

Local expert regression & calibration for probabilistic classification

Nick Normandin

This talk

- ▶ overview of a different kind of ensemble method for supervised learning problems ('local expert regression')
- ▶ discussion of calibration method for probabilistic classifiers, and why it's important

Local expert

Ensemble models recap

- ▶ many models better than single model, especially when ensemble constituents have low correlation
- ▶ independent methods usually involve resampling (eg: bagging, random forests)
- ▶ dependent methods usually involve reweighting instances (eg: boosting)

What is local expert?

It involves the decomposition of a supervised learning task with a continuous target variable (*regression*) into a series of many $\{0, 1\}$ mappings corresponding to separate *binary probabilistic classification* tasks that produce estimates on the $[0, 1]$ interval.

Why is this useful ?!

Because you can aggregate the ensemble predictions to form a completely unique *probability distribution function* for each prediction. You can understand **risk** not just in terms of a model, but in terms of each individual forecast.

... see github.com/nnormandin/localexpert

How is local expert different from normal regression?

weather data vector



what will the high
temperature be
tomorrow?

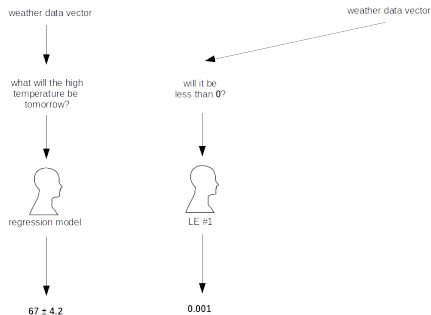


regression model

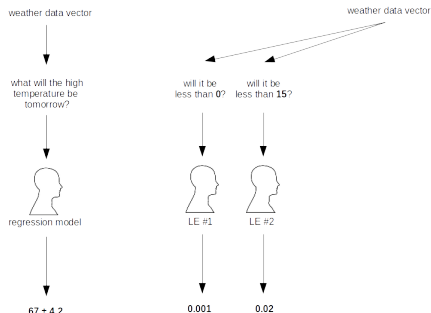


67 ± 4.2

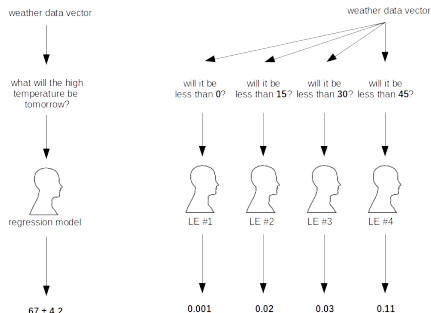
How is local expert different from normal regression?



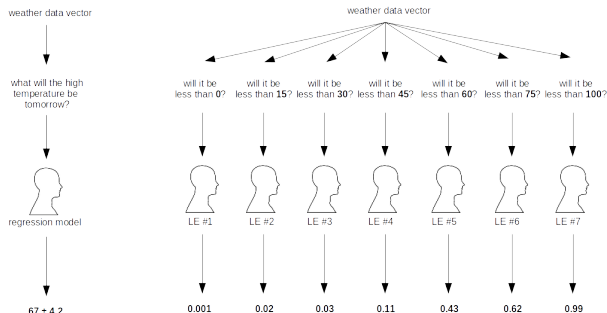
How is local expert different from normal regression?



How is local expert different from normal regression?



How is local expert different from normal regression?



How is local expert different from normal regression?

- ▶ The local expert predictions can be reconstructed to form an empirical CDF
- ▶ By fitting a spline to these points and then differentiating, we can create a unique PDF for each prediction
- ▶ We can forecast \hat{y} based on the moments of the distribution, or fit another meta-model on the output of the local expert ensemble (see Wolpert's work re: stacked generalization)
- ▶ **BUT**, all of these cool things we could do are dependent on having well-calibrated probability scores

Formally...

1. **Input:** labeled data $\{\mathbf{X}, \vec{y} \in \mathbb{R}\}$, probabilistic classifier \mathcal{L} , and stacked regressor \mathcal{L}^*
2. select N values $\{v_1, v_i, \dots, v_N\}$ throughout the range of observed \vec{y} values
3. **for** each value v_i :
 - 3.1 define $\mathbf{1}_{v_i} := \begin{cases} 0, & \text{if } y \leq v_i \\ 1, & \text{otherwise} \end{cases}$
 - 3.2 induce \mathcal{L}_{v_i} on $\{\mathbf{X}, \mathbf{1}_{v_i}(\vec{y})\}$
 - 3.3 **return** local expert $\mathcal{L}_{v_i} : p_i \in [0, 1]$
4. using cross-validation, for each \vec{x} in \mathbf{X} construct local expert predicted probability vector $\vec{p} = \{\mathcal{L}_1(\vec{x}), \mathcal{L}_i(\vec{x}), \dots, \mathcal{L}_N(\vec{x})\}$
5. for each probability vector \vec{p} , fit smoothing spline and differentiate smoothed function. **return** mean, standard deviation, and skewness of PDF
6. induce regressor \mathcal{L}^* on $\{\mathbf{P}, \vec{y}\}$. **return** stacked prediction model $\mathcal{L} : \vec{x} \rightarrow \mathbb{R}$

Probability calibration

The problem

some classifiers produce
well-calibrated probabilities

- ▶ discriminant analysis
- ▶ logistic regression

others don't

- ▶ naive bayes
- ▶ SVMs
- ▶ anything with boosting
- ▶ tree methods
- ▶ sometimes neural networks

First of all, who cares?

1. people with asymmetric misclassification costs
2. people who are going to use the scores in post-processing
3. people who want to compare model outputs on a fair basis

Definitions: “classification”

in general, a classifier is a mapping function f such that

$$f : \vec{x} \mapsto c$$

where $\vec{x} \in \mathbb{R}^P$, but we're mostly interested in the intermediate step in where the function produces some membership score s_i for each instance \vec{x}_i

Definitions: “well-calibrated”

- ▶ for a model f and score s_i to be well-calibrated for class c_i , the empirical probability of a correct classification $P(c_i | f(c_i | x_i) = s_i)$ must converge to $f(c_i | x_i) = s_i$
- ▶ **example:** when $s_i = 0.9$, the probability of a correct classification should converge to $P(c_i | s_i = 0.9) = 0.9$. Otherwise, this isn't *really* a ‘probability.’

Definitions: “calibration”

the calibration process is a separate mapping such that

$$g : s_i \mapsto P(c_i | s_i)$$

it's really important to note that we're fitting another model on top of our model output, where your feature matrix is just the vector of probability scores \vec{s} and the target variable is the vector of true class labels $\vec{y} \in \{0, 1\}$

Visualizing calibration

Visualizing calibration

```
# train SVM w/ linear kernel on Pima Indian Diabetes data
m <- train(x = PimaIndiansDiabetes[,1:8],
           y = PimaIndiansDiabetes[,9], tuneLength = 1,
           method = 'svmLinear',
           trControl = trainControl(method = 'cv',
                                     savePredictions = T,
                                     classProbs = T))

pred <- m$pred[order(m$pred$rowIndex),]
result <- data.table(prob = pred$pos,
                    class = ifelse(pred$obs == 'pos', 1, 0))
```

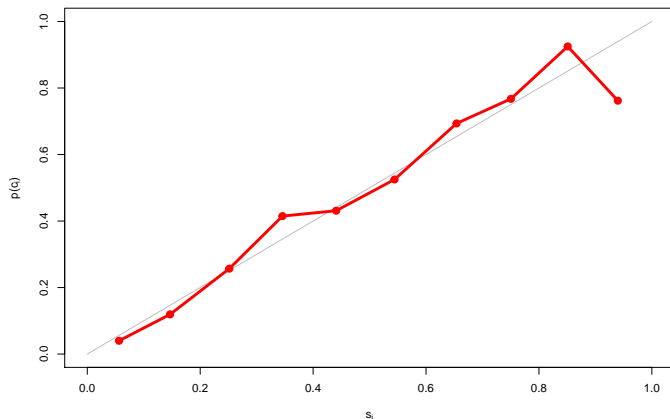
Visualizing calibration

plotting the model class score s_i vs the true label y_i . Is this a useful representation?



Reliability plots

(1) Bin predictions by s_i (x-axis), **(2)** calculate $p(c_i)$ by bin (y-axis)



Common methods

Platt scaling

pass s_i through the sigmoid

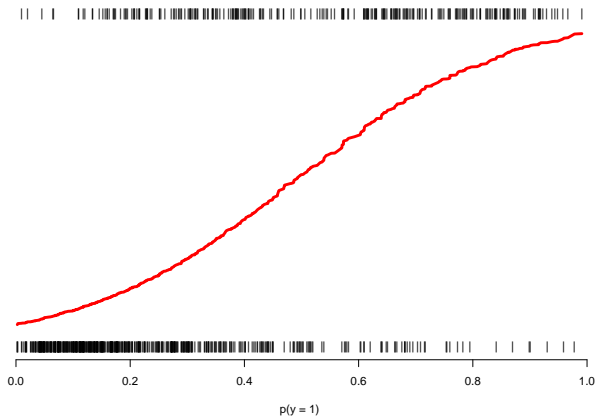
$$P(c_i | s_i) = \frac{1}{1 + \exp(As_i + B)}$$

where A and B are the solution to

$$\operatorname{argmax}_{A,B} - \sum_i y_i \log(p_i) + (1 - y_i) \log(1 - p_i)$$

Platt scaling

applied to the Pima Indian Diabetes scores



Isotonic regression

a strictly-nondecreasing piecewise linear function m , where

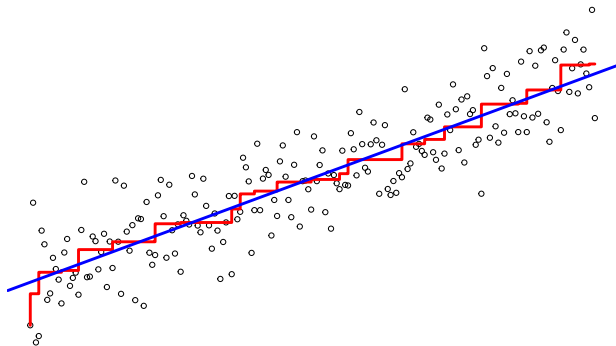
$$y_i = m(s_i) + \epsilon$$

fit such that

$$\hat{m} = \operatorname{argmin}_z \sum_i (y_i - z(s_i))^2$$

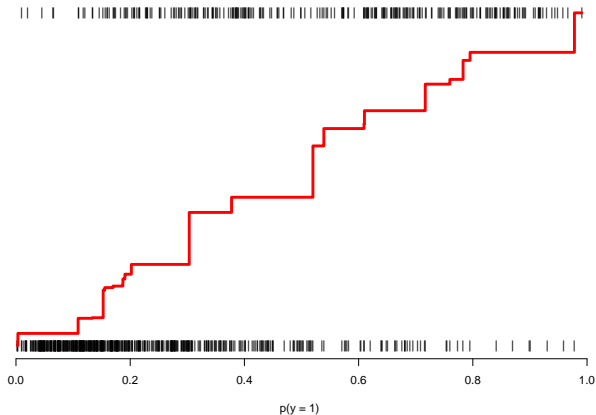
Isotonic regression

linear and isotonic regression fit to random noise with drift



Isotonic regression

applied to the Pima Indian Diabetes scores



Notes on applying calibration

- ▶ it's really easy to overfit
 - ▶ calibration partition
 - ▶ cross-validation
- ▶ isotonic regression is generally more flexible (and can closely approximate sigmoid)
- ▶ best technique is dependent on the family of model used to generate s_i

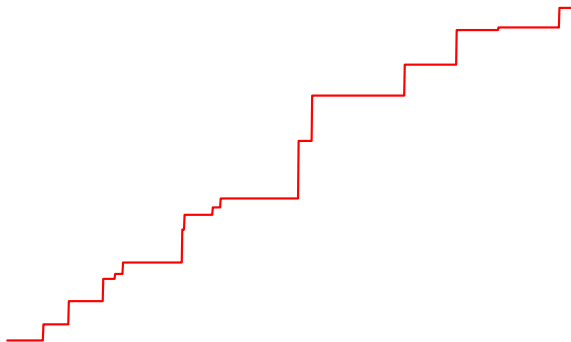
Bootstrap aggregated isotonic regression

1. sample (w/ replacement) some proportion of data
2. fit isotonic regression to sampled data
3. repeat steps 1-2 N times
4. return average prediction from all functions

note: the average of many piecewise linear monotonically non-decreasing functions is still a piecewise linear monotonically non-decreasing function

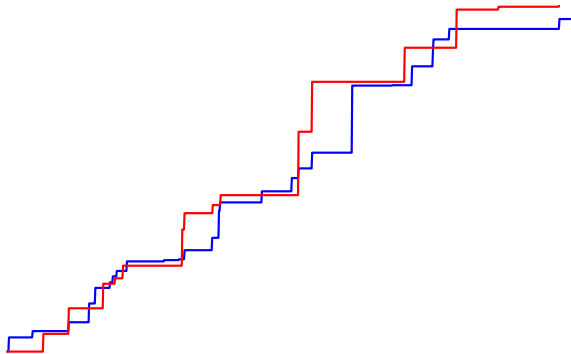
Bootstrap aggregated isotonic regression

original isotonic regression fit



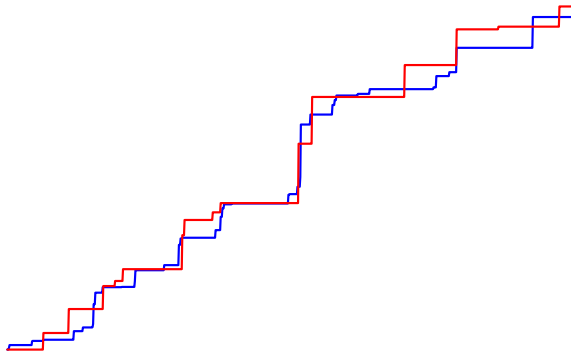
Bootstrap aggregated isotonic regression

make it smoother by aggregating and averaging over 3 resampled isotonic regression fits



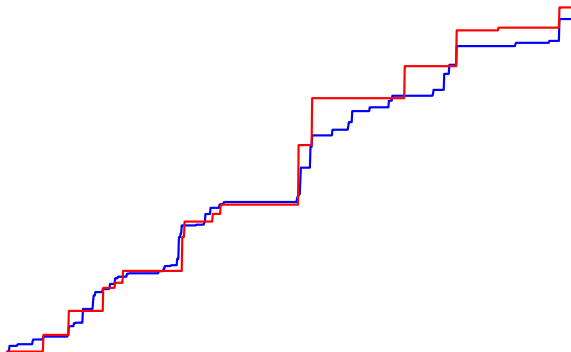
Bootstrap aggregated isotonic regression

make it smoother by aggregating and averaging over 5 resampled isotonic regression fits



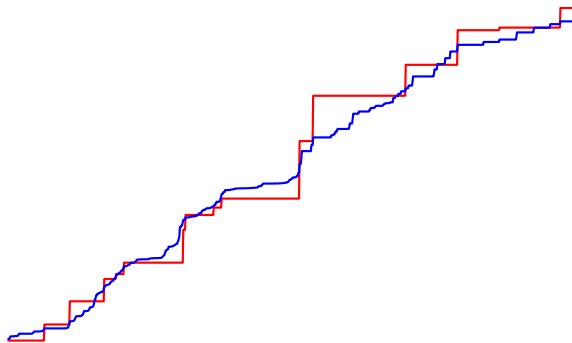
Bootstrap aggregated isotonic regression

make it smoother by aggregating and averaging over 10 resampled isotonic regression fits



Bootstrap aggregated isotonic regression

make it smoother by aggregating and averaging over 10,000 resampled isotonic regression fits



When $k > 2$

Probabilistic classification as a simplex

- ▶ if we view the task of probabilistic classification as a vector-valued function, we can visualize the co-domain of this task as assigning the location of a prediction in a regular (unit) simplex, Δ^{K-1}
- ▶ why is this hard when $K > 2$?

Probabilistic classification as a simplex

 Δ^1  Δ^2 

trivial with Δ^1 because we're only concerned with one unknown value and its complement. With $\Delta^{K>2}$ the simplex becomes a triangle, tetrahedron, five-cell, etc.

Multi-class probability estimation

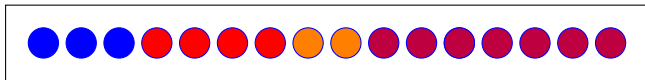


Figure 1: classification problem with $k = 4$

Strategy: decompose into separate binary classification problems

- ▶ one vs. all
- ▶ all pairs

One vs. all

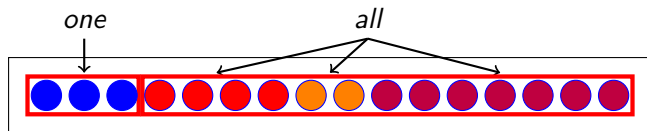


Figure 2: *one* vs. *all* reduces to $k - 1$ calibrations

All pairs

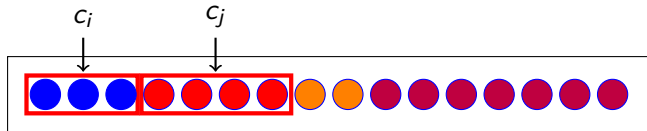


Figure 3: *all pairs* reduces to $\binom{K}{2}$ calibrations

Combining multi-class probability estimates

- ▶ least squares: minimize squared error loss w/ non-negativity
- ▶ coupling (only *all pairs*): minimize log loss w/ non-negativity
- ▶ normalization (only *one vs all*): divide by sum of probabilities estimates

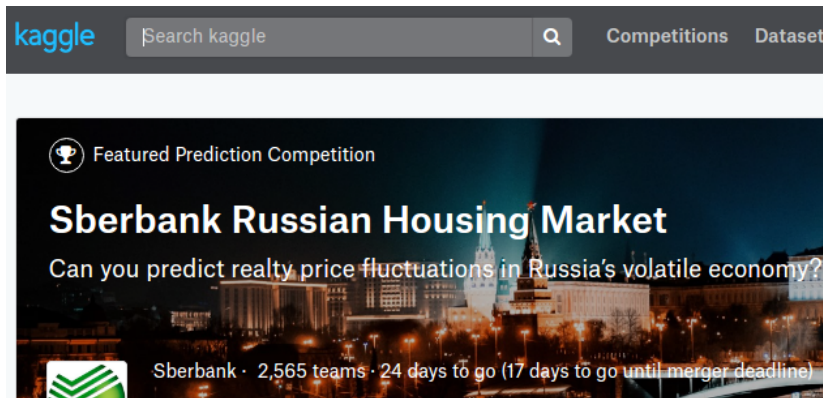
Exerimental results

- ▶ **Multivariate calibration of classifier scores into the probability space** by Martin Gebel
- ▶ **Transforming Classifier Scores into Accurate Multiclass Probability Estimates** by Zadrozny and Elkan
- ▶ **Obtaining Calibrated Probabilities from Boosting** by Niculescu-Mizil and Caruana
- ▶ **Predicting Good Probabilities With Supervised Learning** by Niculescu-Mizil and Caruana

Case study

The task

- ▶ 30,000 records of 300+ variables related to Russian housing transactions
- ▶ very dirty, lots of multicollinearity



The image is a screenshot of the Kaggle website's header and a featured competition banner. The header is dark grey with the 'kaggle' logo in blue and white on the left. To its right is a search bar with the placeholder text 'Search kaggle' and a magnifying glass icon. Further right are the links 'Competitions' and 'Dataset'. Below the header is a large banner for the 'Sberbank Russian Housing Market' competition. The banner has a dark background with a night-time photograph of the Moscow skyline, featuring the illuminated Spasskaya Tower of the Moscow Kremlin. The text on the banner includes a trophy icon and the words 'Featured Prediction Competition' in orange. The main title 'Sberbank Russian Housing Market' is in large white font. Below it, a question is posed: 'Can you predict realty price fluctuations in Russia's volatile economy?'. At the bottom left of the banner is the Sberbank logo (a green stylized 'S' inside a white square). At the bottom right, the text 'Sberbank · 2,565 teams · 24 days to go (17 days to go until merger deadline)' is displayed in white.

Preparing the data

I cleaned and pre-processed separately, so we'll just read in those files and partition the train and test sets.

```
X <- readRDS('./dataX')
y <- readRDS('./dataY')

trainset <- createDataPartition(y, p = 0.6)[[1]]

Xtrain <- X[trainset,]; Xtest <- X[-trainset,]
ytrain <- y[trainset]; ytest <- y[-trainset]
```

COA 0: Tune and train a regression model

```
mod0 <- train(x = Xtrain, y = ytrain, method = 'gbm',  
              tuneLength = 5, trControl = tc,  
              verbose = FALSE)  
  
mod0_preds <- ExtractBestPreds(mod0)  
mod0_preds[mod0_preds < 0] <- 0
```


COA 1A: Use a local expert ensemble

Map the continuous y vector into a binary matrix

```
yb <- BinCols(ytrain, n = 25, mode = 'EP')
```

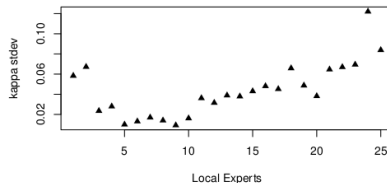
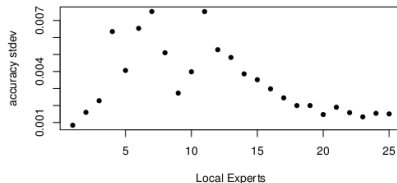
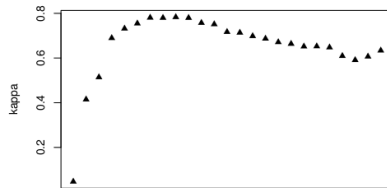
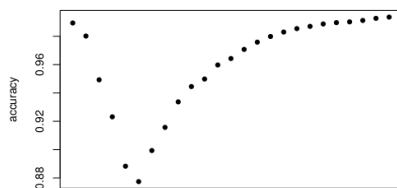
Induce separate models across each column in the matrix

```
LEs <- TrainLEs(x = Xtrain, bincols = yb$cols, trControl = tc,  
               method = 'gbm', n.folds = 8, tuneLength = 8,  
               verbose = FALSE)
```

```
LE_info <- ExtractModelInfo(LEs)
```

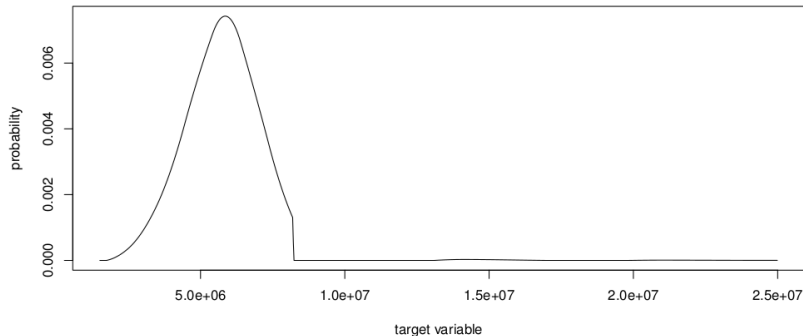
COA 1A: Use a local expert ensemble

```
PlotLEs(LE_info)
```



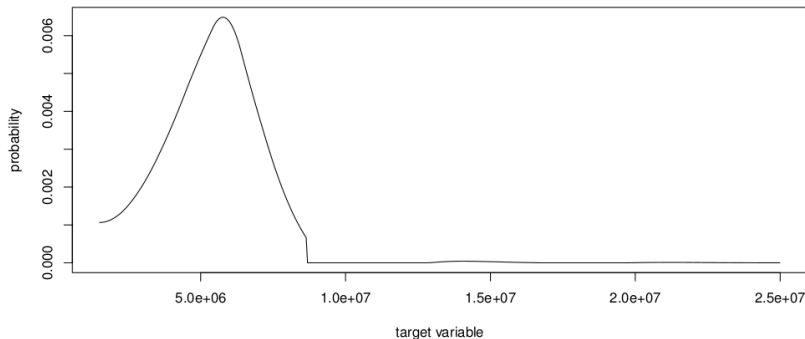
COA 1A: Use a local expert ensemble

Now for each separate instance, we're predicting a distribution instead of a value:



COA 1A: Use a local expert ensemble

Now for each separate instance, we're predicting a distribution instead of a value:



COA 1A: Use a local expert ensemble

Predict \hat{y} using just the means of the empirical distributions

```
LE_fits <- FitMatrix(LE_info$preds.matrix, yb$y.vals)
plot(x = log(1+ytrain), y = log(1+LE_fits$mean))
```

COA 1B:

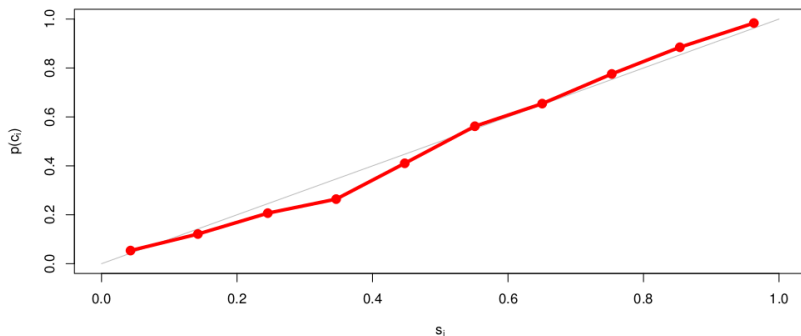
Fit a stacked meta-model on the local expert layer output (include distribution mean as additional meta-feature).

```
meta_X1 <- pm
meta_X1$mean <- LE_fits$mean

mod_1B <- train(x= meta_X1, y = ytrain, method = 'gbm',
                tuneLength = 10, trControl = tc1,
                verbose = FALSE)
```

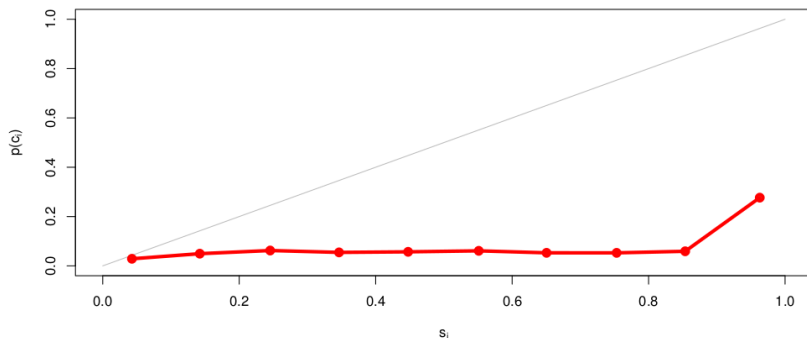
COA 2A: Local expert with calibrated probabilities

The output of each local expert must be calibrated independently. Each has a distinct reliability plot, and class imbalance has a substantial effect on the probability scores generated.



COA 2A: Local expert with calibrated probabilities

The output of each local expert must be calibrated independently. Each has a distinct reliability plot, and class imbalance has a substantial effect on the probability scores generated.



COA 2A: Local expert with calibrated probabilities

Instead of simply fitting an isotonic regression model to our local expert output, we can remove some of the bumpiness of the step function and reduce overfitting by bootstrapping. The steps are:

1. Draw random sample with replacement
2. Fit isotonic step function using PAVA algorithm
3. Repeat steps 1 & 2 for 500 iterations
4. Average all step functions
5. Return final 'step function' (still monotonic)

COA 2B: Stacked model with calibrated probabilities

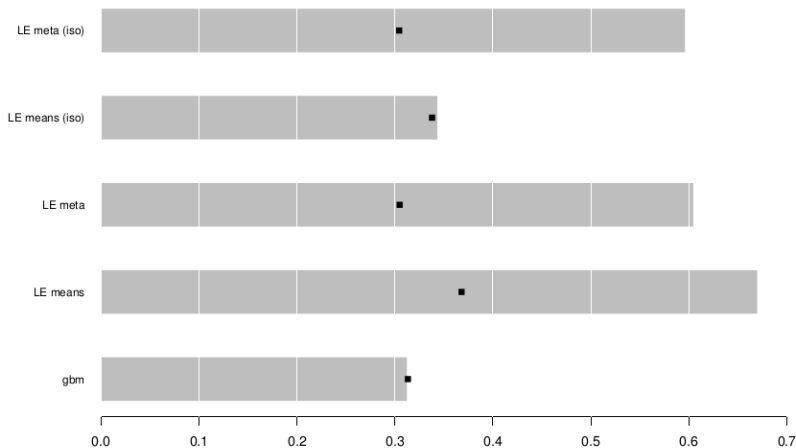
Fit a stacked meta-model on the local expert layer output (include distribution mean as additional meta-feature).

```
meta_X2 <- pm2
meta_X2$mean <- LE_fits2$mean

mod_2B <- train(x= meta_X2, y = ytrain, method = 'gbm',
                tuneLength = 8, trControl = tc1,
                verbose = FALSE)
```

Results

Model CV and out-of-sample RMLSE



Notes on results

- ▶ probabilities corrected with isotonic regression dramatically outperformed uncorrected probabilities in terms of both in/out of sample error
- ▶ with a stacked (meta) model, the performance benefits of isotonic regression are erased (makes sense, since model learns its own interpretation of data)
- ▶ the stacked local expert method outperforms the popular Gradient Boosting Machines both with/without isotonic regression in terms of out of sample test performance

Questions