

Package: tabpfn (via r-universe)

May 26, 2026

Title Prior-Data Fitted Network Foundational Model for Tabular Data

Version 0.2.0.9000

Description Provides a consistent API for classification and regression models based on the 'TabPFN' model of Hollmann et al. (2025), ``Accurate predictions on small data with a tabular foundation model," Nature, 637(8045) <[doi:10.1038/s41586-024-08328-6](https://doi.org/10.1038/s41586-024-08328-6)>. The calculations are served via 'Python' to train and predict the model.

License Apache License (>= 2)

URL <https://tabpfn.tidymodels.org>,

<https://github.com/tidymodels/tabpfn>

BugReports <https://github.com/tidymodels/tabpfn/issues>

Depends R (>= 4.1.0)

Imports cli, dplyr, generics, hardhat, purrr, reticulate (>= 1.41.0.1), rlang (>= 1.1.0), tibble

Suggests covr, modeldata, recipes, spelling, testthat (>= 3.0.0), withr

Config/Needs/website tidyverse/tidytemplate

Config/testthat/edition 3

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libpng-dev python3

Repository <https://tidymodels.r-universe.dev>

Date/Publication 2026-05-26 14:40:25 UTC

RemoteUrl <https://github.com/tidymodels/tabpfn>

RemoteRef HEAD

RemoteSha 3da6e6e54f4005e3a92be06d2795201d3bf137fb

Contents

control_tab_pfn	2
is_tab_pfn_installed	3
predict.tab_pfn	4
tab_pfn	5
tabpfn_download_models	11
tabpfn_list_versions	11
Index	12

control_tab_pfn	<i>Controlling TabPFN execution</i>
-----------------	-------------------------------------

Description

Controlling TabPFN execution

Usage

```
control_tab_pfn(
    n_preprocessing_jobs = 1L,
    device = "auto",
    ignore_pretraining_limits = FALSE,
    inference_precision = "auto",
    fit_mode = "fit_preprocessors",
    memory_saving_mode = "auto",
    random_state = sample.int(10^6, 1),
    ...
)
```

Arguments

n_preprocessing_jobs	An integer for the number of worker processes. A value of -1L indicates all possible resources.
device	A character value for the device used for torch (e.g., "cpu", "cuda", "mps", etc.). Th default is "auto".
ignore_pretraining_limits	A logical to bypass the default data limits on:the number of training set samples (10,000) and, the number of predictors (500). There is an unchangeable limit to the number of classes (10).
inference_precision	A character value for the trade off between speed and reproducibility. This can be a torch dtype, "autocast" (for torch's mixed-precision autocast), or "auto".
fit_mode	A character value to control how the are preprocessed and/or cached. Values are "fit_preprocessors" (the default), "low_memory", "fit_with_cache", and "batched".

memory_saving_mode	A character string to help with out-of-memory errors. Values are either a logical or "auto".
random_state	An integer to set the random number stream.
...	Additional named arguments passed directly to the TabPFN Python constructor. Use this to supply options not covered by the named parameters above (e.g. arguments added in newer versions of the Python package).

Value

A list with extra class "control_tab_pfn" that has named elements for each of the argument values.

References

<https://github.com/PriorLabs/TabPFN/blob/main/src/tabPFN/classifier.py>, <https://github.com/PriorLabs/TabPFN/blob/main/src/tabPFN/regressor.py>

Examples

```
control_tab_pfn()
```

is_tab_pfn_installed *Check the Python package installation*

Description

Attempts to import the Python package

Usage

```
is_tab_pfn_installed()
```

Value

A single logical

Examples

```
if (interactive()) {  
  # This may take a minute  
  is_tab_pfn_installed()  
}
```

predict.tab_pfn *Predict using TabPFN*

Description

Predict using TabPFN

Usage

```
## S3 method for class 'tab_pfn'
predict(object, new_data, type = NULL, ...)
```

```
## S3 method for class 'tab_pfn'
augment(x, new_data, type = NULL, ...)
```

Arguments

object, x	A tab_pfn object.
new_data	A data frame or matrix of new predictors.
type	The type of prediction. For classification, can be "class" or "prob". Defaults to NULL which gives all prediction types possible.
...	Not used, but required for extensibility.

Value

`predict()` returns a tibble of predictions and `augment()` appends the columns in `new_data`. In either case, the number of rows in the tibble is guaranteed to be the same as the number of rows in `new_data`.

For regression data, the prediction is in the column `.pred`. For classification, the class predictions are in `.pred_class` and the probability estimates are in columns with the pattern `.pred_{level}` where `level` is the levels of the outcome factor vector.

Examples

```
# Minimal example for quick execution
car_train <- mtcars[ 1:5, ]
car_test  <- mtcars[6, -1]

## Not run:
# Fit
if (is_tab_pfn_installed() & interactive()) {
  mod <- tab_pfn(mpg ~ cyl + log(drat), car_train)

  # Predict
  predict(mod, car_test)
  augment(mod, car_test)
}
```

```
## End(Not run)
```

tab_pfn	<i>Fit a TabPFN model.</i>
---------	----------------------------

Description

tab_pfn() applies data to a pre-estimated deep learning model defined by Hollmann *et al* (2025). This model emulates Bayesian inference for regression and classification models.

Usage

```
tab_pfn(x, ...)  
  
## Default S3 method:  
tab_pfn(x, ...)  
  
## S3 method for class 'data.frame'  
tab_pfn(  
  x,  
  y,  
  num_estimators = 8L,  
  softmax_temperature = 0.9,  
  balance_probabilities = FALSE,  
  average_before_softmax = FALSE,  
  training_set_limit = 10000,  
  version = NULL,  
  control = control_tab_pfn(),  
  ...  
)  
  
## S3 method for class 'matrix'  
tab_pfn(  
  x,  
  y,  
  num_estimators = 8L,  
  softmax_temperature = 0.9,  
  balance_probabilities = FALSE,  
  average_before_softmax = FALSE,  
  training_set_limit = 10000,  
  version = NULL,  
  control = control_tab_pfn(),  
  ...  
)
```

```

## S3 method for class 'formula'
tab_pfn(
  formula,
  data,
  num_estimators = 8L,
  softmax_temperature = 0.9,
  balance_probabilities = FALSE,
  average_before_softmax = FALSE,
  training_set_limit = 10000,
  version = NULL,
  control = control_tab_pfn(),
  ...
)

## S3 method for class 'recipe'
tab_pfn(
  x,
  data,
  num_estimators = 8L,
  softmax_temperature = 0.9,
  balance_probabilities = FALSE,
  average_before_softmax = FALSE,
  training_set_limit = 10000,
  version = NULL,
  control = control_tab_pfn(),
  ...
)

```

Arguments

<code>x</code>	Depending on the context: <ul style="list-style-type: none"> • A data frame of predictors. • A matrix of predictors. • A recipe specifying a set of preprocessing steps created from <code>recipes::recipe()</code>.
<code>...</code>	Not currently used, but required for extensibility.
<code>y</code>	When <code>x</code> is a data frame or matrix , <code>y</code> is the outcome specified as: <ul style="list-style-type: none"> • A data frame with 1 numeric column. • A matrix with 1 numeric column. • A numeric vector for regression or a factor for classification.
<code>num_estimators</code>	An integer for the ensemble size. Default is 8L.
<code>softmax_temperature</code>	An adjustment factor that is a divisor in the exponents of the softmax function (see Details below). Defaults to 0.9.
<code>balance_probabilities</code>	A logical to adjust the prior probabilities in cases where there is a class imbalance. Default is FALSE. Classification only.

average_before_softmax	A logical. For cases where <code>num_estimators > 1</code> , should the average be done before using the softmax function or after? Default is <code>FALSE</code> .
training_set_limit	An integer greater than <code>2L</code> (and possibly <code>Inf</code>) that can be used to keep the training data within the limits of the data constraints imposed by the Python library.
version	A character string for the model version (e.g., "v2", "v2.5"). When <code>NULL</code> (the default), the Python library's current default version is used. When set, the model is initialized via <code>create_default_for_version()</code> with the corresponding <code>ModelVersion</code> enum value.
control	A list of options produced by <code>control_tab_pfn()</code> .
formula	A formula specifying the outcome terms on the left-hand side, and the predictor terms on the right-hand side.
data	When a recipe or formula is used, data is specified as: <ul style="list-style-type: none"> • A data frame containing both the predictors and the outcome.

Details

Computing Requirements:

This model can be used with or without a graphics processing unit (GPU). However, it is fairly limited when used with a CPU (and no GPU). There might be additional data size limitation warnings with CPU computations, and, understandably, the execution time is much longer. CPU computations can also consume a significant amount of system memory, depending on the size of your data.

GPUs using CUDA (Compute Unified Device Architecture) are most effective. Limited testing with others has shown that GPUs with Metal Performance Shaders (MPS) instructions (e.g., Apple GPUs) have limited utility for these specific computations and might be slower than the CPU for some data sets.

License Requirements:

Starting with version 2.5, using TabPFN requires accepting the model license and obtaining a token from PriorLabs. Each model version (v2.5, v2.6, etc.) has its own license that must be accepted individually.

To set up access:

1. Visit <https://ux.priorlabs.ai> and create an account.
2. Go to the **License** tab and accept the license for each model version you intend to use.
3. Obtain your token from your account page.
4. Set the `TABPFN_TOKEN` environment variable. The easiest way is to add it to your `.Renviron` file:

```
TABPFN_TOKEN=your_token_value
```

The `usethis` function `edit_renviron()` can be very helpful here.

Users who already have `TABPFN_TOKEN` set can use TabPFN v2 without any additional steps.

Python Installation:

You will need a working Python virtual environment with the correct packages to use these modeling functions.

There are at least two ways to proceed.

Ephemeral uv Install:

The first approach, which we *strongly suggest*, is to simply load this package and attempt to run a model. This will prompt **reticulate** to create an ephemeral environment and automatically install the required packages. That process would look like this:

```
> library(tabpfn)
>
> predictors <- mtcars[, -1]
> outcome <- mtcars[, 1]
>
> # XY interface
> mod <- tab_pfn(predictors, outcome)
Downloading uv...Done!
Downloading cpython-3.12.12 (download) (15.9MiB)
  Downloading cpython-3.12.12 (download)
Downloading setuptools (1.1MiB)
Downloading scikit-learn (8.2MiB)
Downloading numpy (4.9MiB)

<downloading and installing more packages>

  Downloading llvmlite
  Downloading torch
Installed 58 packages in 350ms
> mod
TabPFN Regression Model

Training set
i 32 data points
i 10 predictors
```

The location of the environment can be found at `tools::R_user_dir("reticulate", "cache")`. See the documentation for `reticulate::py_require()` to learn more about this method.

Manually created venv Virtual Environment:

Alternatively, you can use the functions in the **reticulate** package to create a virtual environment and install the required Python packages there. An example pattern is:

```
library(reticulate)

venv_name <- "r-tabpfn"    # exact name can be different
venv_seed_python <-
  virtualenv_starter(">=3.11,<3.14")

virtualenv_create(
  envname = venv_name,
  python = venv_seed_python,
```

```
packages = c("numpy", "tabpfn")
)
```

Once you have that virtual environment installed, you can declare it as your preferred Python installation with `use_virtualenv()`. (You must do this before `reticulate` has initialized Python, i.e., before attempting to use **tabpfn**):

```
reticulate::use_virtualenv("r-tabpfn")
```

Data:

By default, there are limits to the training data dimensions:

- Version 2.0: number of training set samples (10,000) and, the number of predictors (500). There is an unchangeable limit to the number of classes (10).
- Version 2.5: number of training set samples (50,000) and, the number of predictors (2,000). There is an unchangeable limit to the number of classes (10).

Predictors do not require preprocessing; missing values and factor vectors are allowed.

Model Selection:

By default, TabPFN uses the Python library's current default model version. There are two ways to override this.

Selecting a model version:

Use the `version` argument to select a specific released model version. For example:

```
# Use version 2.0
mod <- tab_pfn(predictors, outcome, version = "v2")

# Use version 2.5
mod <- tab_pfn(predictors, outcome, version = "v2.5")
```

Pointing to a local model file:

If you have a model file on disk (e.g., downloaded for offline use), pass its path via `control_tab_pfn(model_path = ...)`:

```
ctrl <- control_tab_pfn(model_path = "/path/to/model_file.ckpt")
mod <- tab_pfn(predictors, outcome, control = ctrl)
```

Note that `version` and `model_path` are mutually exclusive: if `version` is set, it overwrites any `model_path` supplied through `control`.

Calculations:

For the `softmax_temperature` value, the softmax terms are:

```
exp(value / softmax_temperature)
```

A value of `softmax_temperature = 1` results in a plain softmax value.

Value

A `tab_pfn` object with elements:

- `fit`: the python object containing the model.
- `levels`: a character string of class levels (or `NULL` for regression)
- `training`: a vector with the training set dimensions.

- logging: any R or python messages produced by the computations.
- blueprint: an object produced by `hardhat::mold()` used to process new data during prediction.

References

Hollmann, Noah, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeyer, and Frank Hutter. "Accurate predictions on small data with a tabular foundation model." *Nature* 637, no. 8045 (2025): 319-326.

Hollmann, Noah, Samuel Müller, Katharina Eggensperger, and Frank Hutter. "Tabpfn: A transformer that solves small tabular classification problems in a second." *arXiv preprint arXiv:2207.01848* (2022).

Müller, Samuel, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. "Transformers can do Bayesian inference." *arXiv preprint arXiv:2112.10510* (2021).

See Also

[control_tab_pfn\(\)](#), [predict.tab_pfn\(\)](#)

Examples

```
predictors <- mtcars[, -1]
outcome <- mtcars[, 1]

## Not run:
if (is_tab_pfn_installed() & interactive()) {
  # XY interface
  mod <- tab_pfn(predictors, outcome)

  # Formula interface
  mod2 <- tab_pfn(mpg ~ ., mtcars)

  # Recipes interface
  if (rlang::is_installed("recipes")) {
    suppressPackageStartupMessages(library(recipes))
    rec <-
      recipe(mpg ~ ., mtcars) %>%
        step_log(dispatch)

    mod3 <- tab_pfn(rec, mtcars)
    mod3
  }
}

## End(Not run)
```

`tabpfn_download_models`*Download all TabPFN pre-trained model checkpoints*

Description

As of 2026-05-05, there are 36 pre-trained models equaling roughly 1.2 GB of storage. Each model is trained on various synthetic & real datasets tailored to classification & regression. This function routine will require you to sign a one-time license for both 2.5 & 2.6 model varieties. Downloading all models will take some time.

Usage

```
tabpfn_download_models(cache_dir = NULL)
```

Arguments

`cache_dir` an option to override the default cache directory

Value

Invisibly returns NULL. Called for its side effect of downloading model files.

Examples

```
tabpfn_download_models()
```

`tabpfn_list_versions` *List available TabPFN model versions*

Description

Returns a character vector of valid model version strings accepted by `tab_pfn()`'s version argument. The available model versions are queried directly from the currently installed Python tabpfn library, not hard-coded in this package, so results may differ across Python library versions.

Usage

```
tabpfn_list_versions()
```

Value

A character vector of model version strings.

Index

`augment()`, [4](#)
`augment.tab_pfn` (`predict.tab_pfn`), [4](#)

`control_tab_pfn`, [2](#)
`control_tab_pfn()`, [7](#), [10](#)

`hardhat::mold()`, [10](#)

`is_tab_pfn_installed`, [3](#)

`predict()`, [4](#)
`predict.tab_pfn`, [4](#)
`predict.tab_pfn()`, [10](#)

`recipes::recipe()`, [6](#)
`reticulate::py_require()`, [8](#)

`tab_pfn`, [5](#)
`tab_pfn()`, [11](#)
`tabpfn_download_models`, [11](#)
`tabpfn_list_versions`, [11](#)