Package: IBLM (via r-universe)

October 20, 2026

Type Package

Title Interpretable Boosted Linear Models

Version 2.0.1

Description Implements Interpretable Boosted Linear Models (IBLMs). These combine a conventional generalized linear model (GLM) with a machine learning component, such as XGBoost. The package also provides tools within for explaining and analyzing these models. For more details see Gawlowski and Wang (2025) <https://ifoa-adswp.github.io/IBLM/reference/figures/iblm_paper.pdf>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0)

RoxygenNote 7.3.0

Imports cli, dplyr, fastDummies, ggExtra, ggplot2, purrr, scales, statmod, stats, utils, withr, xgboost (>= 3.1.2.1)

Suggests knitr, rmarkdown, testthat (>= 3.0.0), gt, patchwork

VignetteBuilder knitr

Config/testthat/edition 3

URL https://ifoa-adswp.github.io/IBLM/, https://github.com/IFoA-ADSWP/IBLM

BugReports https://github.com/IFoA-ADSWP/IBLM/issues

Config/pak/sysreqs cmake make libicu-dev libuv1-dev zlib1g-dev

Repository https://ifoa-adswp.r-universe.dev

Date/Publication 2026-05-23 17:36:34 UTC

RemoteUrl https://github.com/ifoa-adswp/iblm

RemoteRef HEAD

RemoteSha d495d193fe8df815c5c99570509a1f5d95d0dba2

Contents

beta_corrected_density . . . 2
beta_corrected_scatter . . . 4
beta_corrections_derive . . . 5
bias_density . . . 7
check_iblm_model . . . 8
correction_corridor . . . 9
create_beta_corrected_density . . . 10
create_beta_corrected_scatter . . . 11
create_bias_density . . . 12
create_overall_correction . . . 14
data_beta_coeff_booster . . . 15
data_beta_coeff_glm . . . 16
data_to_onehot . . . 17
explain_iblm . . . 18
extract_booster_shap . . . 19
freMTPLmini . . . 20
get_pinball_scores . . . 21
load_freMTPL2freq . . . 22
overall_correction . . . 24
predict.iblm . . . 25
shap_to_onehot . . . 26
split_into_train_validate_test . . . 27
theme_iblm . . . 28
train_iblm_xgb . . . 29
train_xgb_as_per_iblm . . . 31

Index **33**

`beta_corrected_density`

Density Plot of Beta Corrections for a Variable

Description

Generates a density plot showing the distribution of corrected Beta values to a GLM coefficient, along with the original Beta coefficient, and standard error bounds around it.

NOTE This function signature documents the interface of functions created by `create_beta_corrected_density`.

Usage

```
beta_corrected_density(varname, q = 0.05, type = "kde")
```

Arguments

varname	Character string specifying the variable name OR coefficient name is accepted as well.
q	Number, must be between 0 and 0.5. Determines the quantile range of the plot (i.e. value of 0.05 will only show shaps within 5pct –> 95pct quantile range for plot)
type	Character string, must be "kde" or "hist"

Details

The plot shows:

- Density curve of corrected coefficient values
- Solid vertical line at the original GLM coefficient
- Dashed lines at plus/minus 1 standard error from the coefficient
- Automatic x-axis limits that cut off the highest and lowest q pct. If you want axis unaltered, set q = 0

Value

ggplot object(s) showing the density distribution of corrected beta coefficients with vertical lines indicating the original coefficient value and standard error bounds.

The item returned will be:

- single ggplot object when ‘varname‘ was a numerical variable OR a coefficient name
- list of ggplot objects when ‘varname‘ was a categorical variable

See Also

create_beta_corrected_density, explain_iblm

Examples

```
# This function is created inside explain_iblm() and is output as an item

df_list <- freMTPLmini |>
  dplyr::mutate(LogExposure = log(Exposure), .keep = "unused") |>
  split_into_train_validate_test(seed = 9000)

iblm_model <- train_iblm_xgb(
  df_list,
  response_var = "ClaimNb",
  offset_var = "LogExposure",
  family = "poisson"
)

explain_objects <- explain_iblm(iblm_model, df_list$test)

# plot can be for a single categorical level OR a categorical variable
```

```
explain_objects$beta_corrected_density(varname = "AreaB")

# output can be numerical variable
explain_objects$beta_corrected_density(varname = "DrivAge")


# This function must be created, and cannot be called directly from the package
try(
beta_corrected_density(varname = "DrivAge")
)
```

`beta_corrected_scatter`

Scatter Plot of Beta Corrections for a Variable

Description

Generates a scatter plot or boxplot showing SHAP corrections for a specified variable from a fitted model. For numerical variables, creates a scatter plot with optional coloring and marginal densities. For categorical variables, creates a boxplot with model coefficients overlaid.

NOTE This function signature documents the interface of functions created by `create_beta_corrected_scatter`.

Usage

```
beta_corrected_scatter(varname, q = 0, color = NULL, marginal = FALSE)
```

Arguments

`varname`	Character. Name of the variable to plot SHAP corrections for. Must be present in the fitted model.
`q`	Numeric. Quantile threshold for outlier removal. When 0 (default) the function will not remove any outliers
`color`	Character or NULL. Name of variable to use for point coloring. Must be present in the model. Currently not supported for categorical variables.
`marginal`	Logical. Whether to add marginal density plots (numerical variables only).

Details

The function handles both numerical and categorical variables differently:

- Numerical: Creates scatter plot of variable values vs. beta + SHAP deviations
- Categorical: Creates boxplot of SHAP deviations for each level with coefficient overlay

For numerical variables, horizontal lines show the model coefficient (solid) and confidence intervals (dashed). SHAP corrections represent local deviations from the global model coefficient.

Value

A ggplot2 object. For numerical variables: scatter plot with SHAP corrections, model coefficient line, and confidence bands. For categorical variables: boxplot with coefficient points overlaid.

See Also

create_beta_corrected_scatter, explain_iblm

Examples

```
# This function is created inside explain_iblm() and is output as an item

df_list <- freMTPLmini |>
  dplyr::mutate(LogExposure = log(Exposure), .keep = "unused") |>
  split_into_train_validate_test(seed = 9000)

iblm_model <- train_iblm_xgb(
  df_list,
  response_var = "ClaimNb",
  offset_var = "LogExposure",
  family = "poisson"
)

explain_objects <- explain_iblm(iblm_model, df_list$test)

# plot can be for a categorical variable (box plot)
explain_objects$beta_corrected_scatter(varname = "Area")

# plot can be for a numerical variable (scatter plot)
explain_objects$beta_corrected_scatter(varname = "DrivAge")


# This function must be created, and cannot be called directly from the package
try(
beta_corrected_scatter(varname = "DrivAge")
)
```

---

beta_corrections_derive

Compute Beta Corrections based on SHAP values

Description

Processes SHAP values in one-hot (wide) format to create beta coefficient corrections.

This includes:

- scaling shap values of continuous variables by the predictor value for that row
- migrating shap values to the bias for continuous variables where the predictor value was zero
- migrating shap values to the bias for categorical variables where the predictor value was reference level

Usage

```
beta_corrections_derive(
  shap_wide,
  wide_input_frame,
  iblm_model,
  migrate_reference_to_bias = TRUE
)
```

Arguments

shap_wide	Data frame containing SHAP values from XGBoost that have been converted to wide format by [shap_to_onehot()]
wide_input_frame	Wide format input data frame (one-hot encoded).
iblm_model	Object of class 'iblm'
migrate_reference_to_bias	Logical, migrate the beta corrections to the bias for reference levels? This applied to categorical vars only. It is recommended to leave this setting on TRUE

Value

A data frame with the booster model beta corrections in one-hot (wide) format

Examples

```
df_list <- freMTPLmini |>
  dplyr::mutate(LogExposure = log(Exposure), .keep = "unused") |>
  split_into_train_validate_test(seed = 9000)

iblm_model <- train_iblm_xgb(
  df_list,
  response_var = "ClaimNb",
  offset_var = "LogExposure",
  family = "poisson"
)

shap <- extract_booster_shap(iblm_model$booster_model, df_list$test)

wide_input_frame <- data_to_onehot(df_list$test, iblm_model)

shap_wide <- shap_to_onehot(shap, wide_input_frame, iblm_model)

beta_corrections <- beta_corrections_derive(shap_wide, wide_input_frame, iblm_model)

beta_corrections |> dplyr::glimpse()
```

bias_density *Density Plot of Bias Corrections from SHAP values*

Description

Visualizes the distribution of SHAP corrections that are migrated to bias terms, showing both per-variable and total bias corrections.

NOTE This function signature documents the interface of functions created by `create_bias_density`.

Usage

```
bias_density(q = 0, type = "hist")
```

Arguments

q	Numeric value between 0 and 0.5 for quantile bounds. A higher number will trim more from the edges (useful if outliers are distorting your plot window) Default is 0 (i.e. no trimming)
type	Character string specifying plot type: "kde" for kernel density or "hist" for histogram. Default is "hist".

Value

A list with two ggplot objects:

- `bias_correction_var`: Faceted plot showing bias correction density from each variable. Note that variables with no records contributing to bias correction are dropped from the plot.
- `bias_correction_total`: Plot showing total corrected bias density.

See Also

`create_bias_density`, `explain_iblm`

Examples

```
# This function is created inside explain_iblm() and is output as an item

df_list <- freMTPLmini |>
  dplyr::mutate(LogExposure = log(Exposure), .keep = "unused") |>
  split_into_train_validate_test(seed = 9000)

iblm_model <- train_iblm_xgb(
  df_list,
  response_var = "ClaimNb",
  offset_var = "LogExposure",
  family = "poisson"
)
```

```
explain_objects <- explain_iblm(iblm_model, df_list$test)

explain_objects$bias_density()


# This function must be created, and cannot be called directly from the package
try(
bias_density()
)
```

---

check_iblm_model *Check Object of Class 'iblm'*

Description

Validates an iblm model object has required structure and features

Usage

```
check_iblm_model(model, booster_models_supported = c("xgb.Booster"))
```

Arguments

model	Model object to validate, expected class "iblm"
booster_models_supported	Booster model classes currently supported in the iblm package

Value

Invisible TRUE if all checks pass

Examples

```
df_list <- freMTPLmini |>
  dplyr::mutate(LogExposure = log(Exposure), .keep = "unused") |>
  split_into_train_validate_test(seed = 9000)

iblm_model <- train_iblm_xgb(
  df_list,
  response_var = "ClaimNb",
  offset_var = "LogExposure",
  family = "poisson"
)

check_iblm_model(iblm_model)
```

correction_corridor *Plot GLM vs IBLM Predictions with Different Corridors*

Description

Creates a faceted scatter plot comparing GLM predictions to ensemble predictions across different trim values, showing how the ensemble corrects the base GLM model.

Usage

```
correction_corridor(
  iblm_model,
  data,
  trim_vals = c(NA_real_, 4, 1, 0.2, 0.1, 0),
  sample_perc = 0.2,
  color = NA,
  ...
)
```

Arguments

iblm_model	An IBLM model object of class "iblm".
data	Data frame. If you have used ‘split_into_train_validate_test()‘ this will usually be the "test" portion of your data.
trim_vals	Numeric vector of trim values to compare. The length of this vector will dictate the no. of facets shown in plot output
sample_perc	Proportion of data to randomly sample for plotting (0 to 1). Default is 0.2 to improve performance with large datasets
color	Optional. Name of a variable in ‘data‘ to color points by
...	Additional arguments passed to ‘geom_point()‘

Value

A ggplot object showing GLM vs IBLM predictions faceted by trim value. The diagonal line (y = x) represents perfect agreement between models

Examples

```
df_list <- freMTPLmini |>
  dplyr::mutate(LogExposure = log(Exposure), .keep = "unused") |>
  split_into_train_validate_test(seed = 9000)

iblm_model <- train_iblm_xgb(
  df_list,
  response_var = "ClaimNb",
  offset_var = "LogExposure",
```

```
  family = "poisson"
)

correction_corridor(iblm_model = iblm_model, data = df_list$test, color = "DrivAge")
```

`create_beta_corrected_density` *Create Pre-Configured Beta Corrected Density Plot Function*

Description

Factory function that returns a plotting function with data pre-configured.

Usage

```
create_beta_corrected_density(
  wide_input_frame,
  beta_corrections,
  data,
  iblm_model
)
```

Arguments

`wide_input_frame`	Dataframe. Wide format input data (one-hot encoded).
`beta_corrections`	Dataframe. Output from `beta_corrections_derive`.
`data`	Dataframe. The testing data.
`iblm_model`	Object of class 'iblm'.

Value

Function with signature `function(varname, q = 0.05, type = "kde")`.

See Also

[beta_corrected_density()]

Examples

```
# ------- prepare iblm objects required -------

df_list <- freMTPLmini |>
  dplyr::mutate(LogExposure = log(Exposure), .keep = "unused") |>
  split_into_train_validate_test(seed = 9000)
```

```
iblm_model <- train_iblm_xgb(
  df_list,
  response_var = "ClaimNb",
  offset_var = "LogExposure",
  family = "poisson"
)

test_data <- df_list$test
shap <- extract_booster_shap(iblm_model$booster_model, test_data)
wide_input_frame <- data_to_onehot(test_data, iblm_model)
shap_wide <- shap_to_onehot(shap, wide_input_frame, iblm_model)
beta_corrections <- beta_corrections_derive(shap_wide, wide_input_frame, iblm_model)

# ------- demonstration of functionality -------

# create_beta_corrected_density() can create function of type 'beta_corrected_density'
my_beta_corrected_density <- create_beta_corrected_density(
  wide_input_frame,
  beta_corrections,
  test_data,
  iblm_model
)

# this custom function then acts as per beta_corrected_density()
my_beta_corrected_density(varname = "VehAge")
```

`create_beta_corrected_scatter` *Create Pre-Configured Beta Corrected Scatter Plot Function*

Description

Factory function that returns a plotting function with data pre-configured.

Usage

```
create_beta_corrected_scatter(data_beta_coeff, data, iblm_model)
```

Arguments

`data_beta_coeff`	Dataframe. Contains the corrected beta coefficients for each row of the data.
`data`	Dataframe. The testing data.
`iblm_model`	Object of class 'iblm'.

Value

Function with signature `function(varname, q = 0, color = NULL, marginal = FALSE)`.

See Also

[beta_corrected_scatter()]

Examples

```
# ------- prepare iblm objects required -------

df_list <- freMTPLmini |>
  dplyr::mutate(LogExposure = log(Exposure), .keep = "unused") |>
  split_into_train_validate_test(seed = 9000)

iblm_model <- train_iblm_xgb(
  df_list,
  response_var = "ClaimNb",
  offset_var = "LogExposure",
  family = "poisson"
)

test_data <- df_list$test
shap <- extract_booster_shap(iblm_model$booster_model, test_data)
wide_input_frame <- data_to_onehot(test_data, iblm_model)
shap_wide <- shap_to_onehot(shap, wide_input_frame, iblm_model)
beta_corrections <- beta_corrections_derive(shap_wide, wide_input_frame, iblm_model)
data_glm <- data_beta_coeff_glm(test_data, iblm_model)
data_booster <- data_beta_coeff_booster(test_data, beta_corrections, iblm_model)
data_beta_coeff <- data_glm + data_booster

# ------- demonstration of functionality -------

# create_beta_corrected_scatter() can create function of type 'beta_corrected_scatter'
my_beta_corrected_scatter <- create_beta_corrected_scatter(data_beta_coeff, test_data, iblm_model)

# this custom function then acts as per beta_corrected_scatter()
my_beta_corrected_scatter(varname = "VehAge")
```

---

create_bias_density *Create Pre-Configured Bias Density Plot Function*

Description

Factory function that returns a plotting function with data pre-configured.

Usage

```
create_bias_density(shap, data, iblm_model, migrate_reference_to_bias = TRUE)
```

Arguments

shap	Dataframe. Contains raw SHAP values.
data	Dataframe. The testing data.
iblm_model	Object of class 'iblm'.
migrate_reference_to_bias	TRUE/FALSE determines whether the shap values of categorical reference levels be migrated to the bias? Default is TRUE

Value

Function with signature `function(q = 0, type = "hist")`.

See Also

[bias_density()]

Examples

```
# ------- prepare iblm objects required -------

df_list <- freMTPLmini |>
  dplyr::mutate(LogExposure = log(Exposure), .keep = "unused") |>
  split_into_train_validate_test(seed = 9000)

iblm_model <- train_iblm_xgb(
  df_list,
  response_var = "ClaimNb",
  offset_var = "LogExposure",
  family = "poisson"
)

test_data <- df_list$test
shap <- extract_booster_shap(iblm_model$booster_model, test_data)

# ------- demonstration of functionality -------

# create_bias_density() can create function of type 'bias_density'
my_bias_density <- create_bias_density(shap, test_data, iblm_model)

# this custom function then acts as per bias_density()
my_bias_density()
```

`create_overall_correction` *Create Pre-Configured Overall Correction Plot Function*

Description

Factory function that returns a plotting function with data pre-configured.

Usage

```
create_overall_correction(shap, iblm_model)
```

Arguments

`shap`	Dataframe. Contains raw SHAP values.
`iblm_model`	Object of class 'iblm'.

Value

Function with signature `function(transform_x_scale_by_link = TRUE)`.

See Also

[overall_correction()]

Examples

```
# ------- prepare iblm objects required -------

df_list <- freMTPLmini |>
  dplyr::mutate(LogExposure = log(Exposure), .keep = "unused") |>
  split_into_train_validate_test(seed = 9000)

iblm_model <- train_iblm_xgb(
  df_list,
  response_var = "ClaimNb",
  offset_var = "LogExposure",
  family = "poisson"
)

test_data <- df_list$test
shap <- extract_booster_shap(iblm_model$booster_model, test_data)

# ------- demonstration of functionality -------

# create_overall_correction() can create function of type 'overall_correction'
my_overall_correction <- create_overall_correction(shap, iblm_model)

# this custom function then acts as per overall_correction()
```

```
my_overall_correction()
```

`data_beta_coeff_booster` *Obtain Booster Model Beta Corrections for tabular data*

Description

Creates dataframe of Shap beta corrections for each row and predictor variable of ‘data‘

Usage

```
data_beta_coeff_booster(data, beta_corrections, iblm_model)
```

Arguments

`data`	A data frame containing the dataset for analysis
`beta_corrections`	A data frame or matrix containing beta correction values for all variables and bias
`iblm_model`	Object of class 'iblm'

Value

A data frame with beta coefficient corrections. The structure will be the same dimension as ‘data‘ except for a "bias" column at the start.

Examples

```
df_list <- freMTPLmini |>
  dplyr::mutate(LogExposure = log(Exposure), .keep = "unused") |>
  split_into_train_validate_test(seed = 9000)

iblm_model <- train_iblm_xgb(
  df_list,
  response_var = "ClaimNb",
  offset_var = "LogExposure",
  family = "poisson"
)

explainer_outputs <- explain_iblm(iblm_model, df_list$test)

data_booster <- data_beta_coeff_booster(
  df_list$test,
  explainer_outputs$beta_corrections,
  iblm_model
)
```

```
data_booster |> dplyr::glimpse()
```

`data_beta_coeff_glm` *Obtain GLM Beta Coefficients for tabular data*

Description

Creates dataframe of GLM beta coefficients for each row and predictor variable of ‘data‘

Usage

```
data_beta_coeff_glm(data, iblm_model)
```

Arguments

`data`	Data frame with predictor variables
`iblm_model`	Object of class 'iblm'

Value

A data frame with beta coefficients. The structure will be the same dimension as ‘data‘ except for a "bias" column at the start.

Examples

```
df_list <- freMTPLmini |>
  dplyr::mutate(LogExposure = log(Exposure), .keep = "unused") |>
  split_into_train_validate_test(seed = 9000)

iblm_model <- train_iblm_xgb(
  df_list,
  response_var = "ClaimNb",
  offset_var = "LogExposure",
  family = "poisson"
)

data_glm <- data_beta_coeff_glm(df_list$train, iblm_model)

data_glm |> dplyr::glimpse()
```

`data_to_onehot` *Convert Data Frame to Wide One-Hot Encoded Format*

Description

Transforms categorical variables in a data frame into one-hot encoded format

Usage

```
data_to_onehot(data, iblm_model, remove_target = TRUE)
```

Arguments

`data`	Input data frame to be transformed. This will typically be the "train" data subset
`iblm_model`	Object of class 'iblm'
`remove_target`	Logical, whether to remove the response_var variable from the output (default TRUE).

Value

A data frame in wide format with one-hot encoded categorical variables, an intercept column, and all variables ordered according to "coeff_names$all" from 'iblm_model'

Examples

```
df_list <- freMTPLmini |>
  dplyr::mutate(LogExposure = log(Exposure), .keep = "unused") |>
  split_into_train_validate_test(seed = 9000)

iblm_model <- train_iblm_xgb(
  df_list,
  response_var = "ClaimNb",
  offset_var = "LogExposure",
  family = "poisson"
)

wide_input_frame <- data_to_onehot(df_list$test, iblm_model)

wide_input_frame |> dplyr::glimpse()
```

`explain_iblm` *Explain GLM Model Predictions Using SHAP Values*

Description

Creates a list that explains the beta values, and their corrections, of the ensemble IBLM model

Usage

```
explain_iblm(iblm_model, data, migrate_reference_to_bias = TRUE)
```

Arguments

`iblm_model`	An object of class 'iblm'. This should be output by 'train_iblm_xgb()'
`data`	Data frame. If you have used 'split_into_train_validate_test()' this will be the "test" portion of your data.
`migrate_reference_to_bias`	Logical, migrate the beta corrections to the bias for reference levels? This applied to categorical vars only. It is recommended to leave this setting on TRUE

Details

The following outputs are functions that can be called to create plots:

- beta_corrected_scatter
- beta_corrected_density
- bias_density
- overall_correction

For each of these, the key data arguments (e.g. data, shap, iblm_model) are already populated by 'explain_iblm()'. When calling these functions output from 'explain_iblm()' only key settings like variable names, colours...etc need populating.

Value

A list containing:

beta_corrected_scatter Function to create scatter plots showing SHAP corrections vs variable values (see `beta_corrected_scatter`)

beta_corrected_density Function to create density plots of SHAP corrections for variables (see `beta_corrected_density`)

bias_density Function to create density plots of SHAP corrections migrated to bias (see `bias_density`)

overall_correction Function to show global correction distributions (see `overall_correction`)

shap Dataframe showing raw SHAP values of data records

beta_corrections Dataframe showing beta corrections (in wide/one-hot format) of data records

data_beta_coeff Dataframe showing beta coefficients of data records

Examples

```
df_list <- freMTPLmini |>
  dplyr::mutate(LogExposure = log(Exposure), .keep = "unused") |>
  split_into_train_validate_test(seed = 9000)

iblm_model <- train_iblm_xgb(
  df_list,
  response_var = "ClaimNb",
  offset_var = "LogExposure",
  family = "poisson"
)

ex <- explain_iblm(iblm_model, df_list$test)

# the output contains functions that can be called to visualise iblm
ex$beta_corrected_scatter("DrivAge")
ex$beta_corrected_density("DrivAge")
ex$overall_correction()
ex$bias_density()

# the output contains also dataframes
ex$shap |> dplyr::glimpse()
ex$beta_corrections |> dplyr::glimpse()
ex$data_beta_coeff |> dplyr::glimpse()
```

---

extract_booster_shap *Extract SHAP values from an xgboost Booster model*

Description

A function to extract SHAP (SHapley Additive exPlanations) values from fitted booster model

Usage

```
extract_booster_shap(booster_model, data, ...)

## S3 method for class 'xgb.Booster'
extract_booster_shap(booster_model, data, ...)
```

Arguments

booster_model	A model object. In the IBLM context it will be the "booster_model" item from an object of class "iblm"
data	A data frame containing the predictor variables. Note anything extra will be quietly dropped.
...	Additional arguments passed to methods.

Details

Currently only a booster_model of class ‘xgb.Booster‘ is supported

Value

A data frame of SHAP values, where each column corresponds to a feature and each row corresponds to an observation.

Methods (by class)

- `extract_booster_shap(xgb.Booster)`: Extract SHAP values from an ‘xgb.Booster‘ model

See Also

[xgboost::predict.xgb.Booster()]

Examples

```
df_list <- freMTPLmini |>
  dplyr::mutate(LogExposure = log(Exposure), .keep = "unused") |>
  split_into_train_validate_test(seed = 9000)

iblm_model <- train_iblm_xgb(
  df_list,
  response_var = "ClaimNb",
  offset_var = "LogExposure",
  family = "poisson"
)

booster_shap <- extract_booster_shap(iblm_model$booster_model, df_list$test)

booster_shap |> dplyr::glimpse()
```

`freMTPLmini` *French Motor Insurance Claims Dataset*

Description

A dataset containing information about French motor insurance policies and claims, commonly used for actuarial modeling and risk assessment studies.

This is a "mini" subset of the CASdatasets ‘freMTPL2freq‘ data, with some manipulation (see details) so that it is ready to plug into the IBLM functions

Usage

```
freMTPLmini
```

Format

A data frame with 25,000 rows and 8 variables:

Area Area classification where the policy holder resides (factor with levels A through F)

BonusMalus Bonus-malus coefficient, a rating factor used in French insurance where lower values indicate better driving records (integer)

DrivAge Age of the driver in years (integer)

VehAge Age of the vehicle in years (integer)

VehBrand Vehicle brand/manufacturer code (factor with levels like B6, B12, etc.)

VehPower Vehicle power rating or engine horsepower category (integer)

ClaimNb Number of claims made, at an annualised rate (double)

Exposure Length of Exposure in years (double)

Details

The dataset is a sample of 25,000 records from 'freMTPL2freq' from the 'CASdatasets' package. Other modifications applied are:

- `ClaimRate`: Converted to ClaimNb per Exposure, winsorized at the 99.9th percentile, and rounded.
- `VehAge`: Ceiling of 50 years applied
- Dropped columns: VehGas, Region, Density, ClaimNb, IDpol

Source

['https://github.com/dutangc/CASdatasets/raw/c49cbbb37235fc49616cac8ccac32e1491cdc619/data/freMTPL2freq.rda']

Examples

```
head(freMTPLmini)
```

`get_pinball_scores` *Calculate Pinball Scores for IBLM and Additional Models*

Description

Computes Poisson deviance and pinball scores for an IBLM model alongside homogeneous, GLM, and optional additional models.

Usage

```
get_pinball_scores(
  data,
  iblm_model,
  trim = NA_real_,
  additional_models = list()
)
```

Arguments

data	Data frame. If you have used ‘split_into_train_validate_test()‘ this will be the "test" portion of your data.
iblm_model	Fitted IBLM model object of class "iblm"
trim	Numeric trimming parameter for IBLM predictions. Default is ‘NA_real_‘.
additional_models	(Named) list of fitted models for comparison. These models MUST be fitted on the same data as ‘iblm_model‘ for sensible results. If unnamed, models are labeled by their class.

Details

Pinball scores are calculated relative to a homogeneous model (i.e. a simple mean prediction of training data). Higher scores indicate better predictive performance.

Value

Data frame with 3 columns:

- "model" - will be homog, glm, iblm and any other models specified in ‘additional_models‘
- "‘family‘_deviance" - the value from the loss function based on the family of the glm function
- "pinball_score" - The more positive the score, the better the model than a basic homog model (i.e. all predictions are mean value). A negative score indicates worse than homog model.

Examples

```
df_list <- freMTPLmini |>
  dplyr::mutate(LogExposure = log(Exposure), .keep = "unused") |>
  split_into_train_validate_test(seed = 9000)

iblm_model <- train_iblm_xgb(
  df_list,
  response_var = "ClaimNb",
  offset_var = "LogExposure",
  family = "poisson"
)

get_pinball_scores(data = df_list$test, iblm_model = iblm_model)
```

---

load_freMTPL2freq *Load French Motor Third-Party Liability Frequency Dataset*

Description

Downloads the French Motor Third-Party Liability (freMTPL2freq) dataset from the CASdatasets GitHub repository, and apply minor transformations.

Usage

```
load_freMTPL2freq()
```

Details

The function performs the following modifications to the sourced:

- `ClaimNb`: Converted to ClaimNb per Exposure, and winsorized at the 99.9th percentile
- `VehAge`: Ceiling of 50 years applied
- Any character variables converted to factor type

Value

A data frame containing the following columns:

ClaimNb Number of claims per year.

VehPower The power of the car (ordered categorical).

VehAge The vehicle age in years, capped at 50.

DrivAge The driver age in years (minimum 18, the legal driving age in France).

BonusMalus Bonus/malus coefficient, ranging from 50 to 350. Values below 100 indicate a bonus (discount), while values above 100 indicate a malus (surcharge) in the French insurance system.

VehBrand The car brand (categorical, with unknown/proprietary category labels).

VehGas The car fuel type: "Diesel" or "Regular".

Area The density classification of the city community where the driver lives, ranging from "A" (rural area) to "F" (urban centre).

Density The population density (inhabitants per square kilometer) of the city where the driver lives.

Region The policy region in France, based on the 1970-2015 administrative classification.

Note

This function requires an internet connection to download the data.

References

Dutang, C. CASdatasets: Insurance datasets. https://github.com/dutangc/CASdatasets/raw/c49cbbb37235fc49616cac8ccac32e1491cdc619/data/freMTPL2freq.rda

Examples

```
# Load the preprocessed dataset
freMTPL2freq <- load_freMTPL2freq()

freMTPL2freq |> dplyr::glimpse()
```

`overall_correction` *Plot Overall Corrections from Booster Component*

Description

Creates a visualization showing for each record the overall booster component (either multiplicative or additive)

NOTE This function signature documents the interface of functions created by `create_overall_correction`.

Usage

```
overall_correction(transform_x_scale_by_link = TRUE)
```

Arguments

`transform_x_scale_by_link`	TRUE/FALSE, whether to transform the x axis by the link function

Value

A ggplot2 object.

See Also

`create_overall_correction`, `explain_iblm`

Examples

```
# This function is created inside explain_iblm() and is output as an item

df_list <- freMTPLmini |>
  dplyr::mutate(LogExposure = log(Exposure), .keep = "unused") |>
  split_into_train_validate_test(seed = 9000)

iblm_model <- train_iblm_xgb(
  df_list,
  response_var = "ClaimNb",
  offset_var = "LogExposure",
  family = "poisson"
)

explain_objects <- explain_iblm(iblm_model, df_list$test)

explain_objects$overall_correction()


# This function must be created, and cannot be called directly from the package
try(
overall_correction()
)
```

predict.iblm *Predict Method for IBLM*

Description

This function generates predictions from an ensemble model consisting of a GLM and an XGBoost model.

Usage

```
## S3 method for class 'iblm'
predict(object, newdata, trim = NA_real_, type = "response", ...)
```

Arguments

object	An object of class 'iblm'. This should be output by 'train_iblm_xgb()'
newdata	A data frame or matrix containing the predictor variables for which predictions are desired. Must have the same structure as the training data used to fit the 'iblm' model.
trim	Numeric value for post-hoc truncating of XGBoost predictions. If NA (default) then no trimming is applied.
type	string, defines the type argument used in GLM/Booster Currently only "response" is supported
...	additional arguments affecting the predictions produced.

Details

The prediction process involves the following steps:

1. Generate GLM predictions
2. Generate Booster predictions
3. If trimming is specified, apply to booster predictions
4. Combine GLM and Booster predictions as per "relationship" described within iblm model object

At this point, only an iblm model with a "booster_model" object of class 'xgb.Booster' is supported

Value

A numeric vector of ensemble predictions computed as the element-wise product of GLM response probabilities and (optionally trimmed) XGBoost predictions.

See Also

predict.glm, predict.xgb.Booster

Examples

```
df_list <- freMTPLmini |>
  dplyr::mutate(LogExposure = log(Exposure), .keep = "unused") |>
  split_into_train_validate_test(seed = 9000)

iblm_model <- train_iblm_xgb(
  df_list,
  response_var = "ClaimNb",
  offset_var = "LogExposure",
  family = "poisson"
)

predictions <- predict(iblm_model, df_list$test)

predictions |> dplyr::glimpse()
```

---

shap_to_onehot *Convert Shap values to Wide One-Hot Encoded Format*

Description

Transforms categorical variables in a data frame into one-hot encoded format. Renames "BIAS" to lowercase.

Usage

```
shap_to_onehot(shap, wide_input_frame, iblm_model)
```

Arguments

shap	Data frame containing raw SHAP values from XGBoost.
wide_input_frame	Wide format input data frame (one-hot encoded).
iblm_model	Object of class 'iblm'

Value

A data frame where SHAP values are in wide format for categorical variables. Column "bias" is moved to start.

Examples

```
df_list <- freMTPLmini |>
  dplyr::mutate(LogExposure = log(Exposure), .keep = "unused") |>
  split_into_train_validate_test(seed = 9000)

iblm_model <- train_iblm_xgb(
```

```
  df_list,
  response_var = "ClaimNb",
  offset_var = "LogExposure",
  family = "poisson"
)

shap <- extract_booster_shap(iblm_model$booster_model, df_list$test)

wide_input_frame <- data_to_onehot(df_list$test, iblm_model)

shap_wide <- shap_to_onehot(shap, wide_input_frame, iblm_model)

shap_wide |> dplyr::glimpse()
```

`split_into_train_validate_test` *Split Dataframe into: 'train', 'validate', 'test'*

Description

This function randomly splits a data frame into three subsets for machine learning workflows: training, validation, and test sets. The proportions can be customized and must sum to 1.

Usage

```
split_into_train_validate_test(
  df,
  train_prop = 0.7,
  validate_prop = 0.15,
  test_prop = 0.15,
  seed = NULL
)
```

Arguments

`df`	A data frame to be split into subsets.
`train_prop`	A numeric value between 0 and 1 specifying the proportion of data to allocate to the training set.
`validate_prop`	A numeric value between 0 and 1 specifying the proportion of data to allocate to the validation set.
`test_prop`	A numeric value between 0 and 1 specifying the proportion of data to allocate to the test set.
`seed`	(optional) a numeric value to set the random no. seed within function environment.

Details

The function assigns each row to either "train", "validate" or "test" with the probability defined in the function.

Because each row is assigned a bucket independently, for very small datasets the proportions may not be as desired. This should not be an issue as data used for ‘iblm‘ must be reasonably large.

Value

A named list with three elements:

train A data frame containing the training subset

validate A data frame containing the validation subset

test A data frame containing the test subset

Examples

```
# Using 'mtcars'
split_into_train_validate_test(
  mtcars,
  train_prop = 0.6,
  validate_prop = 0.2,
  test_prop = 0.2,
  seed = 9000
)
```

---

theme_iblm *Custom ggplot2 Theme for IBLM*

Description

Custom ggplot2 Theme for IBLM

Usage

```
theme_iblm()
```

Value

A ggplot2 theme object that can be added to plots.

`train_iblm_xgb` *Train IBLM Model on XGBoost*

Description

This function trains an interpretable boosted linear model.

The function combines a Generalized Linear Model (GLM) with a booster model of XGBoost

The "booster" model is trained on the residuals of the glm model to the response_var, such that: - when the link function is log, IBLM predictions = GLM predictions * Booster predictions - when the link function is identity, IBLM predictions = GLM predictions + Booster predictions

Usage

```
train_iblm_xgb(
  df_list,
  response_var,
  weight_var = NULL,
  offset_var = NULL,
  family = "poisson",
  params = list(),
  nrounds = 1000,
  objective = NULL,
  custom_metric = NULL,
  verbose = 0,
  print_every_n = 1L,
  early_stopping_rounds = 25,
  maximize = NULL,
  save_period = NULL,
  save_name = "xgboost.model",
  xgb_model = NULL,
  callbacks = list(),
  ...,
  strip_glm = TRUE
)
```

Arguments

`df_list`	A named list containing training and validation datasets. Must have elements named "train" and "validate", each containing df_list frames with the same structure. This item is naturally output from the function [split_into_train_validate_test()]
`response_var`	Character string specifying the name of the response variable column in the datasets. The string MUST appear in both ‘df_list$train‘ and ‘df_list$validate‘.
`weight_var`	Character string specifying the name of a variable to weight by. Value of NULL (default) for no weighting. Any string MUST appear in both ‘df_list$train‘ and ‘df_list$validate‘.

`offset_var` Character string specifying the name of a variable to use as offset. Value of NULL (default) for no offset. Any string MUST appear in both 'df_list$train' and 'df_list$validate'.
Any transformations required (e.g. log) must be performed BEFORE 'df_list' is fed into function.

`family` Character string specifying the distributional family for the model. Currently only "poisson", "quasipoisson", "gamma", "tweedie" and "gaussian" is fully supported. See details for how this impacts fitting.

`params` Named list of additional parameters to pass to xgb.train. Note that train_iblm_xgb will select "objective" and "base_score" for you depending on 'family' (see details section). However you may overwrite these (do so with caution)

`nrounds`, `objective`, `custom_metric`, `verbose`, `print_every_n`, `early_stopping_rounds`, `maximize`, `save_period`, `save_name`, `xgb_model`, `callbacks`, `...`
These are passed directly to xgb.train

`strip_glm` TRUE/FALSE, whether to strip superfluous data from the 'glm_model' object saved within 'iblm' class that is output. Only serves to reduce memory constraints.

Details

The 'family' argument will be fed into the GLM fitting. Default 'params' values for the XGBoost fitting are also selected based on family:

- For "poisson" family, the "objective" is set to "count:poisson"
- For "quasipoisson" family, the "objective" is set to "count:poisson"
- For "gamma" family, the "objective" is set to "reg:gamma"
- For "tweedie" family, the "objective" is set to "reg:tweedie". Also, "tweedie_variance_power = 1.5".
- For "gaussian" family, the "objective" is set to "reg:squarederror"

Note: Any xgboost configuration below will be overwritten by any explicit arguments input into 'train_iblm_xgb()'

Value

An object of class "iblm" containing:

`glm_model` The GLM model object, fitted on the 'df_list$train' data that was provided

`booster_model` The booster model object, trained on the residuals leftover from the glm_model

`data` A list containing the data that was used to train and validate this iblm model

`relationship` String that explains how to combine the 'glm_model' and 'booster_model'. Currently only either "Additive" or "Multiplicative"

`response_var` A string describing the response variable used for this iblm model

`predictor_vars` A list describing the predictor variables used for this iblm model

`cat_levels` A list describing the categorical levels for the predictor vars

`coeff_names` A list describing the coefficient names

See Also

glm, xgb.train

Examples

```
df_list <- freMTPLmini |>
  dplyr::mutate(LogExposure = log(Exposure), .keep = "unused") |>
  split_into_train_validate_test(seed = 9000)

iblm_model <- train_iblm_xgb(
  df_list,
  response_var = "ClaimNb",
  offset_var = "LogExposure",
  family = "poisson"
)
```

---

train_xgb_as_per_iblm *Train XGBoost Model Using the IBLM Model Parameters*

Description

Trains an XGBoost model using parameters extracted from the booster residual component of the iblm model. This is a convenient way to fit an XGBoost model for direct comparison with a fitted iblm model

Usage

```
train_xgb_as_per_iblm(iblm_model, ...)
```

Arguments

iblm_model	Ensemble model object of class "iblm" containing GLM and XGBoost model components. Also contains data that was used to train it.
...	optional arguments to insert into xgb.train(). Note this will cause deviation from the settings used for training ‘iblm_model‘

Value

Trained XGBoost model object (class "xgb.Booster").

See Also

xgb.train

Examples

```
df_list <- freMTPLmini |>
  dplyr::mutate(LogExposure = log(Exposure), .keep = "unused") |>
  split_into_train_validate_test(seed = 9000)

# training with plenty of rounds allowed
iblm_model1 <- train_iblm_xgb(
  df_list,
  response_var = "ClaimNb",
  offset_var = "LogExposure",
  family = "poisson",
  params = list(max_depth = 6),
  nrounds = 1000
)

xgb1 <- train_xgb_as_per_iblm(iblm_model1)

# training with severe restrictions (expected poorer results)
iblm_model2 <- train_iblm_xgb(
  df_list,
  response_var = "ClaimNb",
  offset_var = "LogExposure",
  family = "poisson",
  params = list(max_depth = 1),
  nrounds = 2
)

xgb2 <- train_xgb_as_per_iblm(iblm_model2)

# comparison shows the poor training mirrored in second set:
get_pinball_scores(
  df_list$test,
  iblm_model1,
  trim = NA_real_,
  additional_models = list(iblm2 = iblm_model2, xgb1 = xgb1, xgb2 = xgb2)
)
```

Index

∗ **actuarial**
freMTPLmini, 20
∗ **datasets**
freMTPLmini, 20
∗ **insurance**
freMTPLmini, 20

beta_corrected_density, 2, *18*
beta_corrected_scatter, 4, *18*
beta_corrections_derive, 5, *10*
bias_density, 7, *18*

check_iblm_model, 8
correction_corridor, 9
create_beta_corrected_density, *2, 3*, 10
create_beta_corrected_scatter, *4, 5*, 11
create_bias_density, *7*, 12
create_overall_correction, 14, *24*

data_beta_coeff_booster, 15
data_beta_coeff_glm, 16
data_to_onehot, 17

explain_iblm, *3*, *5*, *7*, 18, *24*
extract_booster_shap, 19

freMTPLmini, 20

get_pinball_scores, 21
glm, *31*

load_freMTPL2freq, 22

overall_correction, *18*, 24

predict.glm, *25*
predict.iblm, 25
predict.xgb.Booster, *25*

shap_to_onehot, 26
split_into_train_validate_test, 27

theme_iblm, 28
train_iblm_xgb, 29, *30*
train_xgb_as_per_iblm, 31

xgb.train, *30, 31*