# Machine Learning for Economists: Part 1 – Criterion Functions

Michal Andrle
International Monetary Fund

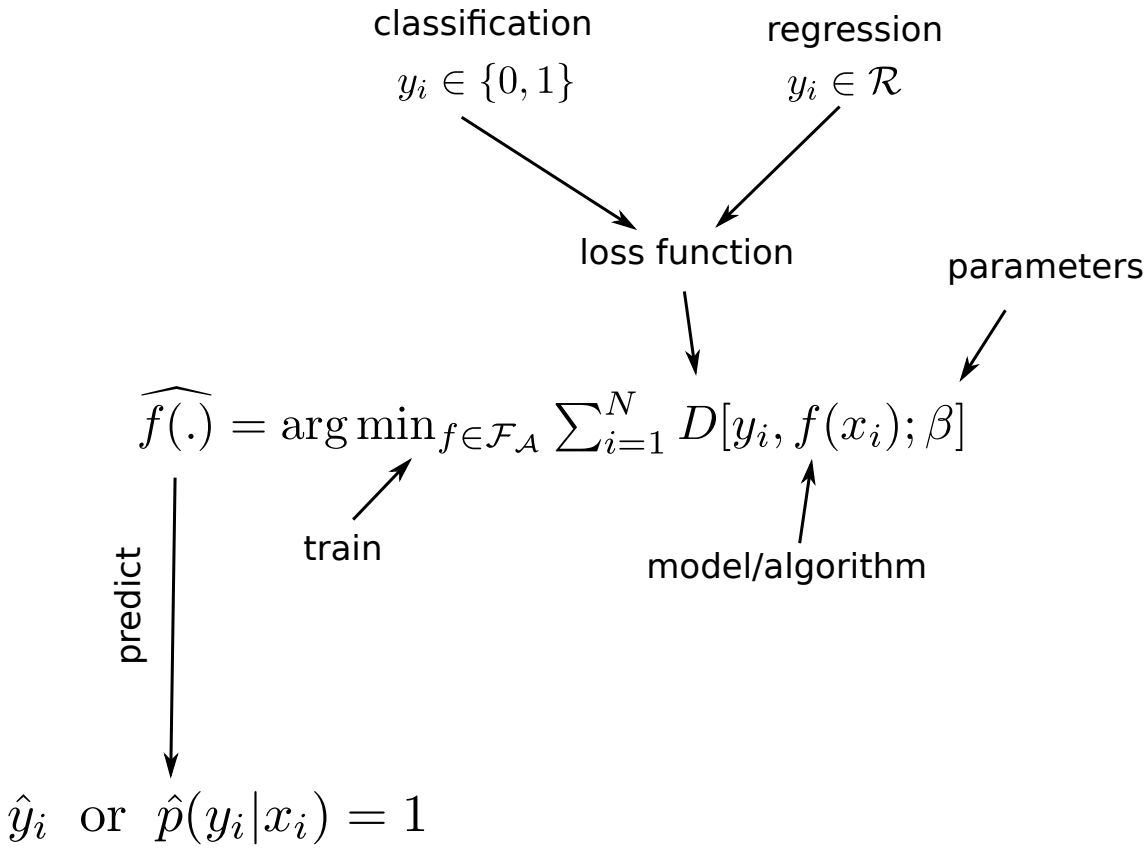
Washington, D.C.,
October, 2018

# Disclaimer #1:

**The views expressed herein are those of the authors and should not be attributed to the International Monetary Fund, its Executive Board, or its management.**

# LOSS FUNCTIONS

# Loss Functions in Algorithm Training



# Decisions and Loss Function Choice (A)

Loss functions **should vary** with the goals of applications...

**Regression**:

- Interested in forecasting cycles? Is optimizing one-step-ahead forecast error the right thing?
- Is your model meant to explain all frequencies (likelihood)
- Are your shocks close to Normal or fat tails, t-distrib? (robustness)
- ...

**Classification**:

- Is one type of error much worse than another type of error?
- Are you worried your decision will kill someone? Or crash a car?
- Is the sample *unbalanced*? E.g. just 3% of positive samples?
- ...

# Decisions and Loss Function Choice (B)

**Convex surrogate loss functions**:

Sometimes the criterion of interest is too costly to compute, too hard to optimize (non-smooth), or both.

Design a convex surrogate!

**Example:**

Minimizing Misclassification Rate, AUC (Area and the Curve) may be hard to work with directly...

Yet, in principle optimize a criterion that you really care for. Especially with smaller data/problems...

See Yi Shen (2003) or Hand and Vinciatti (2003, AmStat), ...

# Loss Function 'Bestiary'

Numerical loss functions and surrogate loss functions:

1. **REGRESSION:**
   - mean squared error (L2)
   - mean absolute deviation (L1)
   - Huber loss function
   - Student-t loss function
   - ...

2. **CLASSIFICATION:** multivariate $\times$ binary
   - log-loss/cross-entropy/Kullback-Leibler divergence
   - Brier score
   - misclassification rate
   - 0-1 loss
   - hinge loss
   - ...

By principle of **estimation by analogy** (Goldberger, 1964), one should create such a loss function that expresses best the problem at hand...

# Regression Loss Functions (1a)

**Squared Error ("L2")** $(y - \hat{y})^2$

**Absolute Deviation ("L1")** $|y - \hat{y}|$

**Huber loss**

$$L_H = \begin{cases} r^2/2 & \text{if} |r| \leq \delta \\ \delta|r| - \delta^2/2 & \text{if} |r| > \delta \end{cases} \quad (1)$$

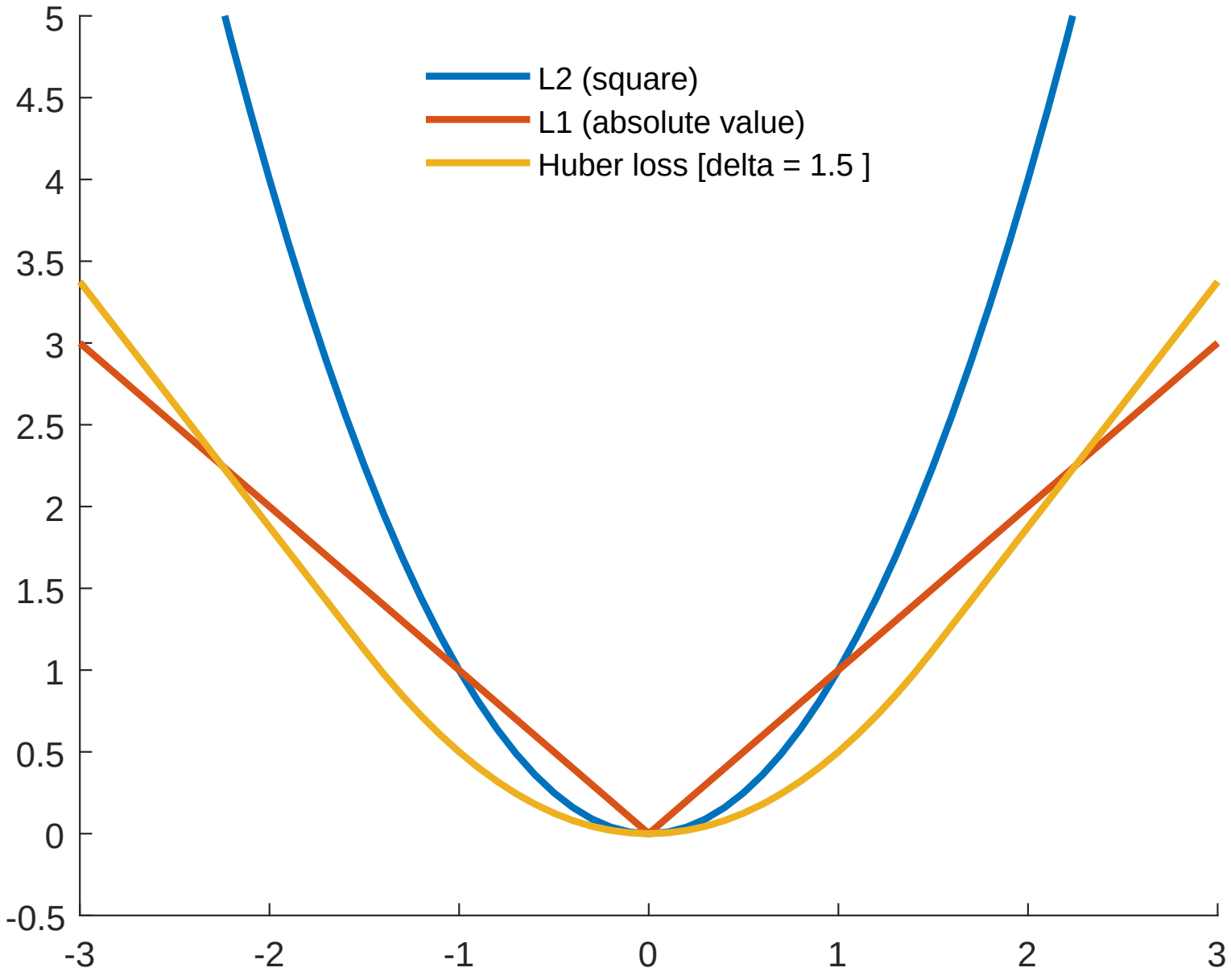
with $r \equiv y - \hat{y}, \quad \hat{y} = f(x)$

**General Minkowski** $|y - \hat{y}|^q$

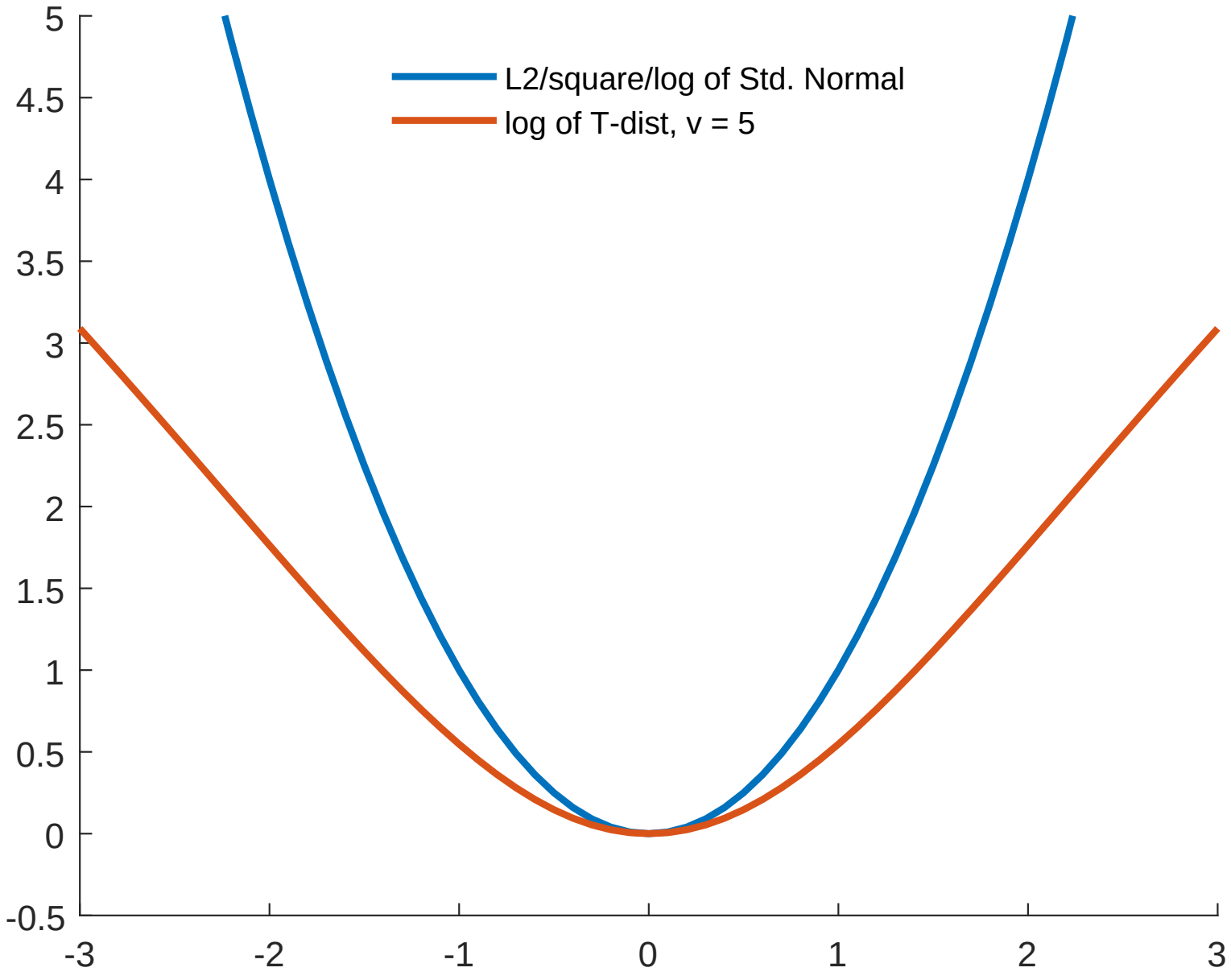
...

User-defined **asymmetric** loss function should reflect the problem and the cost of decision errors...

# Regression Loss Functions (1b)



# Regression Loss Functions (1c)



# Classification Loss Functions

# Classification – How to Encode?

**1) Two-Class Problems**:

For **binary** classification encoding $\{0|1\}$ or $\{-1|1\}$ is used.

**2) Multiple-Class Problems**:

"**One Hot**" Encoding of CATEGORICAL variables

| obs. | red | green | blue |
|---|---|---|---|
| #1 | 1 | 0 | 0 |
| #2 | 0 | 1 | 0 |
| #3 | 0 | 0 | 1 |
| ⋮ | | | |
| #N | 0 | 1 | 0 |

# Classification – Model Outcomes

- **Class outcome**
  - Only class chosen is reported (e.g. 0 or 1). No probabilities, e.g. SVMs or KNN. . .
  - Usually can be converted to prob.-like outcome
- **Probability or probability-like outcome**
  - Probability-like outcome for given input
  - Threshold levels needed for assignment

# Classification – Getting prob-like outcomes

Without a probabilistic model, model predictions about classes need to be mapped to 'probability-like' scores...

- **Real-valued index to probability**, $z \in \mathcal{R} \rightarrow [0, 1]$

  Squashing functions (softmax, ...)

- **Class outcome to probability**

  Relative counts, ...

# Classification – “Squashing functions”

**Softmax** (Multinoulli Distribution)

$$q_i = \text{softmax}(\mathbf{z})_i = \frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_j)}, \tag{2}$$

with $\sum_i q_i = 1, 0 \leq q_i \leq 1$, $z_i \in \mathcal{R}$.

Now $q$ satisfies all the properties of a probability.

This is an **important concept** in classification and neural networks.

**Examples:** logistic regression, neural-net classifiers, . . .

# Classification – Majority Class Outcome

With a learner classifying by the ‘majority voting’ for a class getting a probability-like scores is simple

$$p(class_i) = \frac{\#\ \text{class}_i\ \text{elements}}{\#\ \text{group elements}}$$

# Classification – Evaluating Loss Function

The **classifier** model may return probability or probability-like quantities for each category, *q*, to compare with data, *p*.

| obs. | red | green | blue | R | G | B |
|---|---|---|---|---|---|---|
| | p (data) | | | q (model fit) | | |
| #1 | 1 | 0 | 0 | 0.7 | 0.1 | 0.2 |
| #2 | 0 | 1 | 0 | 0.3 | 0.6 | 0.1 |
| #3 | 0 | 0 | 1 | 0.1 | 0.1 | 0.8 |
| ⋮ | | | | | | |
| #N | 0 | 1 | 0 | 0.3 | 0.3 | 0.4 |

Now, the model predictions need to be compared to data.

Multiple approaches possible, with different loss functions/distributions.

# Classification – Common Loss Functions

Some commonly-used loss functions to **train** the model are:

- **Log-loss (entropy, K/L distance)** – de facto standard
- Squared error (Brier score)
- Misclassification error (Accuracy)
- ...

Most often, these are **surrogate** loss functions. To evaluate the results, other information is brought on board (coming up...)

For designing **"proper scoring rules"** see T. Gneiting and A.E. Raftery (2007, JASA).

# Classification – Distance between Distributions

**Kullback-Leibler divergence:**
Used to measure distance between distributions $P$ and $Q$.

$$D_{KL}(P||Q) = \underbrace{-\sum_x p(x) \log q(x)}_{\text{cross entropy}} + \underbrace{\sum_x p(x) \log p(x)}_{\text{entropy}},$$

where $p(x), q(x)$ are either a probability or probability-like measure, $0 \leq p(x) \leq 1$.

For a **given** distribution, $P$, K-L divergence, $D_{KL}$, and cross entropy differ by a constant (of entropy of $p$).

In the `code`, take care of $0 \times log(0)$ cases and such...

# Classification

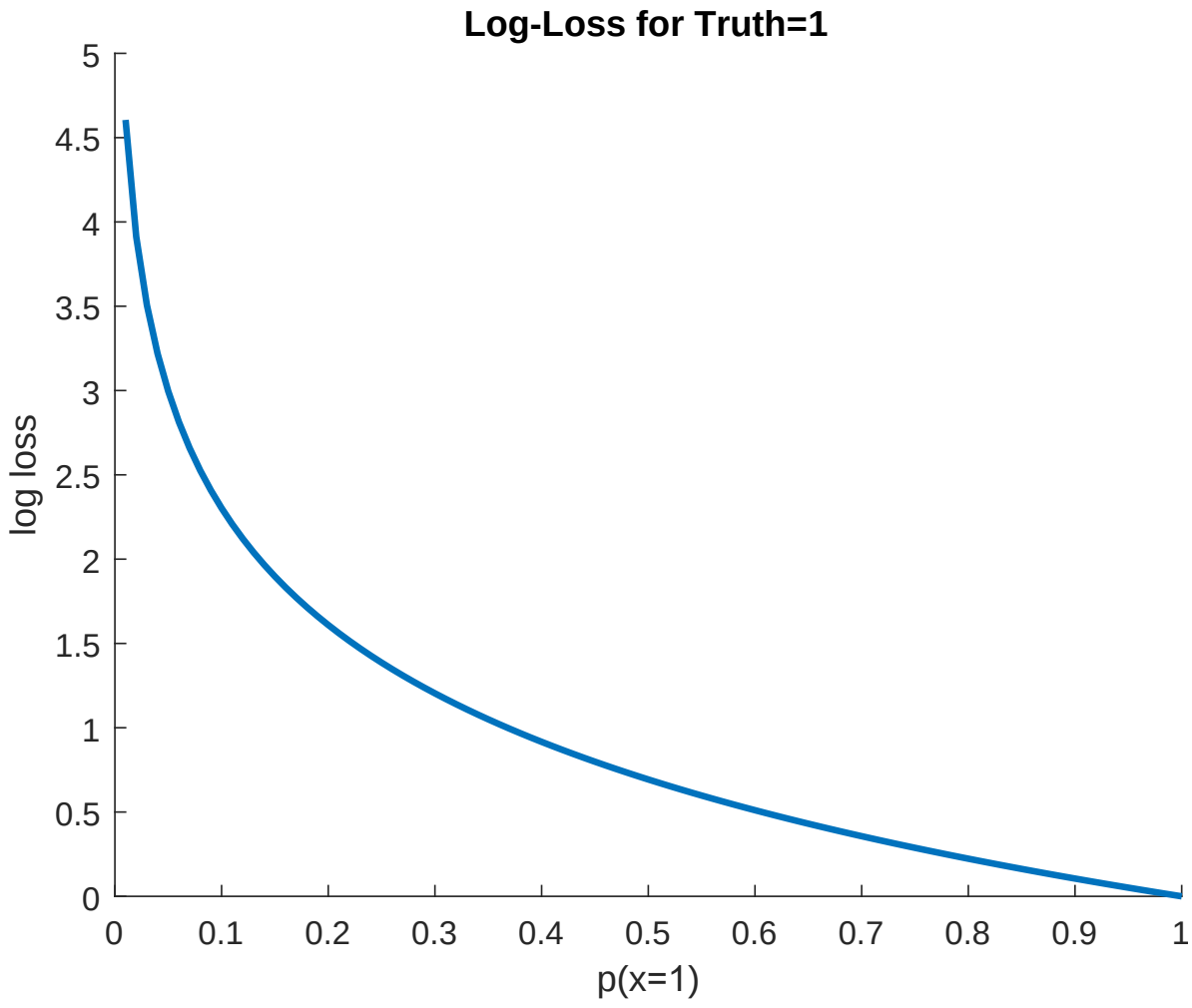
The **classifier** model returns probability or probability-like quantities for each category...

**Cross-Entropy loss** ("distance" between two distributions):

$$H(p, q) = -\sum_{x} p(x) \times \log q(x) \tag{3}$$

| obs. | red | green | blue | R | G | B | H(p,q) |
|---|---|---|---|---|---|---|---|
| #1 | 1 | 0 | 0 | 0.7 | 0.1 | 0.2 | 0.36 |
| #2 | 0 | 1 | 0 | 0.3 | 0.6 | 0.1 | 0.51 |
| #3 | 0 | 0 | 1 | 0.1 | 0.1 | 0.8 | 0.22 |
| ⋮ | | | | | | | ⋮ |
| #N | 0 | 1 | 0 | 0.3 | 0.3 | 0.4 | 1.20 |

# Example: Log-Loss Function Weights




Note: Misclassification rate accounts for errors with high probability with the same weight as for errors with low probability. . .

# Evaluating Classification Results

For classification, the loss function value can be hard to interpret...

**Common Classification Metrics:**

- Misclassification Rate
- Probability Calibration
- Confusion Matrix

- Sensitivity, Specificity, F1, ...
- Receiver Operating Characteristic (ROC) Curves
- ...

# Probability Calibration

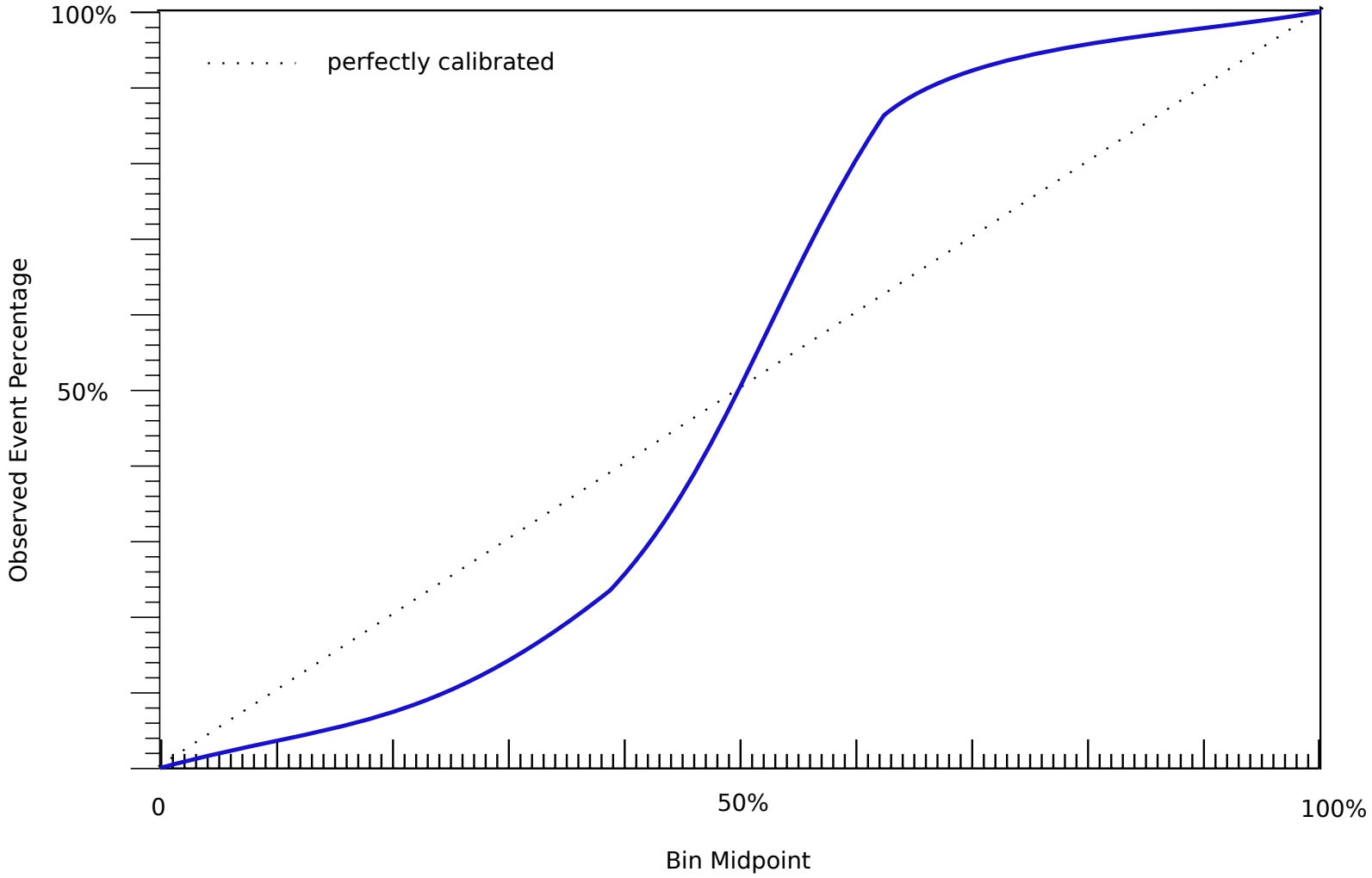
Are the probability-like numbers $p \in [0, 1]$ “true” probabilities?

**Well-Calibrated Probabilities**:
When the predicted class probability reflects the true likelihood of the event.

Essentially, out of all data points you assign 70% probability of being ‘Class A’, roughly 70% should turn out as ‘Class A’

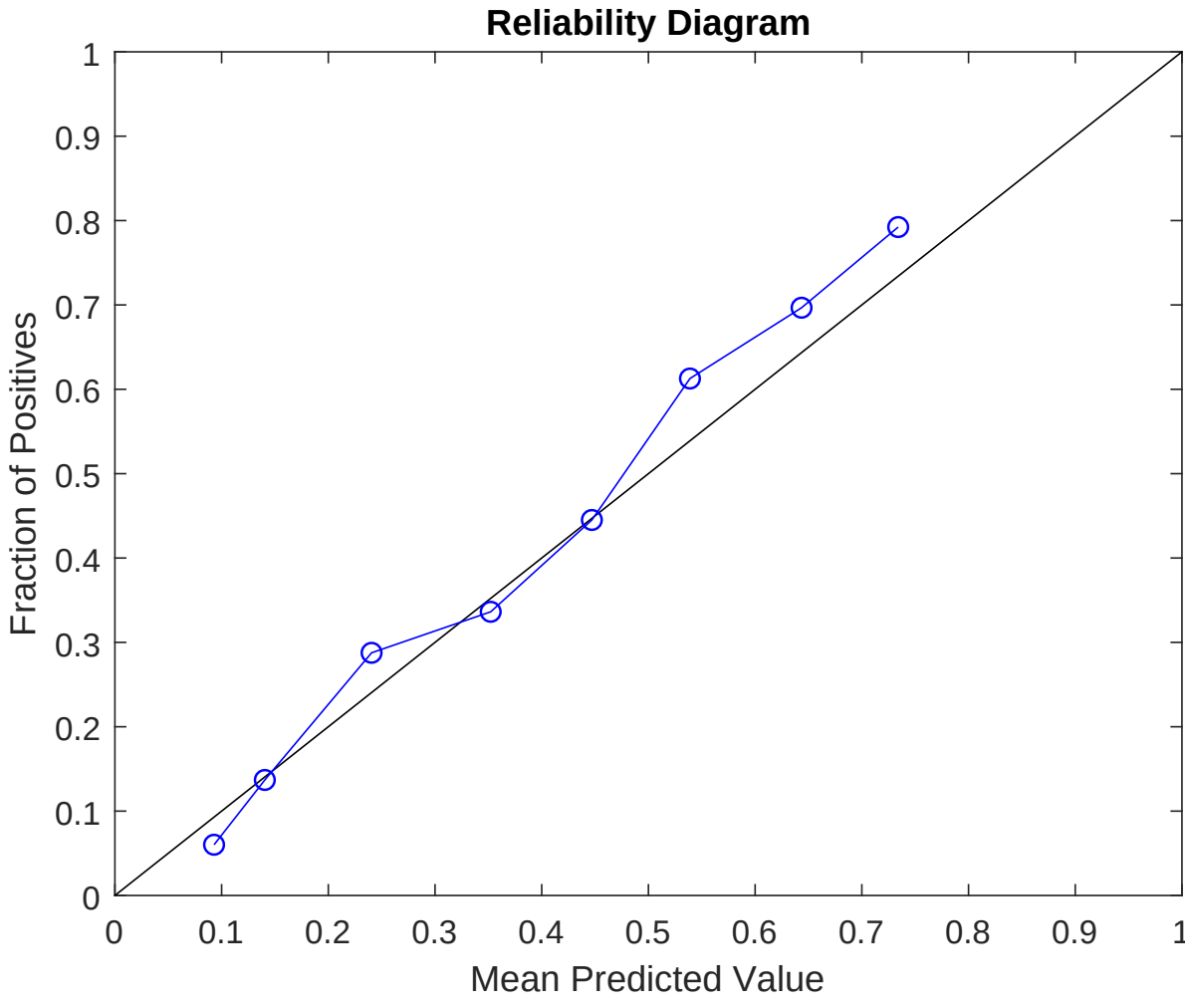
If probabilities are not well calibrated, they can be re-calibrated using a link model (e.g. Platt’s logistic regression, etc.)

# Probability Calibration Plot



# Probability Calibration Plot

Example: Calibration Plot for Credit-Card Default Model




Note: Forecast from RandomForests rarely reach 0 or 1 prob.

# Confusion Matrix – Two-Class Problem

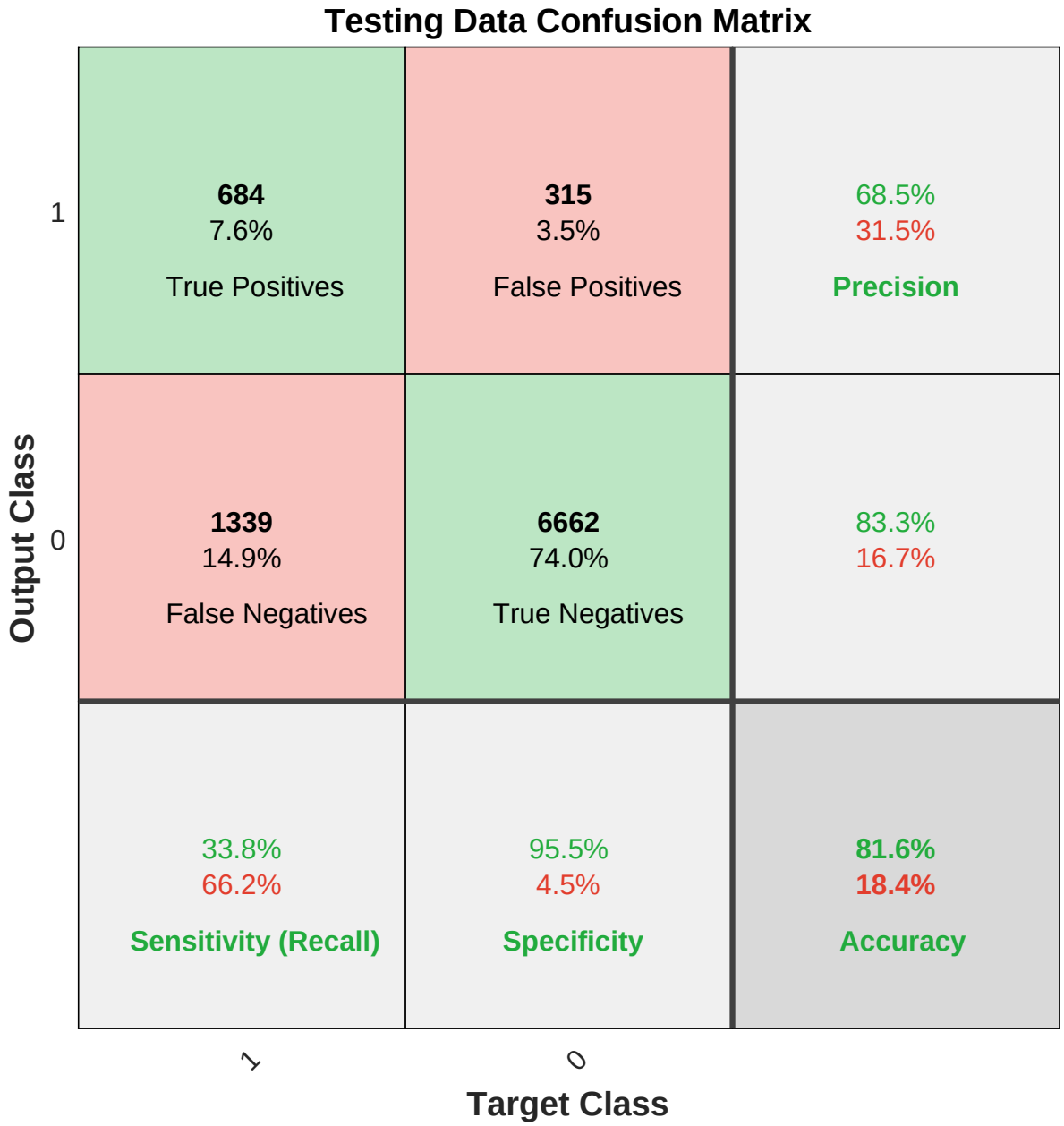
| **Predicted Class** | **Actual Class** | | |
|---|---|---|---|
| | Positive | Negative | |
| Positive | TP | FP | TP/(TP+FP) **precision** |
| Negative | FN | TN | TN/(FN+TN) neg. pred. value |
| | TP/(TP+FN) **recall, sensitivity** | TN/(FP+TN) **specificity** | (TP+TN)/(P+N) **accuracy** |

TP = True Positive
FP = False Positive = Type I Error
TN = True Negative
FN = False Negative = Type II Error

P = TP + FN, All Positive
N = TN + FP, All Negative

Confusion matrices are conditional on chosen classification **threshold**.

# Confusion Matrix for Credit Card Data Defaults



# Understanding Classification (1)

## Sensitivity = Recall = Hit Rate = True Positive Rate (TPR)

- TPR = TP/P = TP/(TP + FN)
- Out of all actual positives (TP + FN), how many were classified correctly?
- How good is the test at detecting positives?

## Specificity = Selectivity = True Negative Rate (TNR)

- TNR = TN/N = TN/(TN + FP)
- Out of all actual negatives (TN + FP), how many were correctly classified as negatives?
- How good at avoiding 'false alarms' (FPs)

## Precision

- Precision = TP/(TP + FP)
- Out of all 'positive' predictions, how many predictions are correct?
- How many positive predictions were *relevant*?

# Understanding Classification (2) – Summary

**Sensitivity = Recall = Hit Rate = True Positive Rate (TPR)**
TPR = TP/P = TP/(TP + FN)

**Specificity = Selectivity = True Negative Rate (TNR)**
TNR = TN/N = TN/(TN + FP)

**Precision = Positive Predictive Value (PPV)**
Precision = TP/ (TP + FP)

**False Discovery Rate**
FDR = FP/(TP + FP) = 1 - Precision

**Accuracy**
AC = (TP + TN)/(P + N) = # correct matches/population
**Prevalence**
PREV = P/(P + N) = # all positive/population

# Understanding Classification (2) – Examples

## Detecting Financial Crisis:

- Imbalanced sample – not all that many crises (say 3%)
- Never crying 'bear' yields high *accuracy* and zero precision...
- Seeing always a crisis yields high *recall* but very low precision (too many false alarms)

## Detecting Loan Default:

- Imbalanced sample – roughly 20% default on loan
- Assume a return on loan is 5% and default -95 %
- To maximize profit, you want to avoid permanent loss of capital, i.e. detect defaults well, risking losing some good business...

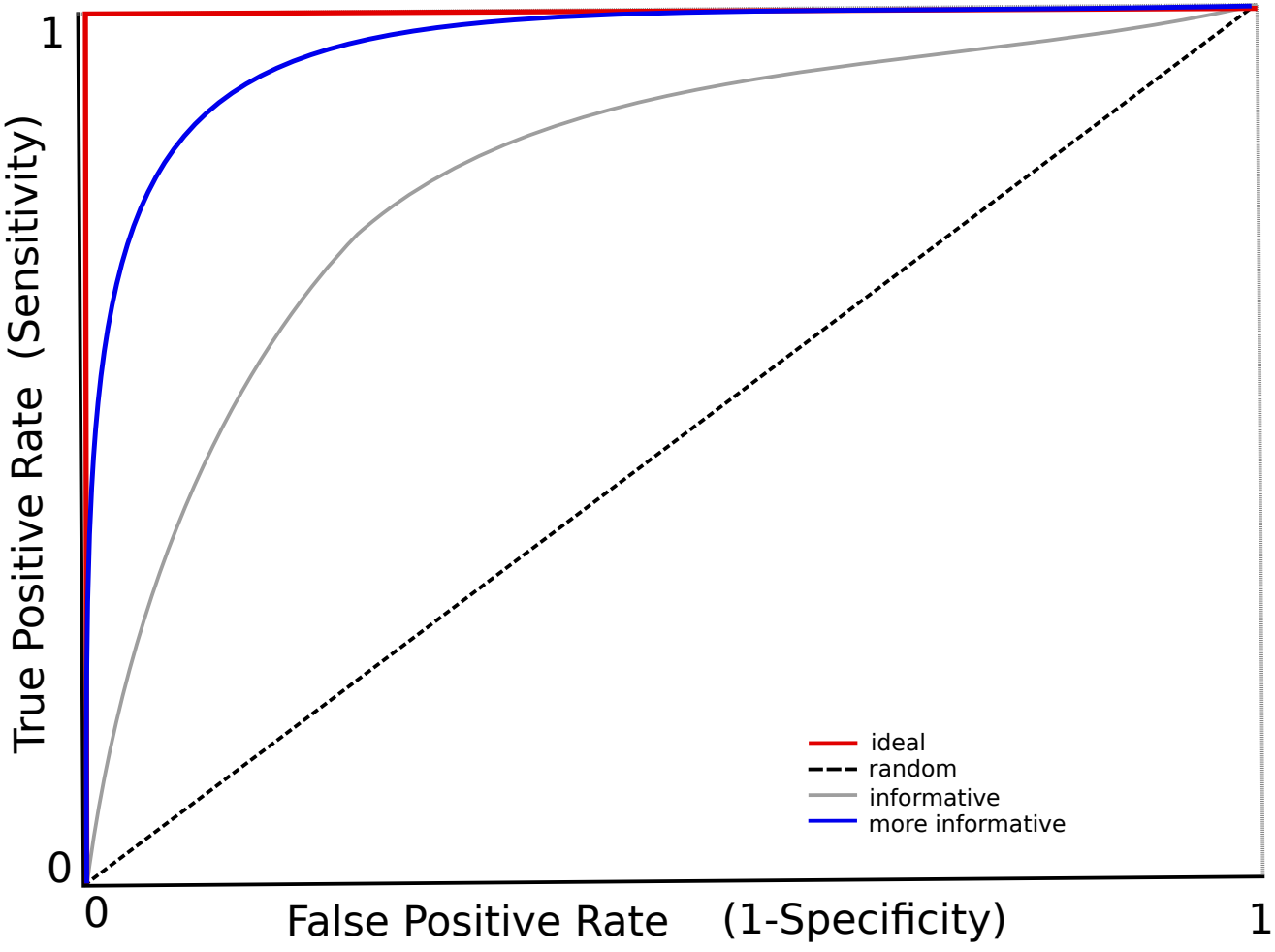
# Receiver Operating Characteristic (ROC) curve

ROC curve is a plot of true positive rate (TPR, **sensitivity**) against the false positive rate (FPR, **1-Specificity**) at various **threshold** values.

For fixed accuracy of the model, there usually is a trade-off between sensitivity and specificity.

ROC curves map this trade-off.

# Receiver Operating Characteristic (ROC) Curve



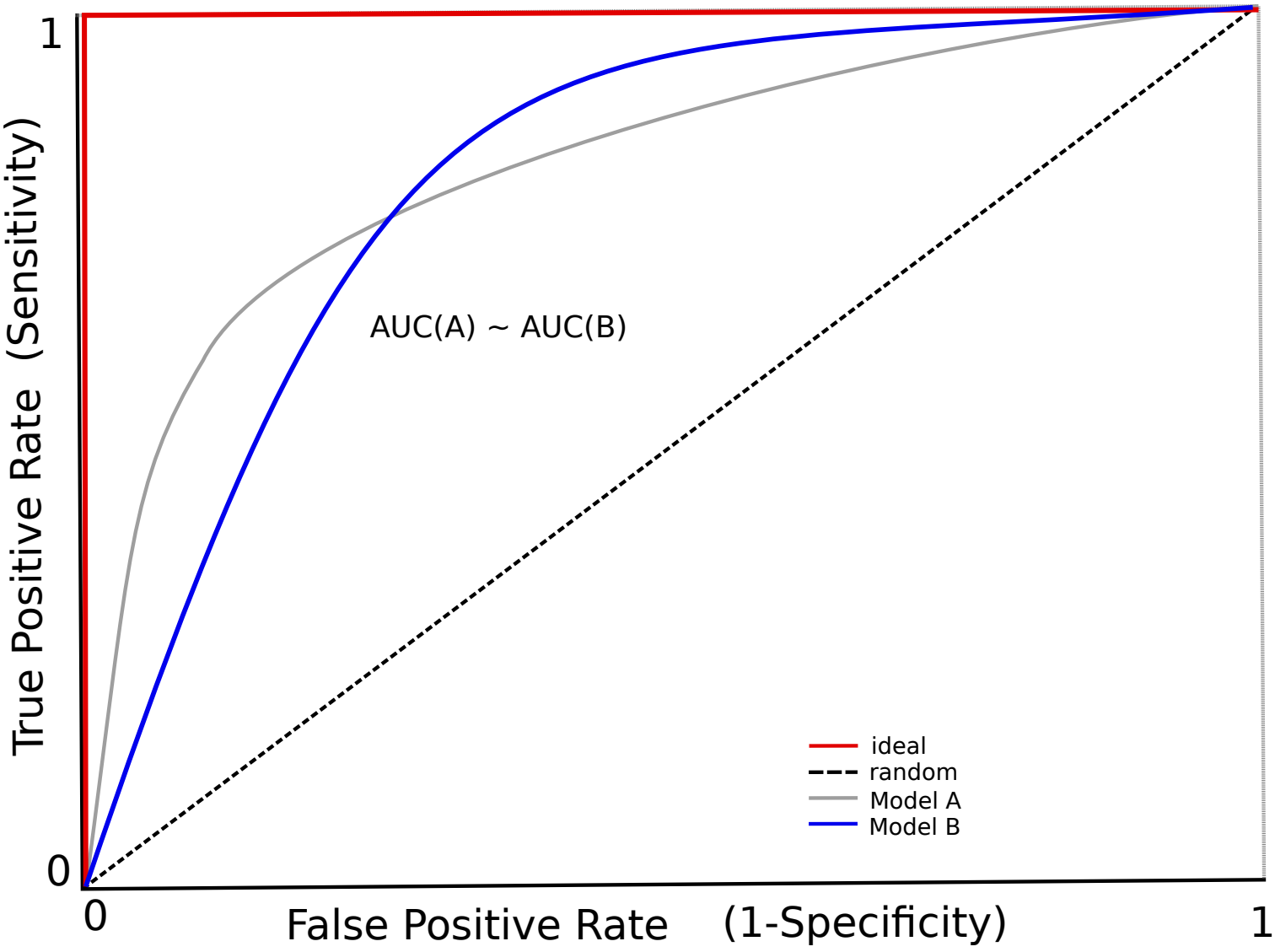

A perfect class separation would have 100% sensitivity and specificity.

# AUC – Area Under the [ROC] Curve

Often, the **area under the ROC curve** (AUC) is used as a summary measure for the ROC curve and model performance, model comparison

AUC may hide information as very different models can have identical AUC, i.e. ROC curves for different models may cross. . .

# AUC vs. ROC Curve

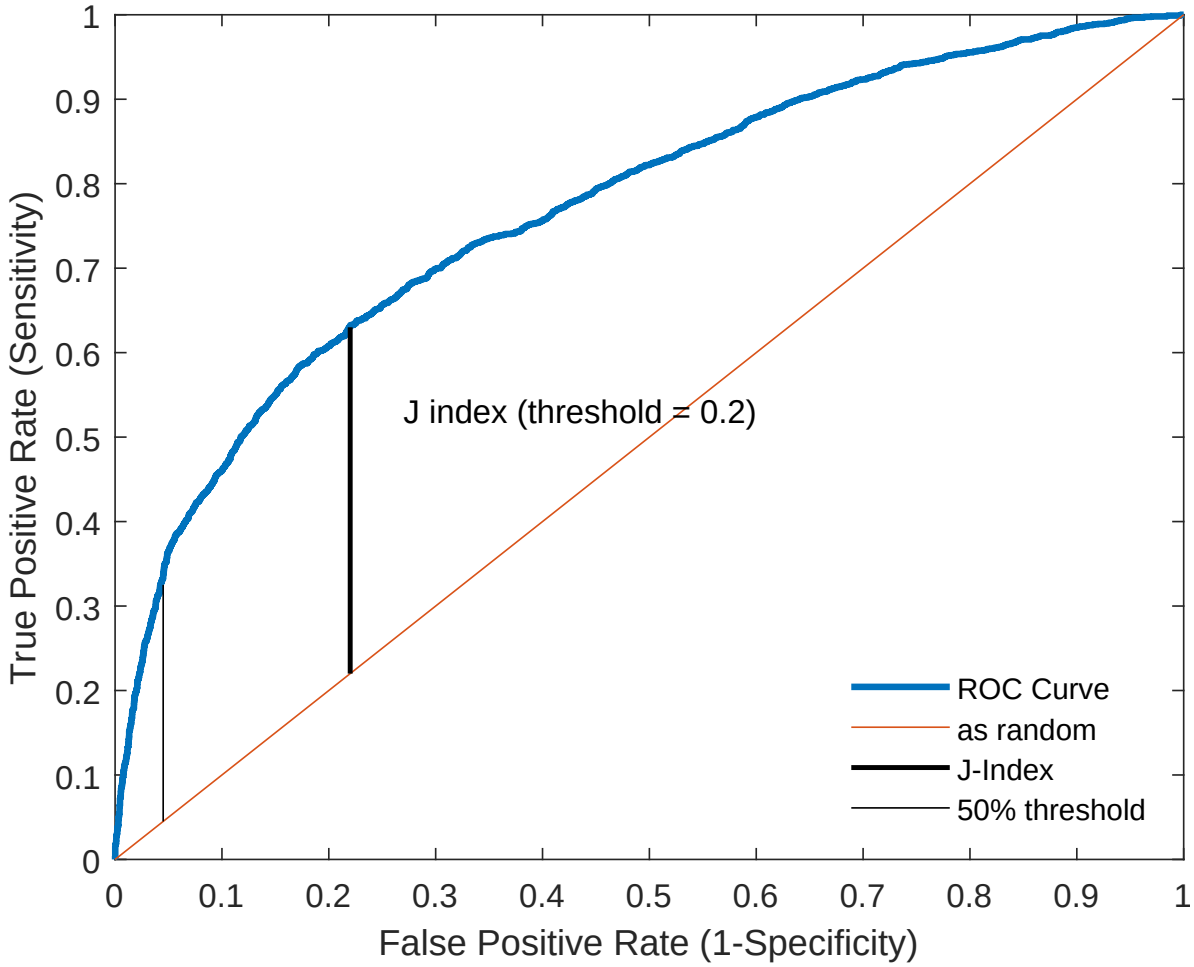


# Other Measures

- **F-score**
  - F = 2× Precision× Recall/(Precision + Recall)
  - Expresses trade-off between precision and recall
- **Youden's J-Index**
  - $J = \textit{Sensitivity} + \textit{Specificity} - 1$
  - Measures correctly indicated proportions for both classses
- Cohen's **Kappa**
  - $\kappa = \frac{\text{accuracy} - \text{expected accuracy}}{1 - \text{expected accuracy}} \in [-1, 1]$
  - Expected accuracy based on marginal totals in conf. matrix
  - If 90% of test are expected positive events, 89% accuracy is not amazing...
- **Non-Accuracy-Based Criteria!**
  - Ideally, the purpose helps to make it clear...
  - Expected profits, expected costs, Sharpe ratios...

# ROC Curve Analysis Example

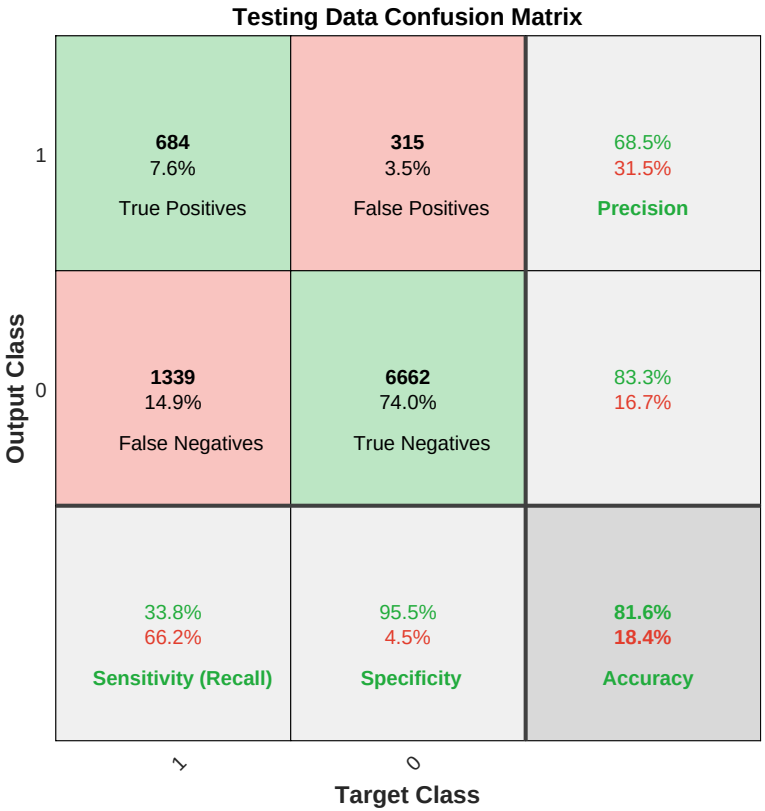
Using Youden's J-index we'll pick a new threshold on the same ROC curve. . .




| Threshold | Accuracy | Precision | Sensitivity | Specificity |
|---|---|---|---|---|
| 0.5 | 0.82 | 0.69 | 0.34 | 0.95 |
| 0.2 | 0.75 | 0.45 | 0.63 | 0.78 |

# Changing Threshold: Classification Trade-Offs

## ‘Traditional’ 50% cut-off



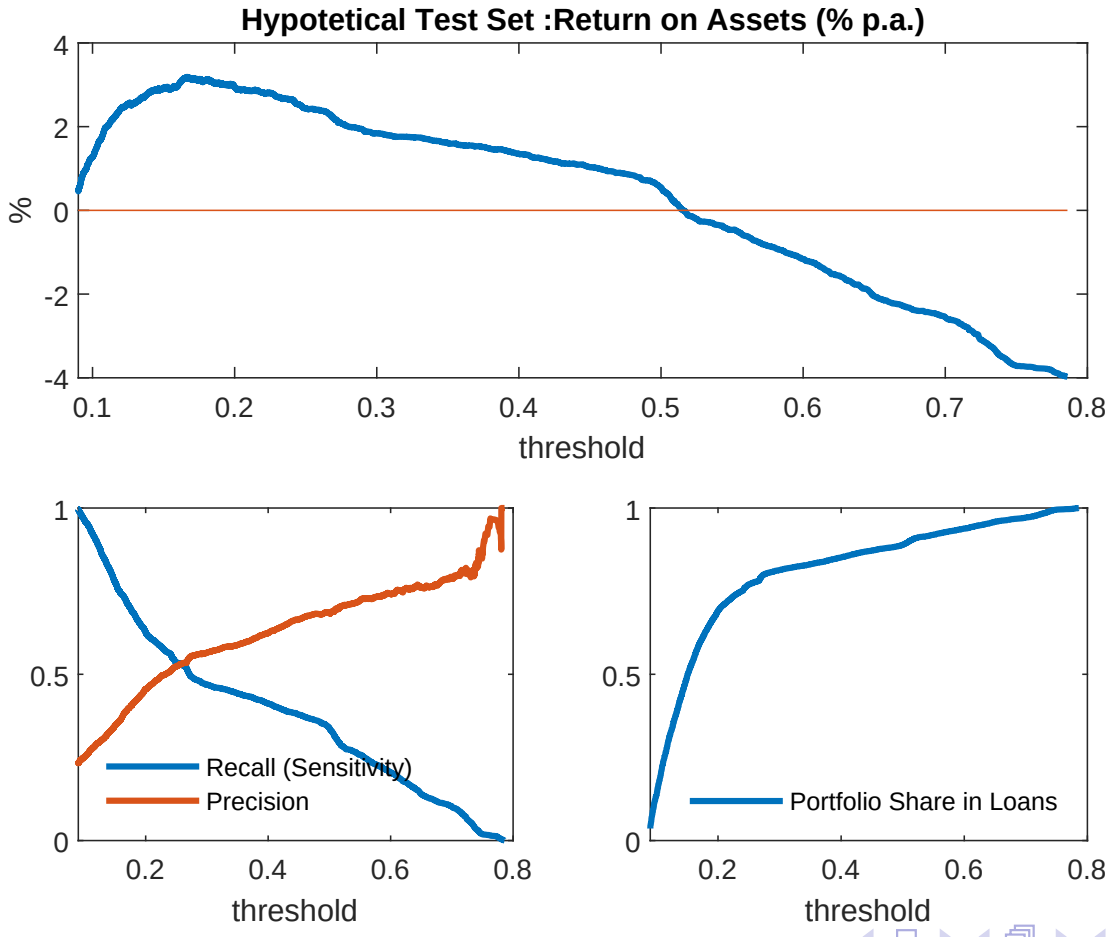

## Alternative 20% cut-off


In reality, the choice for the threshold would depend on the cost of misclassification error.

# Changing Threshold: Hypothetical RoA

Imagine you have a portfolio of USD $N \times \$100$ and can make \$100 loans. Each loan's cost are 1%. If you don't make the loan, you can invest for riskless rate of 1%. If loan is made and successful the return is 15%. If the loan defaults, loss-given-default is 70%. If everybody gets a loan, 22% of borrowers default on it.

So. . . recall or precission? Do you recall?



# Wonkish: Softmax vs. Cross-Entropy

Softmax 'squashing' function and cross entropy are not the same thing.

Since the "truth" is a mass point, i.e. $p_i$ is either 1 or 0, the log of softmax is the cross-entropy.

$$H(p, q) = -(0 \times \log q_1 + 1 \times \log q_2 + 0 \times \log q_3 + \dots) \tag{4}$$

$$H(p, q) = -\log q_2 = -(x_2 - \log \sum_j \exp(x_j)) \tag{5}$$

Hence, oftentimes, classification people talk about softmax and cross entropy as about the same thing...
and it can be puzzling[1]

[1] or maybe it's just me

# Wonkish: Softmax and Cross-Entropy vs. Logistic Regression

Logistic regression is a special case of two classes only ($K = 2$).

$$q_1(x_i) = \frac{exp(\theta_1' x_i)}{exp(\theta_1' x_i)+exp(\theta_2' x_i)} \quad q_2(x_i) = \frac{exp(\theta_2' x_i)}{exp(\theta_1' x_i)+exp(\theta_2' x_i)}$$

'Naive' K-level softmax is overparameterized, only $K - 1$ degrees of freedom. Normalize one group, and define $\psi' = \theta_1' - \theta_2'$.

$$q_2(x_i) = \frac{exp(0_i)}{exp(\psi' x_i)+exp(0' \times x_i)} = \frac{1}{exp(\psi' x_i)+1}$$
$$q_1(x_i) = \frac{exp(\psi' x_i)}{exp(\psi' x_i)+exp(0' \times x_i)} = \frac{exp(\psi' x_i)}{exp(\psi' x_i)+1} = 1 - \frac{1}{exp(\psi' x_i)+1} = 1 - q_2(x_i)$$

Now, the *loss function* defined via cross-entropy for $N$ observations:

$$J(\psi) = \sum_i^N H(\hat{y}_i, q(\phi, x_i)) = \sum_i^N [\hat{y}_i \log q_2(x_i) + (1 - \hat{y}_i) \log(1 - q_2(x_i)] \quad (6)$$

**Thank you for your patience. . .**