Model performance with *yardstick* : : **CHEAT SHEET**

Intro

yardstick measures how well a model predicts. It contains a wide-ranging selection of metrics that work with classification, regression, and survival models. Every metric returns the same tidy table output, which makes it possible to combine multiple metrics, work over grouped data automatically, and plug into model tuning to compare candidate models.

```
library(tidymodels)
two_class_example |>
  accuracy(truth = truth, estimate = predicted)
```

Example of combining multiple metrics:

```
my_metrics <- metric_set(accuracy,
sensitivity, specificity)
two_class_example |>
  my_metrics(truth = truth, estimate = predicted)
```

VECTOR VERSIONS

Every metric has a ***_vec()** counterpart that takes, and returns, vectors instead of a data frame.

```
accuracy_vec(two_class_example$truth,
    two_class_example$predicted)
```

Classification

[metric](data, truth, estimate, estimator = NULL, na_rm = TRUE, case_weights = NULL, event_level = "first")

Labels

In the formulas, the following abbreviations will be used for the following:

		Truth	
		Event	Non-Event
Prediction	Event	*TP*	*FP*
	Non-Event	*FN*	*TN*

More than two classes

There are three ways that the metric is calculated, if the prediction has more than two classes. The choice is made via the `estimator` argument:

1. Mean per class — `estimator = "macro"` (default)
2. Mean weighted by class size — `estimator = "macro_weighted"`
3. Pool TP, FP and FN across classes and score once — `estimator = "micro"`

Classification (cont')

POSITIVE / NEGATIVE RATES

$Se = \frac{TP}{TP + FN}$ **sensitivity**() / **recall**() - Proportion of true events that are predicted as events.

$Sp = \frac{TN}{FP + TN}$ **specificity**() - Proportion of true non-events that are predicted as non-events.

$\frac{FP}{FP + TN}$ **fall_out**() - False positive rate, 1 minus specificity.

$\frac{FN}{TP + FN}$ **miss_rate**() - False negative rate, 1 minus sensitivity.

$Pr = \frac{TP}{TP + FP}$ **precision**() - Proportion of predicted events that are true events.

$D = \frac{TP + FP}{TP + FP + FN + TN}$ **detection_prevalence**() - Number of predicted events divided by the total number of predictions.

OVERALL AGREEMENT

$\frac{TP + TN}{TP + FP + FN + TN}$ **accuracy**() - Proportion of the data that is predicted correctly.

$\frac{Se + Sp}{2}$ **bal_accuracy**() - Average of sensitivity and specificity. Useful with unbalanced classes.

$\frac{(1 + \beta^2) \times Pr \times Se}{(\beta^2 \times Pr) + Se}$ **f_meas**(beta = 1) - Weighted harmonic mean of precision and recall. Use beta to favor one over the other.

$Pe = ((TP + FP)(TP + FN) + (FN + TN)(FP + TN)) / N^2$

$\frac{accuracy - Pe}{1 - Pe}$ **kap**(weighting = "none") - Kappa. Like accuracy(), but normalized by the accuracy expected by chance alone.

mcc() - Matthews correlation coefficient. A measure of correlation for categorical data.

$$\frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

sedi() - Symmetric Extremal Dependence Index. Stays reliable at extreme prevalence. MCC & kappa degrade.

$$\frac{Log(1 - Sp) - Log(Sp) + Log(1 - Se) - Log(Se)}{Log(1 - Sp) + Log(Sp) + Log(1 - Se) + Log(Se)}$$

PREDICTIVE VALUES

These default to the event rate in truth. Set prevalence when that rate differs from the population you will score.

Pv = Prevalence

ppv(prevalence = NULL) - Positive predictive value. Probability of a true event when a event is predicted.

$$\frac{Se \times Pv}{(Se \times Pv) + ((1 - Sp) \times (1 - Pv))}$$

npv(prevalence = NULL) - Negative predictive value. Probability of a true non-event when a non-event is predicted.

$$\frac{Sp \times (1 - Pv)}{((1 - Se) \times Pv) + (Sp \times (1 - Pv))}$$

COMBINED INDEXES

$Se + Sp - 1$ **j_index**() - Youden's J, sensitivity plus specificity minus 1.

$Pr + \left(\frac{TN}{FN + TN}\right) - 1$ **markedness**() - precision + inverse precision - 1. The predictive-value counterpart to j_index().

$\sqrt{(1 - Se)^2 + (1 - Sp)^2}$ **roc_dist**() - Euclidean distance from (sensitivity, specificity) to the ideal corner in ROC space. Useful for picking a threshold.

Class Probabilities

[metric](data, truth, ..., estimator = NULL, na_rm = TRUE, event_level = "first", case_weights = NULL)

Pass class probability columns to …, not an estimate argument. One column for two classes, one per class for more.

PROBABILITY QUALITY

Ok = 1 when the observed class is k, else 0
Pk = predicted probability of class k

$\frac{Mean(Sum((Ok - Pk)^2))}{2}$ **brier_class**() - Brier score. Mean squared difference between the predicted probabilities and the observed classes.

$L = Sum(Ok \times Log(Pk)) - (Mean(L))$ **mn_log_loss**(sum = FALSE) - Log loss. Penalizes confident predictions that are wrong.

$Mean(Sum(Pk \times Cost))$ **classification_cost**(costs = NULL) - Mean cost of a poor prediction, using your own cost per truth and estimate pair.

Class Probabilities (cont')

CURVE SUMMARIES

[metric](data, truth, ..., na_rm = TRUE, case_weights = NULL)

roc_auc() - Area under the ROC curve. See roc_curve() for the full curve.

roc_aunp() - Each class against the rest, weighted by the a priori class distribution. Same as roc_auc(estimator = "macro_weighted").

roc_aunu() - Each class against the rest, weighted uniformly. Same as roc_auc(estimator = "macro").

pr_auc() - Area under the precision recall curve. See pr_curve() for the full curve.

average_precision() - Weighted average of the precision values from pr_curve(). Avoids the ambiguity pr_auc() has when recall is 0.

gain_capture() - Area under a gain curve. See gain_curve() for the full curve.

CURVE

**_curve*(data, truth, ..., na_rm = TRUE, event_level = "first", case_weights = NULL)

These cannot be used in a metric_set(), but can be plotted with autoplot().

roc_curve(thresholds = NULL) - Sensitivity against one minus specificity at every threshold. Returns .threshold, specificity, sensitivity.

pr_curve() - Precision against recall at every threshold. Returns .threshold, recall, precision.

gain_curve() - Percent of events found against percent of data tested. Returns .n, .n_events, .percent_tested, .percent_found.

lift_curve() - Percent found divided by percent tested. Returns .n, .n_events, .percent_tested, .lift.

Model performance with *yardstick* : : **CHEAT SHEET**

Regression

[metric](data, truth, estimate, na_rm = TRUE, case_weights = NULL)

ERROR

***Tr** = Truth | **Es** = Estimate | **Er = Tr - Es** (Error)*

$Mean(Abs(Er))$ **mae**() - Mean absolute error.

$Mean(Er)$ **msd**() - Mean signed deviation. Averages the signed differences, it measures bias not size of error.

$Mean(Er^2)$ **mse**() - Mean squared error.

$Rm = \sqrt{Mean(Er^2)}$ **rmse**() - Root mean squared error.

$\frac{Rm}{Max(Tr) - Min(Tr)}$ **rmse_relative**() - RMSE normalized by the range of the true values. Also called NRMSE.

$\frac{Mean(Abs(Er))}{Mean(Abs(Tr - Lag(Tr, m)))}$ **mase**(m = 1L, mae_train = NULL) - Mean absolute scaled error. Scale independent, for forecast error. Order rows by time first.

$Mean2(x) = Mean(x) \times 100$

$Mean2(\frac{Er}{Tr})$ **mpe**() - Mean percent error. Signed, so it measures bias. Returns Inf if any truth value is 0.

$Mean2(Abs(\frac{Er}{Tr}))$ **mape**() - Mean absolute percent error.

$Mean2\left(\frac{2 \times Abs(Er)}{Abs(Tr) + Abs(Es)}\right)$ **smape**() - Symmetric mean absolute percent error.

AGREEMENT AND CORRELATION

$Cor(Tr, Es)^2$ **rsq**() - R squared, computed from the correlation. Between 0 and 1.

$\frac{Sum((Tr - Es)^2)}{Sum((Tr - Mean(Tr))^2)}$ **rsq_trad**() - R squared, computed from sums of squares. Can go negative.

$\frac{2 \times Cov(Tr, Es))}{Var(Tr) + Var(Es) + (Mean(Tr) - Mean(Es))^2}$ **ccc**(bias = FALSE) - Concordance correlation coefficient. Measures agreement with the 45 degree line, not just correlation.

$\frac{Sd(Tr)}{Rm}$ **rpd**() - Ratio of performance to deviation. Consistency with the observed values, not accuracy.

$\frac{IQR(Tr)}{Rm}$ **rpiq**() - Ratio of performance to inter-quartile.

AGREEMENT AND CORRELATION (CONT')

***Mp** = Mean(Abs(Positive Er)) | **Mn** = Mean(Abs(Negative Er))*

$\frac{Cor(Tr, Es) \times Min(Mn, Mp)}{Max(Mn, Mp)}$ **iic**() - Index of ideality of correlation. Combines the correlation with the mean absolute error.

ROBUST TO OUTLIERS

huber_loss(delta = 1) - Quadratic for small residuals, linear for large ones. Less sensitive to outliers than rmse().

huber_loss_pseudo(delta = 1) - Smooth approximation of huber_loss().

RANKING AND COUNTS

gini_coef() - Normalized Gini coefficient. Measures ranking ability from the Lorenz curve. Used for risk and loss cost models.

poisson_log_loss() - Log loss for count outcomes, using the Poisson distribution.

Fairness

[*metric*](by)

Each builds a metric, not a value. Pass the protected group column unquoted to by, then use it like any metric. 0 means equal scores across groups.

g** = Protected group | **Range(x) = Max(x) - Min(x)

$Range(D(g))$ **demographic_parity**() - Range of detection_prevalence() across groups. Same predicted positive rate everywhere. Ignores the true outcome.

$Range(Se(g))$ **equal_opportunity**() - Range of sensitivity() across groups. Same true positive rate everywhere.

$Max(Range(Se(g)), Range(Sp(g)))$ **equalized_odds**() - Largest range of sensitivity() or specificity() across groups. Same error rates of every kind.

new_groupwise_metric(fn, name, aggregate, direction = "minimize") - Build your own from any metric function, summarizing its per group values with aggregate.

Survival

All of these are for right censored data. Pass a survival::Surv() object to truth.

DYNAMIC PREDICTIONS

[metric](data, truth, ..., na_rm = TRUE, case_weights = NULL)

Pass the list column of predicted survival probabilities from a censored model to .... Results come back one row per .eval_time.

brier_survival() - Mean squared error at each evaluation time.

brier_survival_integrated() - One Brier score across all evaluation times.

roc_auc_survival() - Area under the ROC survival curve at each evaluation time.

roc_curve_survival() - The full ROC survival curve, for plotting.

STATIC AND LINEAR PREDICTOR

[metric](data, truth, estimate).

concordance_survival() - Concordance index. Ranking ability across the whole follow-up, with no evaluation time.

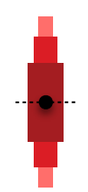
royston_survival() - Royston-Sauerbrei D statistic. Separation between risk groups, from a linear predictor.

Other metrics

ORDINAL

ranked_prob_score(data, truth, ..., na_rm = TRUE, case_weights = NULL) - Ranked probability score. truth is an ordered factor; ... takes one probability column per level, in order. Credits near misses.

QUANTILE



weighted_interval_score(data, truth, estimate, quantile_levels = NULL, na_rm = TRUE, quantile_estimate_nas = "impute", case_weights = NULL) - Quantile based approximation of the continuous ranked probability score. Generalizes absolute error.

Operations

```
library(tidymodels)
```

METRIC SETS

metric_set(...) - Combines metric functions into one function that computes all of them at once. Only compatible types can be mixed.

```
class_mets <- metric_set(
accuracy,sensitivity,specificity)
```

get_metrics(type) - Returns a metric_set() of every metric of the given type, such as "class" or "prob".

```
class_mets <- get_metrics("class")
```

metrics(data, truth, estimate, ...) - Computes a default set of metrics, chosen by the class of truth.

```
metrics(hpc_cv, obs, pred)
```

metric_tweak(.name, .fn, ...) - Presets optional arguments on a metric. Do this before the metric goes into a metric_set(), since you cannot change them afterwards.

```
f2 <- metric_tweak("f2", f_meas, beta = 2)
```

CONFUSION MATRIX

conf_mat(data, truth, estimate, dnn = c("Prediction", "Truth")) - Cross tabulates observed against predicted classes.

```
cm <- conf_mat(hpc_cv, obs, pred)
```

summary(object, prevalence = NULL, beta = 1, estimator = NULL, event_level = "first") - Computes most of the classification metrics at once from the matrix.

```
summary(cm)
```

tidy(x) - Turns the matrix into a tibble with one row per cell.

```
tidy(cm)
```

autoplot(object, type = "mosaic") - Draws the matrix. Use type = "heatmap" for the other style.

```
autoplot(cm, type = "heatmap")
```