

# Package ‘tidydann’

July 22, 2025

**Title** Add the 'dann' Model and the 'sub\_dann' Model to the Tidymodels Ecosystem

**Version** 1.0.1

**Description** Provides model specifications, tuning parameters for models in 'dann' package. Models based on Hastie (1996)

<[https://web.stanford.edu/~hastie/Papers/dann\\_IEEE.pdf](https://web.stanford.edu/~hastie/Papers/dann_IEEE.pdf)>.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Depends** R (>= 4.1.0)

**Suggests** testthat (>= 3.0.0), dann (>= 1.0.0), recipes, mlbench, modeldata, workflows, rsample, dplyr, magrittr, tune, scales, yardstick, rlang

**Config/testthat/edition** 3

**Imports** dials, generics, tibble, parsnip

**URL** <https://github.com/gmcmacran/tidydann>

**BugReports** <https://github.com/gmcmacran/tidydann/issues>

**NeedsCompilation** no

**Author** Greg McMahan [aut, cre]

**Maintainer** Greg McMahan <gmcmacran@gmail.com>

**Repository** CRAN

**Date/Publication** 2025-04-12 21:00:02 UTC

## Contents

matrix_diagonal . . . . .	2
nearest_neighbor_adaptive . . . . .	2
neighborhood . . . . .	4
sphere . . . . .	5
tunable.nearest_neighbor_adaptive . . . . .	5
update.nearest_neighbor_adaptive . . . . .	6
weighted . . . . .	7

**Index****8**


---

matrix_diagonal	<i>Softening</i>
-----------------	------------------

---

**Description**

Softening

**Usage**

```
matrix_diagonal(range = c(0, 2), trans = NULL)
```

**Arguments**

range	A two-element vector holding the defaults for the smallest and largest possible values, respectively. If a transformation is specified, these values should be in the transformed units.
trans	A trans object from the scales package, such as scales::log10_trans() or scales::reciprocal_trans(). If not provided, the default is used which matches the units used in range. If no transformation, NULL.

**Details**

Softening parameter. Usually has the least affect on performance.

**Value**

An S3 class of type quant\_param from the dials package.

**Examples**

```
library(tidyann)

matrix_diagonal()
```

---

nearest_neighbor_adaptive	<i>Discriminant Adaptive Nearest Neighbor Classification</i>
---------------------------	--

---

**Description**

Discriminant Adaptive Nearest Neighbor Classification

**Usage**

```
nearest_neighbor_adaptive(
  mode = "classification",
  neighbors = NULL,
  neighborhood = NULL,
  matrix_diagonal = NULL,
  weighted = NULL,
  sphere = NULL,
  num_comp = NULL
)
```

**Arguments**

mode	A single character string for the type of model. The only possible value for this model is "classification".
neighbors	The number of data points used for final classification.
neighborhood	The number of data points used to calculate between and within class covariance.
matrix_diagonal	Diagonal elements of a diagonal matrix. 1 is the identity matrix.
weighted	weighted argument to <code>ncoord</code> . See <a href="#">fpc::ncoord()</a> for details. Only <code>sub_dann</code> engine.
sphere	One of "mcd", "mve", "classical", or "none" See <a href="#">fpc::ncoord()</a> for details. Only <code>sub_dann</code> engine.
num_comp	Dimension of subspace used by <code>dann</code> . See <a href="#">fpc::ncoord()</a> for details. Only <code>sub_dann</code> engine.

**Details**

Discriminant Adaptive Nearest Neighbor (`dann`) is a variation of  $k$  nearest neighbors where the shape of the neighborhood is data driven. The neighborhood is elongated along class boundaries and shrunk in the orthogonal direction.

This function has engines `dann` and `sub_dann`.

**Value**

An S3 class of type `nearest_neighbor_adaptive`.

**Examples**

```
library(parsnip)
library(tidydann)

data("two_class_dat", package = "modeldata")

model <- nearest_neighbor_adaptive(neighbors = 2) |>
  set_engine("dann") |>
  fit(formula = Class ~ A + B, data = two_class_dat)
```

```
model |>
  predict(new_data = two_class_dat)
```

---

neighborhood	<i>Neighborhood size</i>
--------------	--------------------------

---

### Description

Number of data points used to calculate the shape of the neighborhood.

### Usage

```
neighborhood(range = c(2L, dials::unknown()), trans = NULL)
```

### Arguments

range	A two-element vector holding the defaults for the smallest and largest possible values, respectively. If a transformation is specified, these values should be in the transformed units.
trans	A trans object from the scales package, such as <code>scales::log10_trans()</code> or <code>scales::reciprocal_trans()</code> . If not provided, the default is used which matches the units used in range. If no transformation, NULL.

### Details

Use `get_n` or `finalize` from `dials` to finalize.

If cross validation is done, use `get_n_frac` with argument `frac` set to  $1/V$ . See README for detailed example.

### Value

An S3 class of type `quant_param` from the `dials` package.

### Examples

```
library(dials)
library(tidyann)

data("taxi", package = "modeldata")
neighborhood() |> finalize(taxi)

neighborhood() |> get_n(taxi)
```

---

sphere	<i>Sphere argument to ncoord</i>
--------	----------------------------------

---

**Description**

Sphere argument to ncoord

**Usage**

```
sphere(values = c("mcd", "mve", "classical", "none"))
```

**Arguments**

values            A one-element vector containing "mcd", "mve", "classical", or "none".

**Value**

An S3 class of type qual\_param from the dials package.

**Examples**

```
library(tidyann)
```

```
sphere()
```

---

tunable.nearest_neighbor_adaptive	<i>Declare tunable parameters</i>
-----------------------------------	-----------------------------------

---

**Description**

Returns information on potential hyper-parameters that can be optimized.

**Usage**

```
## S3 method for class 'nearest_neighbor_adaptive'
tunable(x, ...)
```

**Arguments**

x                    A model specification of type nearest\_neighbor\_adaptive specification.  
 ...                 Other arguments passed to methods.

**Value**

A tibble with a column for the parameter name, information on the default method for generating a corresponding parameter object, the source of the parameter (e.g. "recipe", etc.), and the component within the source.

---

```
update.nearest_neighbor_adaptive
```

*Updating a model specification.*

---

### Description

If parameters of a model specification need to be modified, `update()` can be used in lieu of recreating the object from scratch.

### Usage

```
## S3 method for class 'nearest_neighbor_adaptive'
update(
  object,
  parameters = NULL,
  neighbors = NULL,
  neighborhood = NULL,
  matrix_diagonal = NULL,
  weighted = NULL,
  sphere = NULL,
  num_comp = NULL,
  fresh = FALSE,
  ...
)
```

### Arguments

<code>object</code>	A model specification.
<code>parameters</code>	A 1-row tibble or named list with main parameters to update. Use either parameters or the main arguments directly when updating. If the main arguments are used, these will supersede the values in parameters. Also, using engine arguments in this object will result in an error.
<code>neighbors</code>	The number of data points used for final classification.
<code>neighborhood</code>	The number of data points used to calculate between and within class covariance.
<code>matrix_diagonal</code>	Diagonal elements of a diagonal matrix. 1 is the identity matrix.
<code>weighted</code>	weighted argument to <code>ncoord</code> . See <a href="#">fpc::ncoord()</a> for details. Only <code>sub_dann</code> engine.
<code>sphere</code>	One of "mcd", "mve", "classical", or "none" See <a href="#">fpc::ncoord()</a> for details. Only <code>sub_dann</code> engine.
<code>num_comp</code>	Dimension of subspace used by <code>dann</code> . See <a href="#">fpc::ncoord()</a> for details. Only <code>sub_dann</code> engine.
<code>fresh</code>	A logical for whether the arguments should be modified in-place or replaced wholesale.
<code>...</code>	Not used for <code>update()</code> .

---

weighted	<i>Weighted argument to ncoord</i>
----------	------------------------------------

---

**Description**

Weighted argument to ncoord

**Usage**

```
weighted(values = c(FALSE, TRUE))
```

**Arguments**

values            A one-element vector containing FALSE or TRUE.

**Value**

An S3 class of type qual\_param from the dials package.

**Examples**

```
library(tidyann)
```

```
weighted()
```

# Index

`fpc::ncoord()`, [3](#), [6](#)

`matrix_diagonal`, [2](#)

`nearest_neighbor_adaptive`, [2](#)

`neighborhood`, [4](#)

`sphere`, [5](#)

`tunable.nearest_neighbor_adaptive`, [5](#)

`update.nearest_neighbor_adaptive`, [6](#)

`weighted`, [7](#)