).
- ‘positive’ Positive class (binary classification only).
- ‘seed’ Random seed used.
- ‘n\_train’ Number of training rows.
- ‘data\_hash’ SHA-256 digest of training data for provenance.
- ‘trained\_at’ POSIXct timestamp.
- ‘n\_features’ Number of predictor features.
- ‘feature\_names’ Names of predictor features.

**Examples**

```
# Binary classification (auto-detected from factor target)
data(civic_voting)
m <- civic_fit(voted ~ age + education + political_interest,
              data = civic_voting)
print(m)

# Regression (auto-detected from numeric target)
data(civic_education)
m2 <- civic_fit(civic_knowledge_score ~ age + stats_course + news_consumption,
               data = civic_education)

# Explicit model choice
m3 <- civic_fit(voted ~ ., data = civic_voting, model = "logistic")

# Works on any data frame - here using the built-in iris dataset
m4 <- civic_fit(Species ~ ., data = iris) # multi-class
m5 <- civic_fit(Sepal.Length ~ ., data = iris) # regression
```

---

civic\_german\_credit    *civic\_german\_credit dataset*

---

**Description**

Synthetic German credit scoring fairness benchmark.

**Usage**

```
civic_german_credit
```

**Format**

A tibble with 1000 rows and 8 variables.

**Source**

Synthetic data generated by civic.icarm team.

---

civic_metrics	<i>Compute performance metrics for any task type</i>
---------------	--

---

**Description**

Returns a named numeric vector of performance metrics appropriate for the task. Task is inferred automatically unless 'type' is given.

**\*\*Binary / multi-class classification metrics:\*\*** 'accuracy', 'balanced\_acc', 'f1', 'precision', 'recall', 'specificity' (binary only), 'auc' (binary only, requires 'pROC').

**\*\*Regression metrics:\*\*** 'mae', 'rmse', 'r2'.

**Usage**

```
civic_metrics(y_true, y_pred, y_prob = NULL, positive = NULL, type = "auto")
```

**Arguments**

y_true	True outcome values (factor or numeric).
y_pred	Predicted values (factor/character for classification, numeric for regression).
y_prob	Numeric probability vector for the <b>**positive**</b> class (binary classification only). Used to compute AUC.
positive	Character. Positive class level (binary classification). Defaults to first factor level.
type	One of "auto" (default), "binary", "multiclass", or "regression".

**Value**

A named numeric vector of metrics.

**Examples**

```
# Classification
y <- factor(c("yes","no","yes","yes","no","no"))
yhat <- factor(c("yes","no","no","yes","no","yes"))
civic_metrics(y, yhat, positive = "yes")

# Regression (any numeric target)
y2 <- c(10, 20, 30, 40, 50)
yhat2 <- c(12, 18, 33, 39, 48)
civic_metrics(y2, yhat2)

# Works with iris
m <- civic_fit(Species ~ ., iris)
yhat3 <- predict(m, iris)
civic_metrics(iris$Species, yhat3)
```

---

civic\_plots

*Visualisation functions for civic.icarm*


---

**Description**

A family of ggplot2-based visualisation functions for civic.icarm model objects. All functions return ggplot2 objects that can be further customised with standard ggplot2 syntax.

**Value**

A ggplot2 object that can be further customised with standard ggplot2 syntax.

---

civic\_plot\_calibration

*Plot calibration curve*


---

**Description**

Plot calibration curve

**Usage**

```
civic_plot_calibration(calibration, title = NULL)
```

**Arguments**

**calibration** A civic\_calibration from civic\_calibrate().  
**title** Optional title.

**Value**

A ggplot2 object.

---

civic\_plot\_comparison *Plot multi-model comparison*

---

### Description

Automatically detects whether models were trained for regression or classification and selects appropriate metrics accordingly.

### Usage

```
civic_plot_comparison(comparison, metrics = NULL, title = NULL)
```

### Arguments

comparison	A civic_comparison from civic_compare().
metrics	Optional character vector of metric columns to plot. If NULL, metrics are chosen automatically based on task type.
title	Optional title.

### Value

A ggplot2 object.

### Examples

```
sp <- civic_split(iris, stratify="Species")
m1 <- civic_fit(Species ~ ., sp$train, model="cart")
m2 <- civic_fit(Species ~ ., sp$train, model="multinomial")
cmp <- civic_compare(list(CART=m1, Multinom=m2),
                     sp$test, outcome="Species")
civic_plot_comparison(cmp)
```

---

civic\_plot\_confusion *Plot confusion matrix*

---

### Description

Plot confusion matrix

### Usage

```
civic_plot_confusion(y_true, y_pred, title = NULL)
```

**Arguments**

y_true	Factor of true outcomes.
y_pred	Factor of predicted outcomes.
title	Optional title.

**Value**

A ggplot2 object.

**Examples**

```
m <- civic_fit(Species ~ ., iris)
yhat <- predict(m, iris)
civic_plot_confusion(iris$Species, yhat)
```

---

civic\_plot\_fairness *Plot group-level fairness metric*

---

**Description**

Plot group-level fairness metric

**Usage**

```
civic_plot_fairness(fairness, metric = "acc", title = NULL, ref_line = NULL)
```

**Arguments**

fairness	A civic_fairness from civic_fairness().
metric	Character. Column name to plot.
title	Optional title.
ref_line	Optional numeric reference line.

**Value**

A ggplot2 object.

**Examples**

```
m <- civic_fit(Species ~ ., iris)
iris2 <- iris
iris2$size <- factor(iffelse(iris2$Sepal.Length>5.8,"large","small"))
f <- civic_fairness(m, iris2, "Species", "size")
civic_plot_fairness(f, metric="acc")
```

---

civic\_plot\_importance *Plot feature importance*

---

**Description**

Plot feature importance

**Usage**

```
civic_plot_importance(explainer, n_features = 15L, title = NULL)
```

**Arguments**

explainer	A civic_explainer from civic_explain().
n_features	Max features to display. Default 15.
title	Optional plot title.

**Value**

A ggplot2 object.

**Examples**

```
m <- civic_fit(Species ~ ., iris)
ex <- civic_explain(m)
civic_plot_importance(ex)
```

---

civic\_plot\_roc\_groups *Plot per-group ROC curves*

---

**Description**

Plot per-group ROC curves

**Usage**

```
civic_plot_roc_groups(eoc_tbl, title = NULL)
```

**Arguments**

eoc_tbl	A tibble from civic_equalized_odds_curve().
title	Optional title.

**Value**

A ggplot2 object.

---

civic\_plot\_thresholds *Plot threshold performance curves*

---

**Description**

Plot threshold performance curves

**Usage**

```
civic_plot_thresholds(  
  thresholds_tbl,  
  metrics = c("accuracy", "recall", "precision", "f1"),  
  title = NULL  
)
```

**Arguments**

thresholds\_tbl A tibble from civic\_thresholds().  
metrics Character vector of metric columns to plot.  
title Optional title.

**Value**

A ggplot2 object.

---

civic\_racism\_survey *Synthetic Racism and Civic Participation Survey*

---

**Description**

A synthetic dataset of 150 individuals capturing experiences of racism, policing, migration, and civic participation. Designed for fairness auditing, regression, binary, and multi-class classification demonstrations within the DataCitizen-Pro framework.

A synthetic dataset of 1000 individuals capturing experiences of racism, policing, migration, and civic participation. Designed for fairness auditing, regression, binary, and multi-class classification demonstrations within the DataCitizen-Pro framework.

**Usage**

```
civic_racism_survey  
  
civic_racism_survey
```

**Format**

A tibble with 150 rows and 16 variables:

**age** Integer. Age in years (18-75).  
**gender** Factor. male, female, diverse.  
**hair\_color** Factor. blonde, brown, black, other.  
**skin\_color** Factor. light, medium, dark.  
**relationship\_status** Factor. single, engaged, married.  
**racism\_impact** Integer 0-10. Perceived racism impact.  
**police\_stop** Factor. 0 times, 1 time, 2 or more.  
**migrant\_status** Factor. Yes or No.  
**income** Numeric. Monthly income in EUR.  
**education\_level** Factor. Education category.  
**employment\_status** Factor. Employment category.  
**area\_type** Factor. Urban or Rural.  
**religion** Factor. Religion category.  
**language\_proficiency** Factor. Language level.  
**number\_of\_friends** Integer 0-10.  
**political\_orientation** Integer 0-3. Left to right.

A tibble with 150 rows and 16 variables:

**age** Integer. Age in years (18-75).  
**gender** Factor. male, female, diverse.  
**hair\_color** Factor. blonde, brown, black, other.  
**skin\_color** Factor. light, brown, dark.  
**relationship\_status** Factor. single, engaged, married.  
**racism\_impact** Integer 0-10. Perceived racism impact.  
**police\_stop** Factor. 0 times, 1 time, 2 or more.  
**migrant\_status** Factor. Yes or No.  
**income** Numeric. Monthly income in EUR.  
**education\_level** Factor. Education category.  
**employment\_status** Factor. Employment category.  
**area\_type** Factor. Urban or Rural.  
**religion** Factor. Religion category.  
**language\_proficiency** Factor. Language level.  
**number\_of\_friends** Integer 0-10.  
**political\_orientation** Integer 0-3. Left to right.

**Source**

Synthetic data generated by the civic.icarm team.

Synthetic data generated by the civic.icarm team.

**Examples**

```
data(civic_racism_survey)
m <- civic_fit(racism_impact ~ ., civic_racism_survey,
              model = "linear")
ex <- civic_explain(m)
data(civic_racism_survey)
m <- civic_fit(racism_impact ~ ., civic_racism_survey,
              model = "linear")
```

---

civic_scorecard	<i>Generate a full civic accountability scorecard</i>
-----------------	---

---

**Description**

Synthesises model provenance, interpretability rating, performance, and equity into a printed civic accountability scorecard, with an optional JSON output. Works for all task types.

**Usage**

```
civic_scorecard(
  object,
  test_data,
  outcome,
  protected = NULL,
  positive = NULL,
  analyst = NULL,
  project = "civic.icarm",
  path = NULL
)
```

**Arguments**

object	A 'civic_model'.
test_data	Data frame of held-out test data.
outcome	Character. Outcome column name.
protected	Character. Protected attribute column (optional).
positive	Positive class (binary only).
analyst	Character analyst name.
project	Character project name.
path	Optional file path for JSON output.

**Value**

Invisibly, a named list (the scorecard structure).

**Examples**

```
splits <- civic_split(civic_voting, stratify = "voted")
m <- civic_fit(voted ~ age + education + political_interest, splits$train)
civic_scorecard(m, splits$test, outcome = "voted",
  protected = "gender", positive = "yes",
  project = "DataCitizen-Pro")
```

---

civic\_split

*Reproducible train/test split*


---

**Description**

Splits a data frame into training and test sets. The seed is always stored in the returned object so the split is fully reproducible. Optional stratification preserves class proportions.

**Usage**

```
civic_split(data, prop = 0.75, seed = 2025L, stratify = NULL)
```

**Arguments**

data	A ‘data.frame’ or ‘tibble’.
prop	Proportion for training (default ‘0.75’).
seed	Integer random seed (default ‘2025’).
stratify	Optional column name (character) to stratify on. Ensures class proportions are preserved in both splits. Works for both factor (classification) and numeric targets (stratifies by quartile).

**Value**

A named list with elements ‘train’, ‘test’, ‘seed’, and ‘prop’.

**Examples**

```
# Any data frame works
splits <- civic_split(iris, prop = 0.8, stratify = "Species")
nrow(splits$train) # ~120
nrow(splits$test)  # ~30

# Numeric stratification (by quartile)
splits2 <- civic_split(mtcars, prop = 0.75, stratify = "mpg")
```

---

civic_thresholds	<i>Threshold sweep for binary classification</i>
------------------	--

---

## Description

Computes performance metrics across a grid of decision thresholds. Essential for understanding the accuracy-vs-fairness tradeoffs that arise when choosing a classification cutoff — a DataCitizen-Pro democratic judgment teaching tool.

## Usage

```
civic_thresholds(  
  y_true,  
  y_prob,  
  positive = NULL,  
  thresholds = seq(0.1, 0.9, by = 0.05)  
)
```

## Arguments

y_true	Factor of true class labels.
y_prob	Numeric vector of predicted probabilities for the positive class.
positive	Character. Positive class level.
thresholds	Numeric vector of thresholds to evaluate. Default: 'seq(0.1, 0.9, by = 0.05)'.

## Value

A tibble with one row per threshold and columns: 'threshold', 'accuracy', 'balanced\_acc', 'precision', 'recall', 'specificity', 'f1', 'rate\_positive'.

## Examples

```
y <- factor(sample(c("yes", "no"), 200, replace = TRUE))  
p <- runif(200)  
thr <- civic_thresholds(y, p, positive = "yes")  
civic_plot_thresholds(thr)
```

---

civic_voting	<i>civic_voting dataset</i>
--------------	-----------------------------

---

**Description**

Synthetic civic voting participation data.

**Usage**

```
civic_voting
```

**Format**

A tibble with 1000 rows and 10 variables.

**Source**

Synthetic data generated by civic.icarm team.

---

predict.civic_model	<i>Predict from a civic_model</i>
---------------------	-----------------------------------

---

**Description**

Generates predictions from a fitted civic\_model object for any task type.

**Usage**

```
## S3 method for class 'civic_model'
predict(object, newdata,
        type = c("class", "prob"), threshold = 0.5, ...)
```

**Arguments**

object	A civic_model.
newdata	A data frame for prediction.
type	For classification: "class" or "prob". For regression: ignored.
threshold	Decision threshold for binary (default 0.5).
...	Ignored.

**Value**

For classification with type = "class": a factor vector. For classification with type = "prob": a numeric matrix. For regression: a numeric vector.

**Examples**

```
data(civic_voting)
m <- civic_fit(voted ~ age + education, data = civic_voting)
predict(m, civic_voting[1:5, ], type = "class")
predict(m, civic_voting[1:5, ], type = "prob")
```

---

print.civic\_model      *Print a civic\_model*

---

**Description**

Print a civic\_model

**Usage**

```
## S3 method for class 'civic_model'
print(x, ...)
```

**Arguments**

x                    A civic\_model object.  
...                  Further arguments passed to or from other methods.

**Value**

Invisibly returns the civic\_model object x. Called for its side effect of printing a formatted summary to the console.

---

summary.civic\_model      *Summary of a civic\_model*

---

**Description**

Summary of a civic\_model

**Usage**

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

**Arguments**

object                A civic\_model object.  
...                    Further arguments passed to or from other methods.

**Value**

Invisibly returns the summary of the underlying fitted model. Called for its side effect of printing a detailed model summary to the console.

# Index

## \* datasets

- civic\_education, [6](#)
- civic\_german\_credit, [11](#)
- civic\_racism\_survey, [17](#)
- civic\_voting, [22](#)

- civic\_audit, [2](#)
- civic\_calibrate, [3](#)
- civic\_compare, [4](#)
- civic\_dashboard, [5](#)
- civic\_education, [6](#)
- civic\_equalized\_odds\_curve, [6](#)
- civic\_equity\_summary, [7](#)
- civic\_explain, [7](#)
- civic\_explain\_local, [8](#)
- civic\_fairness, [8](#)
- civic\_fit, [9](#)
- civic\_german\_credit, [11](#)
- civic\_metrics, [12](#)
- civic\_plot\_calibration, [13](#)
- civic\_plot\_comparison, [14](#)
- civic\_plot\_confusion, [14](#)
- civic\_plot\_fairness, [15](#)
- civic\_plot\_importance, [16](#)
- civic\_plot\_roc\_groups, [16](#)
- civic\_plot\_thresholds, [17](#)
- civic\_plots, [13](#)
- civic\_racism\_survey, [17](#)
- civic\_scorecard, [19](#)
- civic\_split, [20](#)
- civic\_thresholds, [21](#)
- civic\_voting, [22](#)

- predict.civic\_model, [22](#)
- print.civic\_model, [23](#)

- summary.civic\_model, [23](#)