Package 'modelimpact'

July 8, 2026

Type Package

Title Functions to Assess the Business Impact of Churn Prediction Models

Version 1.1.0

Description Calculate and visualise the financial impact of using a classification model, such as a churn model, to target customers. Provides cost, revenue, profit and return-on-investment curves as a function of the share of customers targeted, cumulative gains and lift, marginal profit per bin, and confusion-matrix based payoff across probability thresholds. Also includes 'ggplot2' 'autoplot()' methods and an interactive 'shiny' application for exploring the results.

License MIT + file LICENSE

URL https://github.com/PeerChristensen/modelimpact

BugReports https://github.com/PeerChristensen/modelimpact/issues

Encoding UTF-8

LazyData true

Imports dplyr, magrittr, utils

Suggests ggplot2, scales, shiny, testthat (>= 3.0.0)

Depends R (>= 2.10)

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Peer Christensen [aut, cre]

Maintainer Peer Christensen <hr.pchristensen@gmail.com>

Repository CRAN

Date/Publication 2026-07-08 21:30:17 UTC

Contents

`bootstrap_profit` *Bootstrap confidence bands for the profit curve*

Description

Resamples the observations with replacement a number of times, recomputes the cumulative profit curve for each resample, and returns pointwise quantile bands. This turns the single profit curve from [profit()] into a range, making it clear how much of the curve's shape is signal versus sampling noise.

Usage

```
bootstrap_profit(
  x,
  fixed_cost = 0,
  var_cost = 0,
  tp_val = 0,
  prob_col = NA,
  truth_col = NA,
  positive = "Yes",
  n_boot = 200,
  probs = c(0.05, 0.5, 0.95)
)
```

Arguments

x	A data frame containing predicted probabilities of a target event and the actual outcome/class.
fixed_cost	Fixed cost (e.g. of a campaign).
var_cost	Variable cost (e.g. discount offered) per targeted customer.
tp_val	The average value of a True Positive.
prob_col	The unquoted name of the column with probabilities of the event of interest.
truth_col	The unquoted name of the column with the actual outcome/class.
positive	The value in 'truth_col' that identifies the event of interest. Defaults to 'Yes'.
n_boot	Number of bootstrap resamples. Defaults to 200.
probs	Lower, middle and upper quantiles for the bands. Defaults to 'c(0.05, 0.5, 0.95)'.

Value

A data frame with the following columns:

row = row numbers
prop_pop = proportion of the population targeted (row / n)
lower = lower quantile of profit across resamples
median = median profit across resamples
upper = upper quantile of profit across resamples

Examples

```
bootstrap_profit(predictions,
   fixed_cost = 1000,
   var_cost   = 100,
   tp_val     = 2000,
   prob_col   = Yes,
   truth_col  = Churn,
   n_boot     = 100)
```

---

break_even *Break-even and optimal targeting point*

Description

Summarises the two key operating points of the cumulative profit curve produced by [profit()]: the point of maximum profit (the recommended share of customers to target) and the break-even point (the largest share that can be targeted while still turning a profit).

Usage

```
break_even(
  x,
  fixed_cost = 0,
  var_cost = 0,
  tp_val = 0,
  prob_accept = 1,
  prob_col = NA,
  truth_col = NA,
  positive = "Yes"
)
```

Arguments

x	A data frame containing predicted probabilities of a target event and the actual outcome/class.
fixed_cost	Fixed cost (e.g. of a campaign).
var_cost	Variable cost (e.g. discount offered) per targeted customer.
tp_val	The average value of a True Positive.
prob_accept	Probability of the offer being accepted. Variable cost is only incurred when accepted. Defaults to 1.
prob_col	The unquoted name of the column with probabilities of the event of interest.
truth_col	The unquoted name of the column with the actual outcome/class.
positive	The value in 'truth_col' that identifies the event of interest. Defaults to 'Yes'.

Value

A one-row data frame with the following columns:

optimal_row = row at which profit is maximised
optimal_pct = share of the population targeted at maximum profit
max_profit = the maximum profit
breakeven_row = last row at which cumulative profit is still non-negative
breakeven_pct = share of the population targeted at the break-even point

Examples

```
break_even(predictions,
   fixed_cost = 1000,
   var_cost   = 100,
   tp_val     = 2000,
   prob_col   = Yes,
   truth_col  = Churn)
```

compare_models *Compare several models on a business-impact metric*

Description

Computes a targeting curve for two or more models so they can be compared and overlaid on a single plot. Rather than picking a model on AUC alone, this lets you choose the one that produces the most profit (or the steepest gains, lift or ROI) at the share of customers you intend to target.

Usage

```
compare_models(
  x,
  prob_cols,
  truth_col = NA,
  metric = c("profit", "gains", "lift", "roi"),
  fixed_cost = 0,
  var_cost = 0,
  tp_val = 0,
  positive = "Yes"
)
```

Arguments

x	A data frame containing one probability column per model and a shared actual outcome/class.
prob_cols	A character vector of column names holding each model's predicted probabilities.
truth_col	The unquoted name of the column with the actual outcome/class.
metric	The curve to compute: one of "profit", "gains", "lift" or "roi".
fixed_cost	Fixed cost (used by "profit" and "roi").
var_cost	Variable cost per targeted customer (used by "profit" and "roi").
tp_val	The average value of a True Positive (used by "profit" and "roi").
positive	The value in 'truth_col' that identifies the event of interest. Defaults to 'Yes'.

Value

A data frame with the following columns:

model = the model (column) name
row = row numbers
prop_pop = proportion of the population targeted (row / n)
value = the chosen metric at that point

Examples

```
compare_models(predictions,
   prob_cols = c("Yes", "No"),
   truth_col = Churn,
   metric    = "gains")
```

`confusion_payoff` *Confusion matrix and payoff at a single threshold*

Description

Classifies observations at a single probability threshold, returns the resulting confusion matrix (TP/FP/TN/FN) and the associated payoff. This is the single-threshold companion to [profit_thresholds()], which sweeps every threshold, and uses the same value model.

Usage

```
confusion_payoff(
  x,
  threshold = 0.5,
  var_cost = 0,
  prob_accept = 1,
  tp_val = 0,
  fp_val = 0,
  tn_val = 0,
  fn_val = 0,
  prob_col = NA,
  truth_col = NA,
  positive = "Yes"
)
```

Arguments

`x`	A data frame containing predicted probabilities of a target event and the actual outcome/class.
`threshold`	Probability cutoff above which an observation is classified as the positive class.
`var_cost`	Variable cost (e.g. of a campaign offer).
`prob_accept`	Probability of offer being accepted. Defaults to 1.
`tp_val`	The average value of a True Positive. ‘var_cost‘ is automatically subtracted.
`fp_val`	The average cost of a False Positive. ‘var_cost‘ is automatically subtracted.
`tn_val`	The average value of a True Negative.
`fn_val`	The average cost of a False Negative.
`prob_col`	The unquoted name of the column with probabilities of the event of interest.
`truth_col`	The unquoted name of the column with the actual outcome/class.
`positive`	The value in ‘truth_col‘ that identifies the event of interest. Defaults to ’Yes’.

Value

A one-row data frame with the following columns:

threshold = the threshold used
tp = number of true positives
fp = number of false positives
tn = number of true negatives
fn = number of false negatives
payoff = total payoff at the threshold

Examples

```
confusion_payoff(predictions,
   threshold   = 0.3,
   var_cost    = 100,
   prob_accept = .8,
   tp_val      = 2000,
   fp_val      = 0,
   tn_val      = 0,
   fn_val      = -2000,
   prob_col    = Yes,
   truth_col   = Churn)
```

---

cost_revenue *Calculate cost and revenue*

Description

Calculates cost and revenue after sorting observations.

Usage

```
cost_revenue(
  x,
  fixed_cost = 0,
  var_cost = 0,
  tp_val = 0,
  prob_accept = 1,
  prob_col = NA,
  truth_col = NA,
  positive = "Yes"
)
```

Arguments

x	A data frame containing predicted probabilities of a target event and the actual outcome/class.
fixed_cost	Fixed cost (e.g. of a campaign)

var_cost	Variable cost (e.g. discount offered)
tp_val	The average value of a True Positive
prob_accept	Probability of the offer being accepted. Variable cost is only incurred when accepted. Defaults to 1.
prob_col	The unquoted name of the column with probabilities of the event of interest.
truth_col	The unquoted name of the column with the actual outcome/class.
positive	The value in 'truth_col' that identifies the event of interest. Defaults to 'Yes'.

Value

A data frame with the following columns:

row = row numbers
pct = percentiles
cost_sum = cumulated costs
cum_rev = cumulated revenue

Examples

```
cost_revenue(predictions,
   fixed_cost = 1000,
   var_cost   = 100,
   tp_val     = 2000,
   prob_col   = Yes,
   truth_col  = Churn)
```

---

cumulative_gains *Cumulative gains*

Description

Calculates a cumulative gains curve after sorting observations by descending predicted probability. The gain at a given point is the proportion of all actual positives (events) that have been captured by targeting the top rows.

Usage

```
cumulative_gains(x, prob_col = NA, truth_col = NA, positive = "Yes")
```

Arguments

x	A data frame containing predicted probabilities of a target event and the actual outcome/class.
prob_col	The unquoted name of the column with probabilities of the event of interest.
truth_col	The unquoted name of the column with the actual outcome/class.
positive	The value in 'truth_col' that identifies the event of interest. Defaults to 'Yes'.

Value

A data frame with the following columns:

row = row numbers
pct = percentiles
prop_pop = proportion of the population targeted (row / n)
cum_events = cumulated number of actual positives captured
gain = proportion of all positives captured (cum_events / total positives)
baseline = expected gain from random targeting (equal to prop_pop)

Examples

```
cumulative_gains(predictions,
   prob_col  = Yes,
   truth_col = Churn)
```

---

impact_summary *One-line business-impact summary*

Description

Rolls the ranking-based views (profit, ROI, gains and break-even) up into a single row of headline numbers for reporting. It answers: what share of customers should we target, how much profit does that make, what return does it represent, how many churners do we catch, how far can we go before losing money, and what happens if we simply target everyone.

Usage

```
impact_summary(
  x,
  fixed_cost = 0,
  var_cost = 0,
  tp_val = 0,
  prob_accept = 1,
  prob_col = NA,
  truth_col = NA,
  positive = "Yes"
)
```

Arguments

x	A data frame containing predicted probabilities of a target event and the actual outcome/class.
fixed_cost	Fixed cost (e.g. of a campaign).
var_cost	Variable cost (e.g. discount offered) per targeted customer.
tp_val	The average value of a True Positive.

prob_accept	Probability of the offer being accepted. Variable cost is only incurred when accepted. Defaults to 1.
prob_col	The unquoted name of the column with probabilities of the event of interest.
truth_col	The unquoted name of the column with the actual outcome/class.
positive	The value in 'truth_col' that identifies the event of interest. Defaults to 'Yes'.

Value

A one-row data frame with the following columns:

optimal_pct = share of the population targeted at maximum profit
max_profit = the maximum profit
roi_at_optimum = ROI at the maximum-profit point
capture_at_optimum = proportion of all positives captured at that point
breakeven_pct = largest share that can be targeted while staying profitable
profit_target_all = profit if every customer is targeted

Examples

```
impact_summary(predictions,
   fixed_cost = 1000,
   var_cost   = 100,
   tp_val     = 2000,
   prob_col   = Yes,
   truth_col  = Churn)
```

---

lift_curve *Lift*

Description

Calculates a cumulative lift curve after sorting observations by descending predicted probability. Lift is the ratio between the proportion of positives captured by the model and the proportion that would be expected from random targeting. A lift of 1 is no better than random. Named 'lift_curve()' to avoid clashing with 'purrr::lift()'.

Usage

```
lift_curve(x, prob_col = NA, truth_col = NA, positive = "Yes")
```

Arguments

x	A data frame containing predicted probabilities of a target event and the actual outcome/class.
prob_col	The unquoted name of the column with probabilities of the event of interest.
truth_col	The unquoted name of the column with the actual outcome/class.
positive	The value in 'truth_col' that identifies the event of interest. Defaults to 'Yes'.

Value

A data frame with the following columns:

row = row numbers
pct = percentiles
prop_pop = proportion of the population targeted (row / n)
gain = proportion of all positives captured
lift = gain / prop_pop

Examples

```
lift_curve(predictions,
   prob_col  = Yes,
   truth_col = Churn)
```

`marginal_profit` *Marginal profit per bin*

Description

Splits observations into equally sized bins (deciles by default) after sorting by descending predicted probability, and calculates the profit contributed by each bin. This makes it easy to see where targeting additional customers stops paying off: the first bin whose ‘marginal_profit‘ turns negative marks the point of diminishing returns. The cumulative sum of ‘marginal_profit‘ reconciles with the cumulative profit reported by [profit()].

Usage

```
marginal_profit(
  x,
  fixed_cost = 0,
  var_cost = 0,
  tp_val = 0,
  prob_accept = 1,
  bins = 10,
  prob_col = NA,
  truth_col = NA,
  positive = "Yes"
)
```

Arguments

`x`	A data frame containing predicted probabilities of a target event and the actual outcome/class.
`fixed_cost`	Fixed cost (e.g. of a campaign). Charged once, to the first bin.
`var_cost`	Variable cost (e.g. discount offered) per targeted customer.
`tp_val`	The average value of a True Positive.

prob_accept	Probability of the offer being accepted. Variable cost is only incurred when accepted. Defaults to 1.
bins	Number of equally sized bins. Defaults to 10 (deciles).
prob_col	The unquoted name of the column with probabilities of the event of interest.
truth_col	The unquoted name of the column with the actual outcome/class.
positive	The value in 'truth_col' that identifies the event of interest. Defaults to 'Yes'.

Value

A data frame with the following columns:

bin = bin number (1 = highest probabilities)
n = number of observations in the bin
events = number of actual positives in the bin
cost = cost incurred in the bin
revenue = revenue generated in the bin
marginal_profit = revenue - cost for the bin
cum_profit = cumulative profit up to and including the bin

Examples

```
marginal_profit(predictions,
   fixed_cost = 1000,
   var_cost   = 100,
   tp_val     = 2000,
   bins       = 10,
   prob_col   = Yes,
   truth_col  = Churn)
```

---

modelimpact-plots *Plot modelimpact results*

Description

Every analysis function in the package returns a classed data frame that can be visualised directly with `autoplot()`, for example `autoplot(profit(...))`. The 'plot_*()' functions are thin, back-compatible wrappers around the corresponding 'autoplot()' method. **ggplot2** is only required when a plot is actually drawn.

Usage

```
autoplot.mi_profit(object, ...)

autoplot.mi_cost_revenue(object, ...)

autoplot.mi_roi(object, ...)
```

```
autoplot.mi_gains(object, ...)

autoplot.mi_lift(object, ...)

autoplot.mi_marginal(object, ...)

autoplot.mi_thresholds(object, ...)

autoplot.mi_roc(object, slope = NULL, ...)

autoplot.mi_compare(object, ...)

autoplot.mi_bootstrap(object, ...)

plot_profit(data)

plot_cost_revenue(data)

plot_roi(data)

plot_gains(data)

plot_lift(data)

plot_marginal(data)

plot_thresholds(data)

plot_roc(data, slope = NULL)
```

Arguments

`object`, `data`	The data frame returned by the matching modelimpact function (e.g. the output of [profit()] for ‘autoplot()‘ / ‘plot_profit()‘).
`...`	Unused, for S3 compatibility.
`slope`	For ROC objects only: optional slope of an iso-profit line. When supplied, the cost-sensitive optimal operating point (maximising ‘tpr - slope * fpr‘) is highlighted. The business-optimal slope equals ‘(cost_fp / value_fn) * (n_neg / n_pos)‘.

Value

A ‘ggplot‘ object.

payoff_grid *Payoff sensitivity grid*

Description

Computes payoff across a grid of classification thresholds and offer-acceptance probabilities, so the robustness of the optimal operating point can be judged at a glance (for example as a heatmap). Uses the same value model as [profit_thresholds()] and [confusion_payoff()].

Usage

```
payoff_grid(
  x,
  thresholds = seq(0, 1, 0.02),
  prob_accept = seq(0, 1, 0.1),
  var_cost = 0,
  tp_val = 0,
  fp_val = 0,
  tn_val = 0,
  fn_val = 0,
  prob_col = NA,
  truth_col = NA,
  positive = "Yes"
)
```

Arguments

x	A data frame containing predicted probabilities of a target event and the actual outcome/class.
thresholds	A vector of probability cutoffs to evaluate. Defaults to ‘seq(0, 1, 0.02)‘.
prob_accept	A vector of offer-acceptance probabilities to evaluate. Defaults to ‘seq(0, 1, 0.1)‘.
var_cost	Variable cost (e.g. of a campaign offer).
tp_val	The average value of a True Positive. ‘var_cost‘ is automatically subtracted.
fp_val	The average cost of a False Positive. ‘var_cost‘ is automatically subtracted.
tn_val	The average value of a True Negative.
fn_val	The average cost of a False Negative.
prob_col	The unquoted name of the column with probabilities of the event of interest.
truth_col	The unquoted name of the column with the actual outcome/class.
positive	The value in ‘truth_col‘ that identifies the event of interest. Defaults to ’Yes’.

Value

A data frame with the following columns:

threshold = classification threshold
prob_accept = offer-acceptance probability
payoff = payoff for that combination

Examples

```
payoff_grid(predictions,
   var_cost  = 100,
   tp_val    = 2000,
   fn_val    = -2000,
   prob_col  = Yes,
   truth_col = Churn)
```

---

predictions *Predictions from a customer churn model.*

Description

A dataset containing 2145 observations with four columns specifying predicted probabilities and predicted and actual class.

Usage

```
predictions
```

Format

A data frame with 2145 rows and 4 variables:

predict Predicted class

No Predicted probability of class 'No'

Yes Predicted probability of class 'Yes'

Churn Actual class ...

`profit` *Calculate profit*

Description

Calculates profit after sorting observations.

Usage

```
profit(
  x,
  fixed_cost = 0,
  var_cost = 0,
  tp_val = 0,
  prob_accept = 1,
  prob_col = NA,
  truth_col = NA,
  positive = "Yes"
)
```

Arguments

`x`	A data frame containing predicted probabilities of a target event and the actual outcome/class.
`fixed_cost`	Fixed cost (e.g. of a campaign)
`var_cost`	Variable cost (e.g. discount offered)
`tp_val`	The average value of a True Positive
`prob_accept`	Probability of the offer being accepted. Variable cost is only incurred when accepted. Defaults to 1.
`prob_col`	The unquoted name of the column with probabilities of the event of interest.
`truth_col`	The unquoted name of the column with the actual outcome/class.
`positive`	The value in ‘truth_col‘ that identifies the event of interest. Defaults to ’Yes’.

Value

A data frame with the following columns:

row = row numbers
pct = percentiles
profit = profit for number of rows selected

Examples

```
profit(predictions,
     fixed_cost = 1000,
     var_cost   = 100,
     tp_val     = 2000,
     prob_col   = Yes,
     truth_col  = Churn)
```

`profit_thresholds` *Find optimal threshold for churn prediction (class)*

Description

Finds the optimal threshold (from a business perspective) for classifying churners.

Usage

```
profit_thresholds(
  x,
  var_cost = 0,
  prob_accept = 1,
  tp_val = 0,
  fp_val = 0,
  tn_val = 0,
  fn_val = 0,
  prob_col = NA,
  truth_col = NA,
  positive = "Yes"
)
```

Arguments

`x`	A data frame containing predicted probabilities of a target event and the actual outcome/class.
`var_cost`	Variable cost (e.g. of a campaign offer)
`prob_accept`	Probability of offer being accepted. Defaults to 1.
`tp_val`	The average value of a True Positive. ‘var_cost‘ is automatically subtracted.
`fp_val`	The average cost of a False Positive. ‘var_cost‘ is automatically subtracted.
`tn_val`	The average value of a True Negative.
`fn_val`	The average cost of a False Negative.
`prob_col`	The unquoted name of the column with probabilities of the event of interest.
`truth_col`	The unquoted name of the column with the actual outcome/class.
`positive`	The value in ‘truth_col‘ that identifies the event of interest. Defaults to ’Yes’.

Value

A data frame with the following columns:

threshold = prediction thresholds
payoff = calculated profit for each threshold

Examples

```
profit_thresholds(predictions,
   var_cost    = 100,
   prob_accept = .8,
   tp_val      = 2000,
   fp_val      = 0,
   tn_val      = 0,
   fn_val      = -2000,
   prob_col    = Yes,
   truth_col   = Churn)
```

---

roc_pr *ROC and precision-recall curve data*

Description

Calculates the points of the ROC curve and the precision-recall curve by walking down the observations sorted by descending predicted probability. The returned data frame contains everything needed to draw both curves and to overlay an iso-profit / cost-sensitive operating point.

Usage

```
roc_pr(x, prob_col = NA, truth_col = NA, positive = "Yes")
```

Arguments

x	A data frame containing predicted probabilities of a target event and the actual outcome/class.
prob_col	The unquoted name of the column with probabilities of the event of interest.
truth_col	The unquoted name of the column with the actual outcome/class.
positive	The value in 'truth_col' that identifies the event of interest. Defaults to 'Yes'.

Value

A data frame with the following columns:

threshold = probability at each operating point
tp = true positives when classifying every row at or above the point as positive
fp = false positives
tpr = true positive rate / recall / sensitivity (tp / all positives)
fpr = false positive rate (fp / all negatives)
precision = tp / (tp + fp)
recall = same as tpr

Examples

```
roc_pr(predictions,
   prob_col  = Yes,
   truth_col = Churn)
```

`roi` *Calculate Return on investment (ROI)*

Description

Calculates ROI after sorting observations with ROI defined as (Current Value - Start Value) / Start Value

Usage

```
roi(
  x,
  fixed_cost = 0,
  var_cost = 0,
  tp_val = 0,
  prob_accept = 1,
  prob_col = NA,
  truth_col = NA,
  positive = "Yes"
)
```

Arguments

`x`	A data frame containing predicted probabilities of a target event and the actual outcome/class.
`fixed_cost`	Fixed cost (e.g. of a campaign)
`var_cost`	Variable cost (e.g. discount offered)
`tp_val`	The average value of a True Positive
`prob_accept`	Probability of the offer being accepted. Variable cost is only incurred when accepted. Defaults to 1.
`prob_col`	The unquoted name of the column with probabilities of the event of interest.
`truth_col`	The unquoted name of the column with the actual outcome/class.
`positive`	The value in ‘truth_col‘ that identifies the event of interest. Defaults to ’Yes’.

Value

A data frame with the following columns:

row = row numbers
pct = percentiles
cum_rev = cumulated revenue
cost_sum = cumulated costs
roi = return on investment

Examples

```
roi(predictions,
   fixed_cost = 1000,
   var_cost   = 100,
   tp_val     = 2000,
   prob_col   = Yes,
   truth_col  = Churn)
```

`run_app` *Launch the interactive modelimpact Shiny app*

Description

Opens a small Shiny application for exploring the package interactively. You can use the bundled ‘predictions‘ data or upload your own CSV (choosing comma or semicolon as the separator), pick the probability and outcome columns and the positive class, adjust the cost/value parameters, and watch every plot update live.

Usage

```
run_app(...)
```

Arguments

`...`	Additional arguments passed to [shiny::runApp()].

Details

Requires the **shiny** and **ggplot2** packages, which are only needed for this function.

Value

Called for its side effect of starting the app; returns nothing.

Examples

```
## Not run:
run_app()

## End(Not run)
```

tornado *Tornado sensitivity analysis of maximum profit*

Description

Varies each cost/value assumption up and down by a fixed fraction, one at a time, and records how the maximum achievable profit (the peak of the [profit()] curve) responds. The result feeds a tornado plot, ordering the assumptions by how much they move the bottom line.

Usage

```
tornado(
  x,
  fixed_cost = 0,
  var_cost = 0,
  tp_val = 0,
  prob_col = NA,
  truth_col = NA,
  positive = "Yes",
  variation = 0.2
)
```

Arguments

x	A data frame containing predicted probabilities of a target event and the actual outcome/class.
fixed_cost	Baseline fixed cost.
var_cost	Baseline variable cost per targeted customer.
tp_val	Baseline average value of a True Positive.
prob_col	The unquoted name of the column with probabilities of the event of interest.
truth_col	The unquoted name of the column with the actual outcome/class.
positive	The value in ‘truth_col‘ that identifies the event of interest. Defaults to ’Yes’.
variation	Fraction by which each parameter is moved up and down. Defaults to 0.2 (+/- 20 percent).

Value

A data frame, one row per parameter, ordered by descending ‘swing‘:

parameter = the assumption varied
low_value = parameter value on the low side
high_value = parameter value on the high side
low_profit = maximum profit at the low value
high_profit = maximum profit at the high value
base_profit = maximum profit at the baseline values
swing = absolute difference between low_profit and high_profit

Examples

```
tornado(predictions,
   fixed_cost = 1000,
   var_cost   = 100,
   tp_val     = 2000,
   prob_col   = Yes,
   truth_col  = Churn)
```

Index