Classification

Carlos Carvalho, Mladen Kolar, Robert McCulloch

1. Classification
2. The Lift
3. Loss Functions
4. Decision Theory Expected Utility

1. Classification

When the Y we are trying to predict is a categorical variable, we say we have a *classification* problem.

In a classification problem, each Y belongs to one of C categories or levels.

Often we label the categories with integers: $Y \in \{1, 2, \ldots, C\}$.

A very large number of important problems are binary classification where $C = 2$ so that we are trying to guess which of two possible outcomes will occur.

- Will the account default?
- Will the customer leave (churn) ?
- Will the customer buy (target marketing) ?

In binary problems, we often label Y with 0 and 1, $Y \in \{0, 1\}$.

While the big ideas (such as the bias-variance tradeoff) are the same for predicting a categorical variable and a numeric variable, some of the details of our modeling are neccessarily different.

kNN:

Given test x and training (x_i, y_i):

Numeric Y:

- find the k training observations with x_i closest to x.
- predict y with the average of the y values for the neighbors.

Categorical Y:

- find the k training observations with x_i closest to x.
- predict y with the most frequent of the y values for the neighbors.
- estimate $P(Y = i \mid x)$ with the proportion on neighbors having $y = i$.

Random Forests:

Given test x and training (x_i, y_i):

Numeric Y:

- Build B big trees.
- predict y with the average of the predictions from the B trees.

Categorical Y:

- Build B big trees.
- predict y with most frequently predicted value from the B trees.
  We let the trees *vote*.
- estimate $P(Y = i \mid x)$ with the average of the estimates from the B trees.

Boosting:

We use the same idea.

Iteratively:

(i) Given the current fit, fit "what is left over".

(ii) Add a crushed version of the fit from the first step into the overall fit.

For numeric Y, "what is left over" is the residuals.

For categorical Y we skip the details.

Regression:

For example, we can use the idea of a linear function

$$f(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p = x' \beta$$

for numeric Y and categorical Y.

Numeric Y:

$$Y = x'\beta + \epsilon$$

Categorical binary Y:

$$P(Y = 1 \mid x) = F(x' \beta), \quad F(a) = \frac{\exp(a)}{1 + \exp(a)}$$

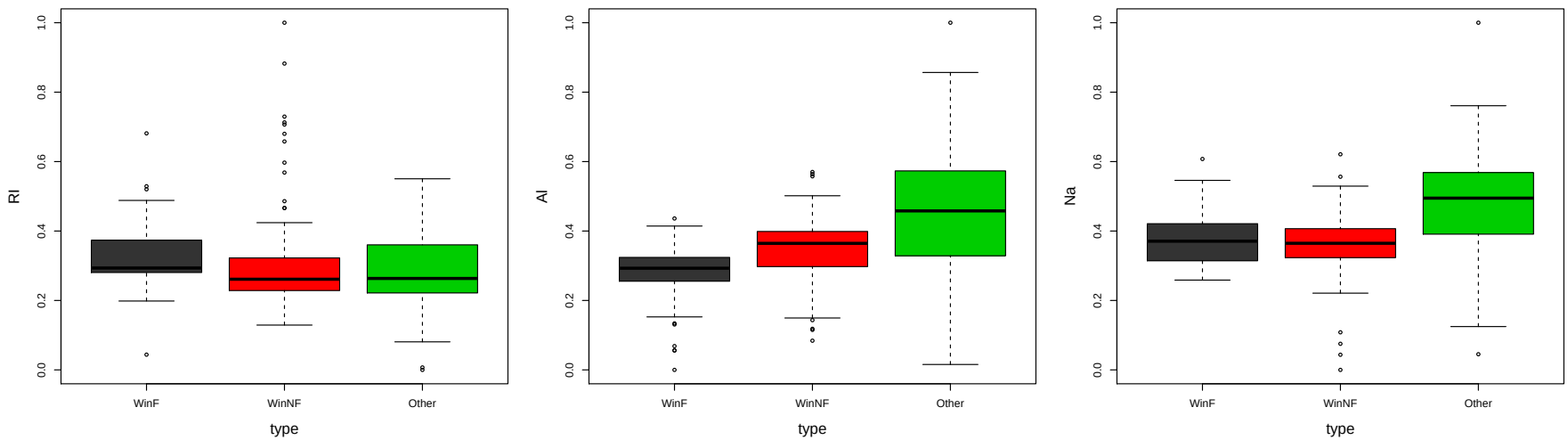
In both cases, x only affects Y through a linear combination but our specification of $Y \mid X = x$ must be different.

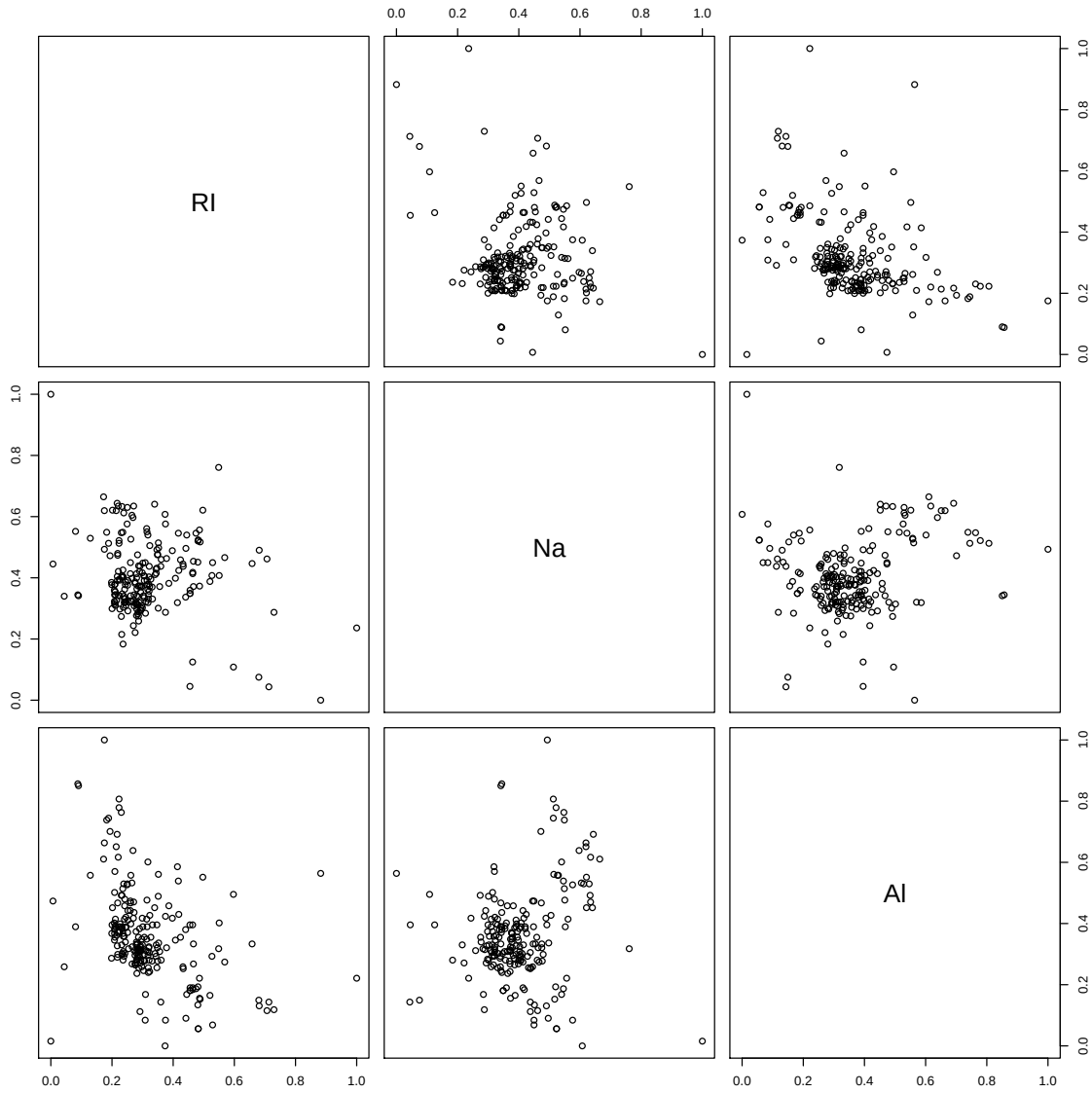
Example, Forensic Glass:

Can you tell what kind of glass it was from measurements on the broken shards??

Y: (glass type, 3 categories)
WinF: float glass window, WinNF: non-float window, Other.
x: (already scaled to be in (0,1))
RI: refractive index, Al, Na

Plot y vs. x.




RI
Na
Al
0.0
0.2
0.4
0.6
0.8
1.0

Since we only have 214 observations we are just going to look at in-sample fit.

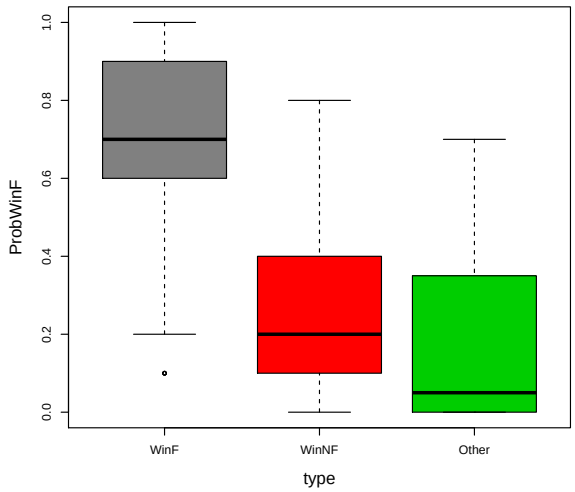
Here are the in-sample fits using kNN with $k = 10$.

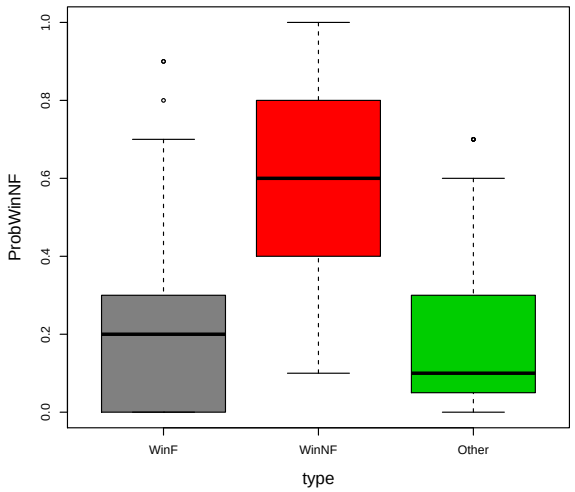
Data label on columns, "predicted" label on rows.

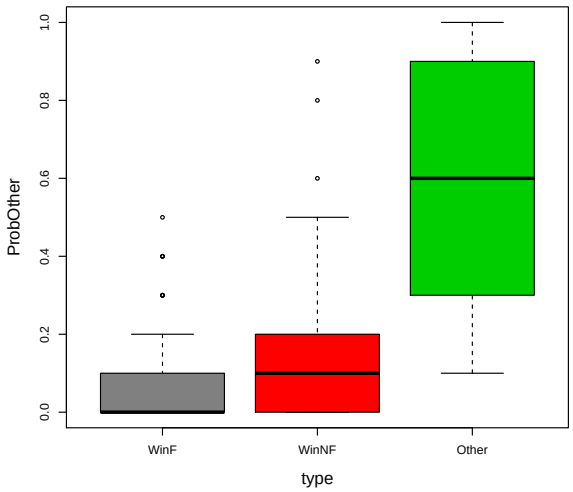
	WinF	WinNF	Other
WinF	58	13	14
WinNF	11	57	12
Other	1	6	42

pretty good !!

The first plot is $P(Y = WinF \mid x)$ vs. y=glass type.
The second plot is $P(Y = WinNF \mid x)$ vs. y=glass type.
The third plot is $P(Y = Other \mid x)$ vs. y=glass type.







pretty good !!

Example: Predicting Delinquency:

This is a kaggle competition data set.

There are 150,000 observations in the kaggle training data.

The Y is:
"Person experienced 90 days past due delinquency or worse: Y/N"

Can you predict when an account is going to be delinquent !!!!!!

We split the kaggle training data into a 50% train and 50% test.

The kaggle test does not come with y !!

We made y=1 if delinquent and 0 else.

```
> table(trainDf$y)

    0     1
69971  5029
> 5029/75000
[1] 0.06705333
```

6 to 7 % of accounts are delinquent.

n=150,000.

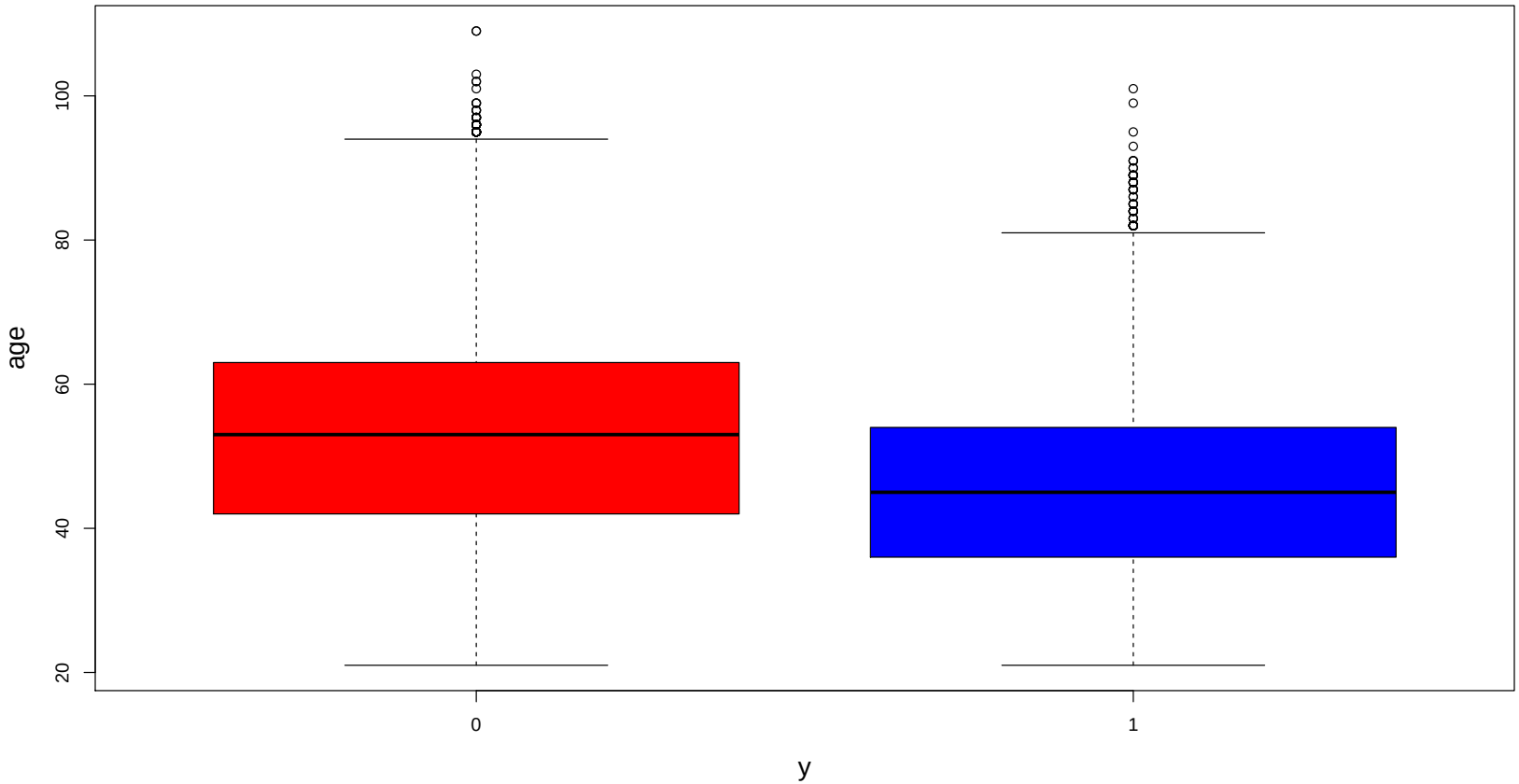
There are 9 x variables.

For example,

DebtRatio:
Monthly debt payments, alimony,living costs divided by monthy gross income: percentage

age:
Age of borrower in years: integer

For example, it looks like older people are less likely to be delinquent.



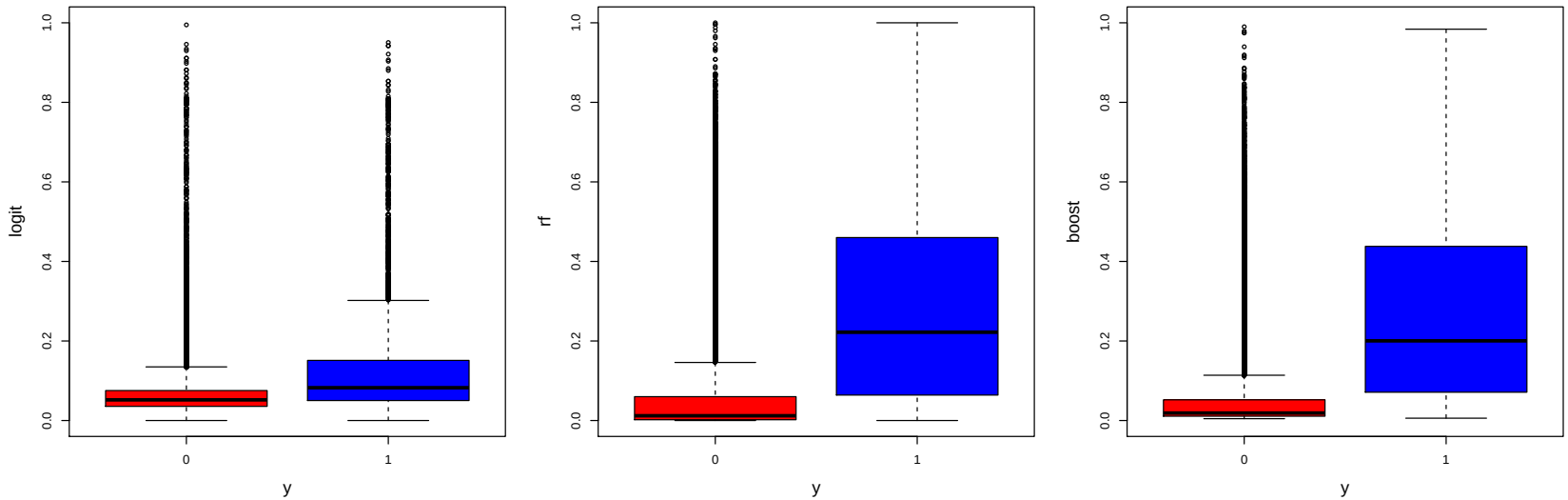
We ran boosting, random forests, and logit on train.

From each method we get $\hat{p}$, the estimate of $P(Y = 1 \mid x)$ for each x in our test data.

Let's graphically compare the $\hat{p}$ to y (on test).

Each plot relates $\hat{p}$ to y.

Going from left to right, $\hat{p}$ is from logit, random forests, and boosting.



Boosting and random forests both look *pretty good !!*

Both are **dramatically better than logit !!**.

2. The Lift

The *lift curve* is a popular method for graphically displaying the effectiveness of an estimate of $\hat{p} = P(Y = 1 \mid x)$ for a binary Y.

You have a vector of y and a corresponding vector of $\hat{p}$.

Each of the y is either a 0 or a 1.

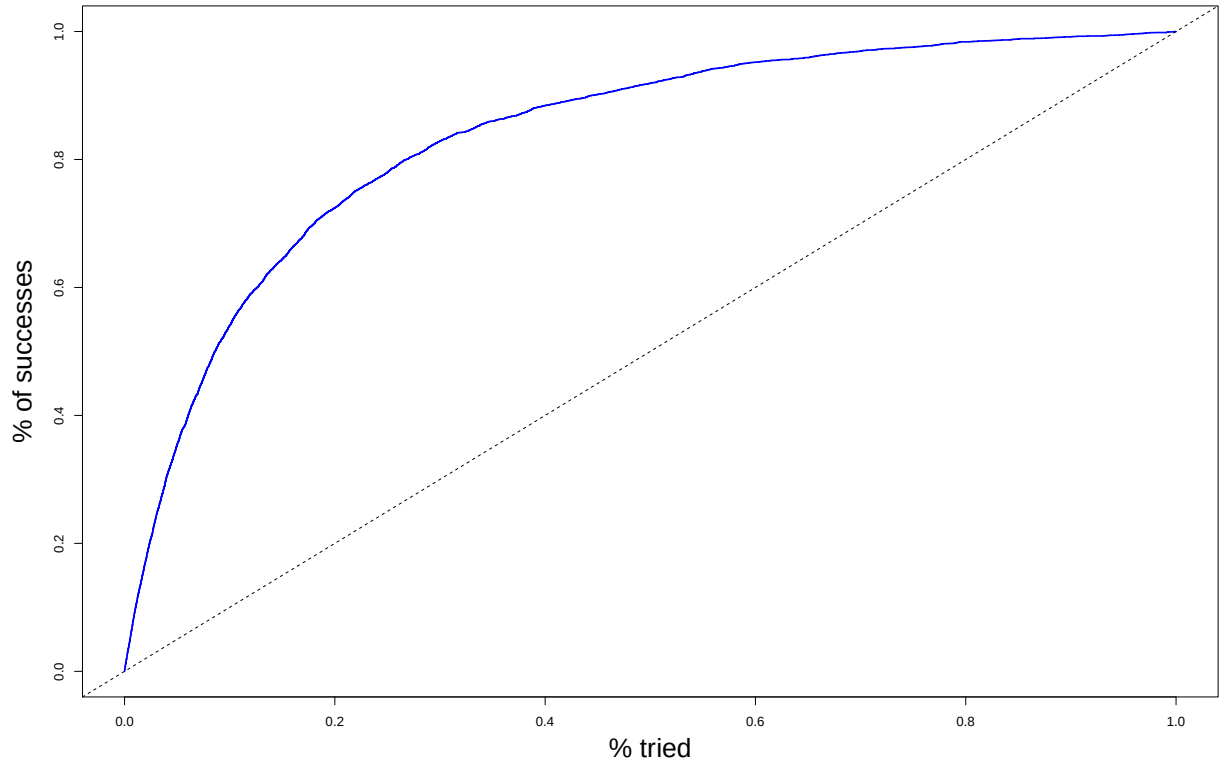
You get to choose observations, and the faster you find all the 1's the better!!

If you believe $\hat{p}$, your first choice will be the one with the biggest $\hat{p}$ your second choice will be the one with the second biggest $\hat{p}$ and so on.

That is, you would sort so that we go from biggest $\hat{p}$ to smallest and then take the observations in that order.

We then plot (% observations taken) vs. (% 1's found).

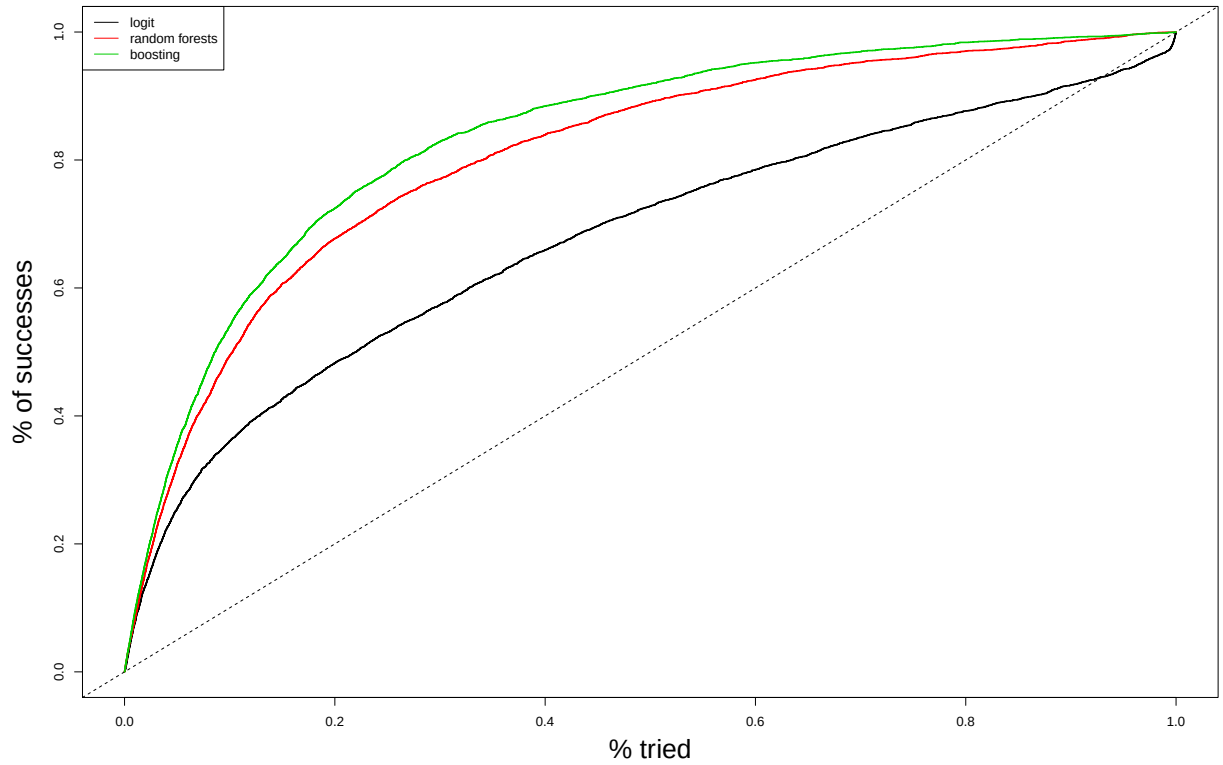
Here is the lift curve for the boosting $\hat{p}$'s.



So, for example, from only 20% of the data, you have found 72% of the 1's.

The line represents the average performance you would get by just choosing randomly.

Here is the lift curve for all three $\hat{p}$'s on the same plot so we can compare.



Boosting and random forests are both much better than logit!!!

Boosing better then RF by a bit.

Example: Tabloid Target Marketing Data

Y is 1 if a customer responds to a promotion (mailed a "tabloid") and 0 otherwise.

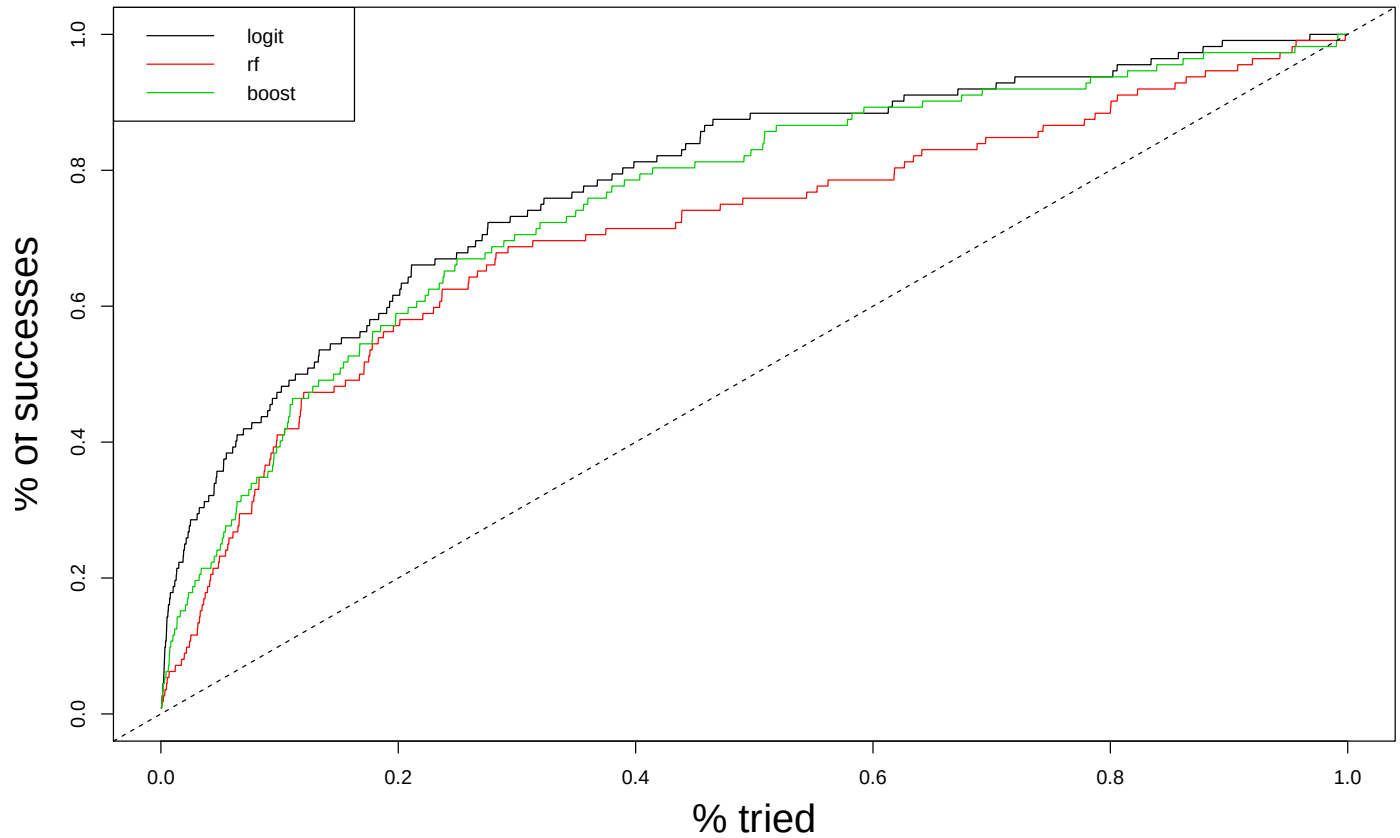
4 x's from the data base (selected from many other similar ones).

- nTab: number of past orders.
- moCbook: months since last order.
- iRecMer1 : 1/ months since last order in merchandise category 1.
- llDol: log of the dollar value of past purchases

10,000 in train; 5,000 in test.

Note: in the train, only 2.58 % respond.

You could get 60% of the potential customers by only mailing to 20%.



Suppose you had a budget that only allowed you to mail to 20% of the customers. This would be a big improvement over guessing!

In this case, logit actually wins!!
But not so dramatically as in the delinquency application.

A little background on the Tabloid example and the delinquency example.

Why do the tree methods kill in delinquency and logit look pretty good in Tabloid?

We go the Tabloid data from a company that has a lot experience using logit models to do target marketing.
They have spent a lot of time learning what variables (and transformation) make logit work.

If you look at the delinquency data, some of the x's "are a mess".
Here is where an automatic method based on trees can work a lot better than fitting a logit without working on your x's.

Of course, there could also be some interesting non-linearity logit cannot capture!!

3. Loss Functions

An important aspect of our approach we will have to consider is the choice of loss function.

While losses that are problem specific are often preferable (e.g how much money lost) it is often useful to have generic loss functions.

We use loss measures to assess our out-of-sample performance (e.g. when doing cross-validation).

We also use loss measures in estimation.

For example to fit a tree we use loss on the training data and a measure of tree complexity.

For numeric Y we often use

$$SE = \sum_{i=1}^{m} (Y_i - \hat{Y}_i)^2, \quad MSE = \frac{1}{m} SE, \quad RMSE = \sqrt{MMSE}$$

Note that for a $(Y, \hat{Y})$ pair we have the loss $L(Y, \hat{Y}) = (Y - \hat{Y})^2$ so that

$$SE = \sum_{i=1}^{m} L(Y_i, \hat{Y}_i)$$

Note that given (x, y), we often we have $\hat{Y} = \hat{f}(x)$
so can could also write

$$L(x, y) = (y - \hat{f}(x))^2$$

The loss when we predicted using x and y actually occured.

Missclassification Rate:

For categorical data, an obvious loss is the *missclassification rate*.

If $Y, \hat{Y} \in \{1, 2, \ldots, C\}$ we can let

$$L(Y, \hat{Y}) = \begin{cases} 0, & Y = \hat{Y} \\ 1, & Y \neq \hat{Y} \end{cases}$$

If you guess right, you lose nothing, if you are wrong, you lose 1.

Then

$$MR = \frac{1}{m} \sum_{i=1}^{m} L(Y, \hat{Y})$$

is the fraction miss-classified.

Example: Hockey Penalty Data

Data on each penalty in an NHL game after the first one in the game:

Y:
1 if the penalty is *not* on the same team as the previous penalty, 0 else.

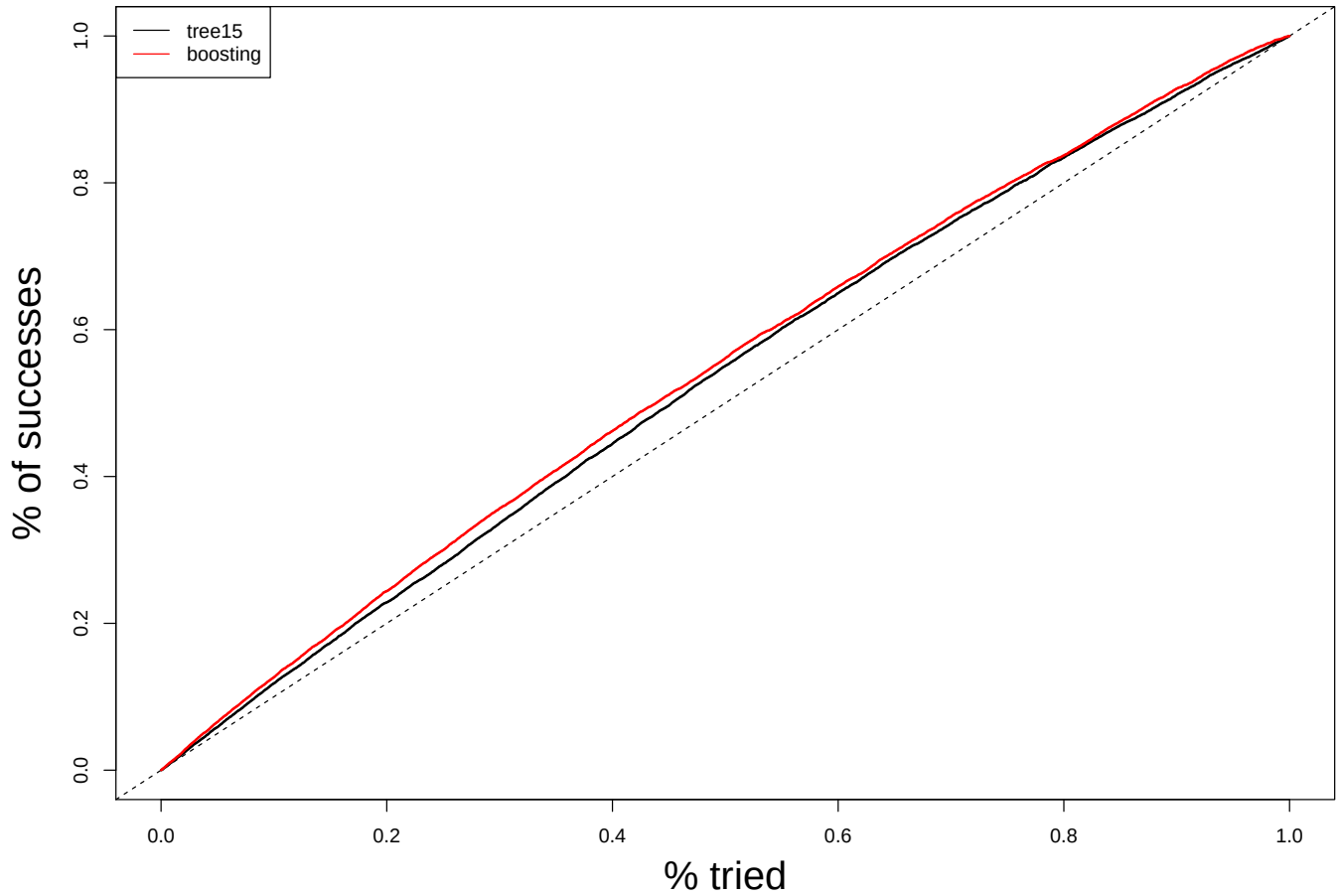
We call $Y = 1$ a *reverse call*.

Several x's describing the game situation.

```
> names(pendu)
 [1] "revcall"     "inrow2"      "inrow3"      "inrow4"      "home"
 [6] "ppgoal"      "tworef"      "timeingame"  "dayofseason" "numpen"
[11] "timebetpens" "goaldiff"    "gf1"         "ga1"         "pf1"
[16] "pa1"         "gf2"         "ga2"         "pf2"         "pa2"
[21] "X1995"       "X1996"       "X1997"       "X1998"       "X1999"
[26] "X2000"       "X2001"
```

train: 47,883 observations.
test: 10,000.

Here are the test lifts for a tree with 15 bottom nodes and boosting.



It may not look like a lot of fit, but given the nature of the problem, *is a lot of preditability !!*.

In this case, we might classify an upcoming penalty as a reverse call if the probability is greater than .5.

Here are the two-way tables relating the revcall outcome (rows) and the classification (columns) for the tree and boosting models.

```
   classboost
    FALSE TRUE
0     779 3361
1     645 5215
```

MR = (645+3361)/10000 = .4, or 40%.

```
   classtree
    FALSE TRUE
0     789 3351
1     715 5145
```

MR = (715+3351)/10000 = 0.4066

MR is about the same for tree and boosting.

These two-way table relating the predicted outcome and the actual outcome for a binary Y are called *confusion matrices*.

```
   classboost
    FALSE TRUE
0     779 3361
1     645 5215
```

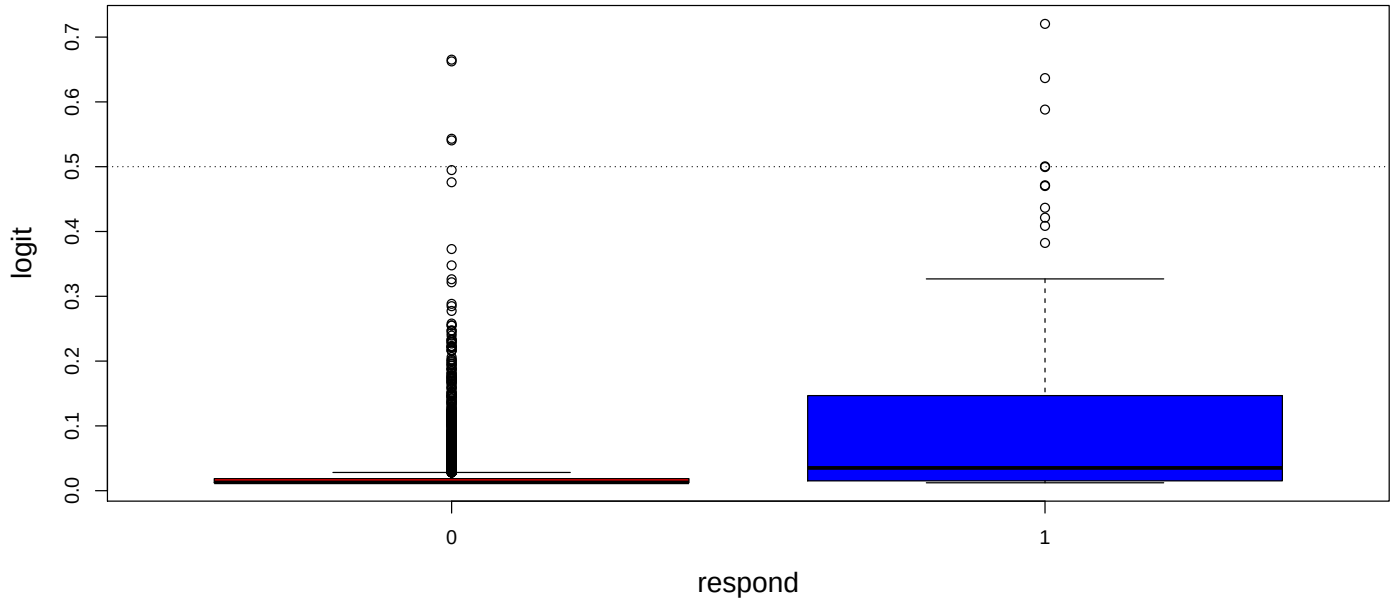
We like big numbers on the diagonals.

Deviance Loss:

For a lot of problems, missclassifcation is not useful because while the model has fit, it is usually pretty obvious what the most likely category is.

For example, in our target marketing example, the probability of a response is almost always less than .5 so if you just predict "they won't respond" you do pretty good in terms of missclassification, but that is not helpful.

Here is the test plot of $\hat{p}$ (from logit) and y=respond for the tabloid data.



While the model works, it is pretty useless to predict a response if $\hat{p} > .5$!!

Only 112 out of 5,000 actually responded so if we just say noone responds the MR is 2%.

Given x, let we could say a measure of how good our model is given we observe $Y = y$ is just $P(Y = y \mid x)$.

The *deviance* measure of how *bad* things are is

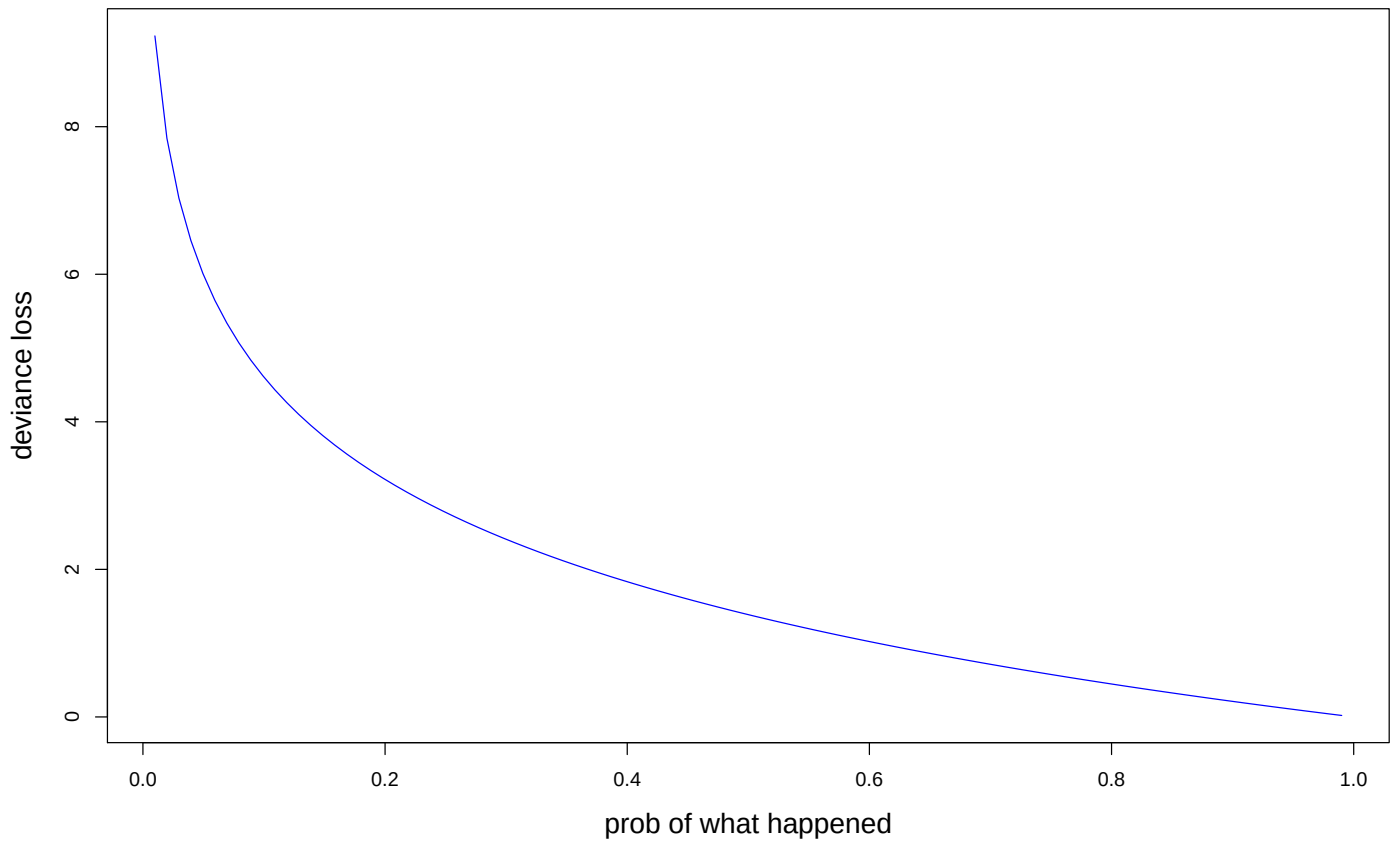
$$L(x, y) = -2 \log(P(Y = y \mid x))$$

-2 $\times$ the log of the probability of what happened, where the probability is determined by your model fit.

This is our measure of loss when we predicted using x and y actually occured.

$$L(x, y) = -2 \log(P(Y = y \mid x))$$

- want $P(Y = y \mid x)$ big.
- want $\log(P(Y = y \mid x)$ big.
- want $-2 \log(P(Y = y \mid x))$ small.
- the 2 is a convention from likelihood analysis.



Example:

Previously, we looked at boosting and random forest predictions for the kaggle delinquency data.

How did we choose the settings?

We tried a variety of settings, fit on train and then computed our loss on test.

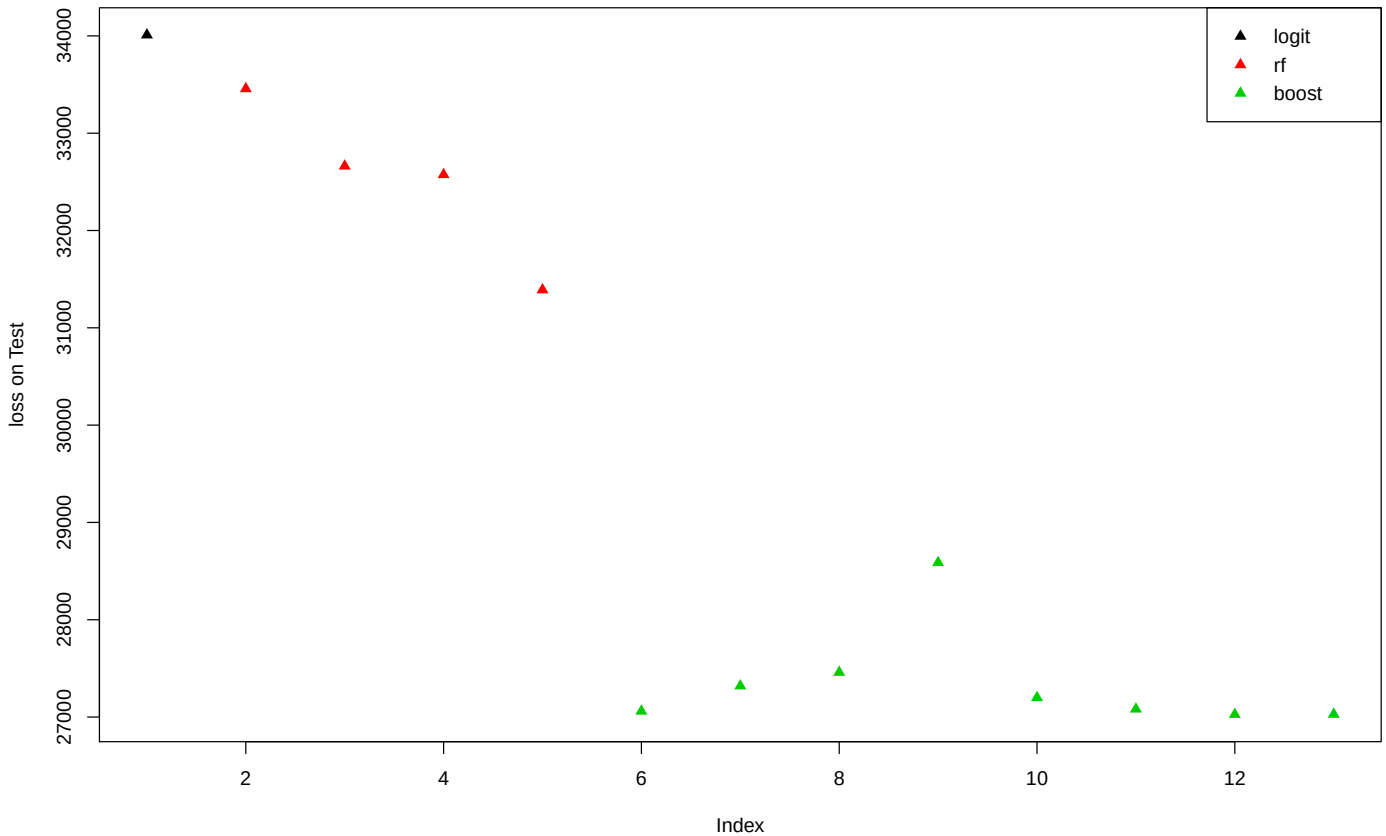
Random Forest settings:

	mtry	ntree
1	9	500
2	3	500
3	9	1000
4	3	1000

Boosting settings:

	tdepth	ntree	shrink
1	2	1000	0.10
2	4	1000	0.10
3	2	5000	0.10
4	4	5000	0.10
5	2	1000	0.01
6	4	1000	0.01
7	2	5000	0.01
8	4	5000	0.01

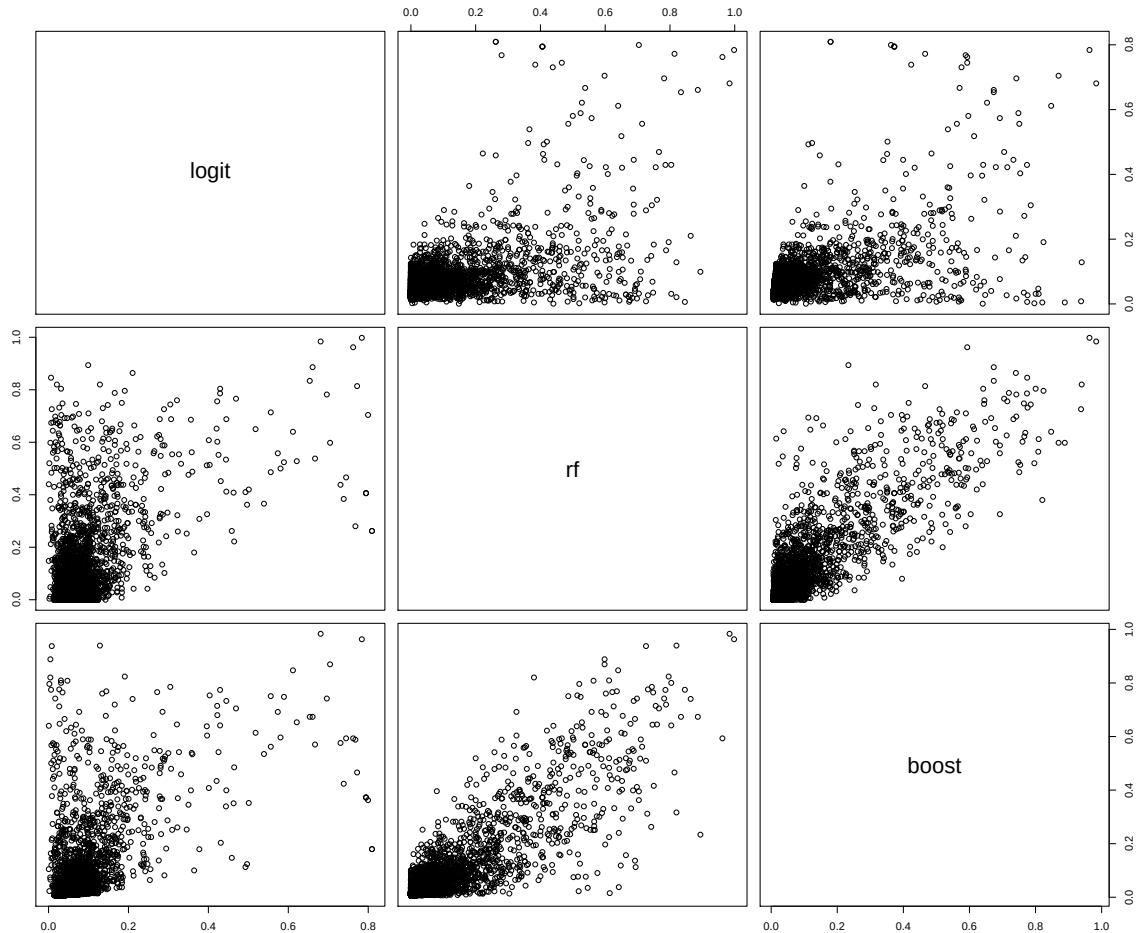
Here are the out-of-sample (test) deviance losses for the logit model, 4 random forest models, and 8 boosting models.



Clearly boosting looks good.

The results we showed previously used the minimal test loss setting.

The test $\hat{p}$ for a sample of 5,000 test observations.



Boosting and random forests are picking up some of the same signal.

Notes:

The deviance is one of two options the tree package gives for loss on the training data.

The in-sample deviance is the -2 $\times$ log-likelihood for a classification model.

The logit estimates of the coefficients (the β's) are chosen by minimizing the (in-sample) deviance which is equivalent to maximizing the likelihood.

While the deviance is not terribly interpretable, it gets used a fair amount in statistics.

For binary classification problems an obvious loss is $|y - P(Y = 1 \mid x)|$ where y is 0/1.

4. Decision Theory Expected Utility

use tabloid data to illustrate choice of optimal action based on $p(y = 1|x)$.

compare boosting and random forests based on how much money you make using the tabloid data.

We actually show that you make more money with boosting than rf!!!!!