Classification

Carlos Carvalho, Mladen Kolar, Robert McCulloch

1. Classification
2. The Lift
3. Loss Functions
4. Decision Theory and Expected Utility
5. The ROC Curve

1. Classification

Our class so far – Numeric $Y \longrightarrow$ Regression:

$$Y = f(x) + \epsilon$$

For example, a linear function:

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

What about categorical or binary Y?

Can we use a linear regression?

Binary $Y \in \{0, 1\}$:

For example, you may have seen logistic regression

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

In both cases, x only affects Y through a linear combination.

However, in this class we want to learn a more generic specification for $Y \mid X = x$.

In general, when we have categorical Y that we are trying to predict, we say we have a *classification* problem.

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

That is, the Y outcome must be a member of a finite set S with C members.

Often we just use integers to label the possible outcomes: $S = \{1, 2, \ldots, C\}$.

Given x, many methods will give

$$P(Y = y \mid x), \quad y \in S.$$

the conditional distribution of Y given x.

Some methods just give a predicted y from S.

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, so that $Y \in S = \{0, 1\}$.

While the big ideas (such as the bias-variance trade-off) are the same for predicting a categorical variable and a numeric variable, some of the details of our modeling are necessarily 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 $P(Y = y \mid x)$ with the proportion on neighbors having $Y = y$.

Example, Forensic Glass:

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

Y: glass type, 3 categories.

$Y \in S = \{\text{WinF}, \text{WinNF}, \text{Other}\}$.

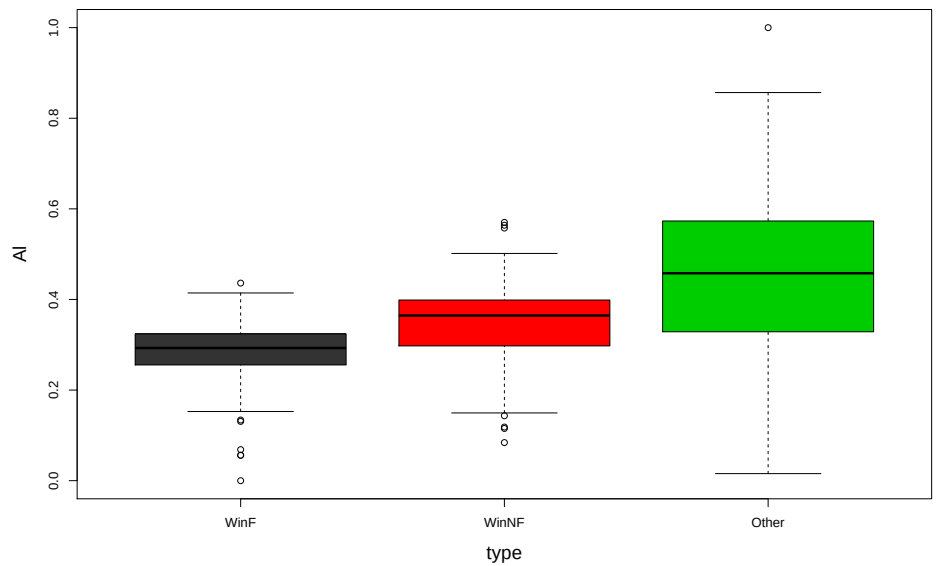
WinF: float glass window
inNF: non-float window
Other.

x: 3 numeric x's:
$x_1 = \text{RI}$: refractive index
$x_2 = \text{Al}$
$x_3 = \text{Na}$

Is Y related to x_2 =AI?

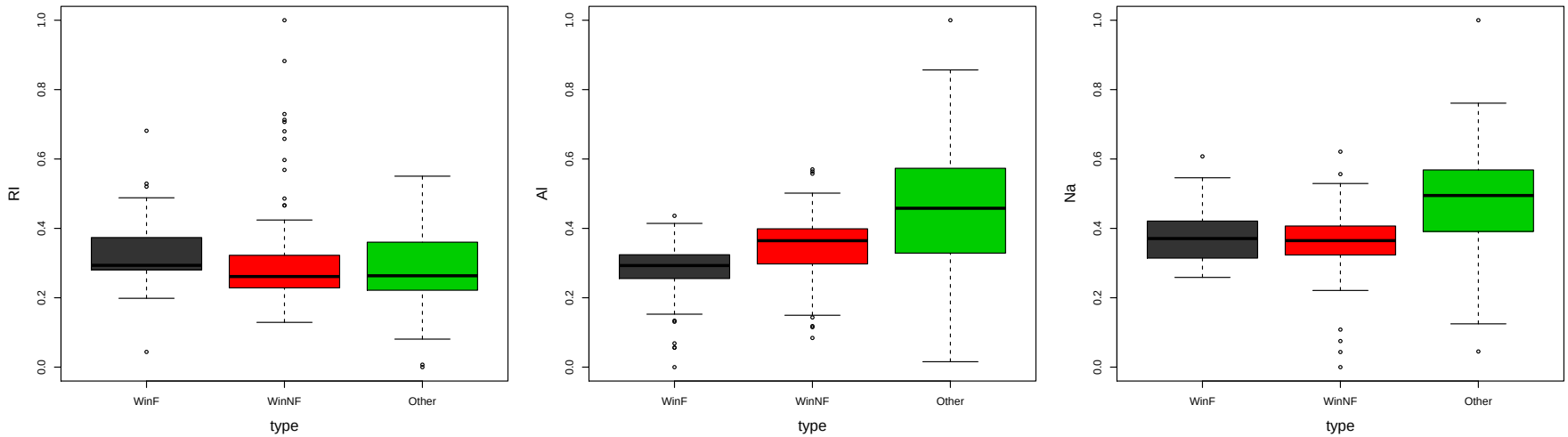
How do we plot a categorical Y vs a numeric x?

For each level of Y, pick off the subset of the data such that Y is at that level and then display the x values using a boxplot.



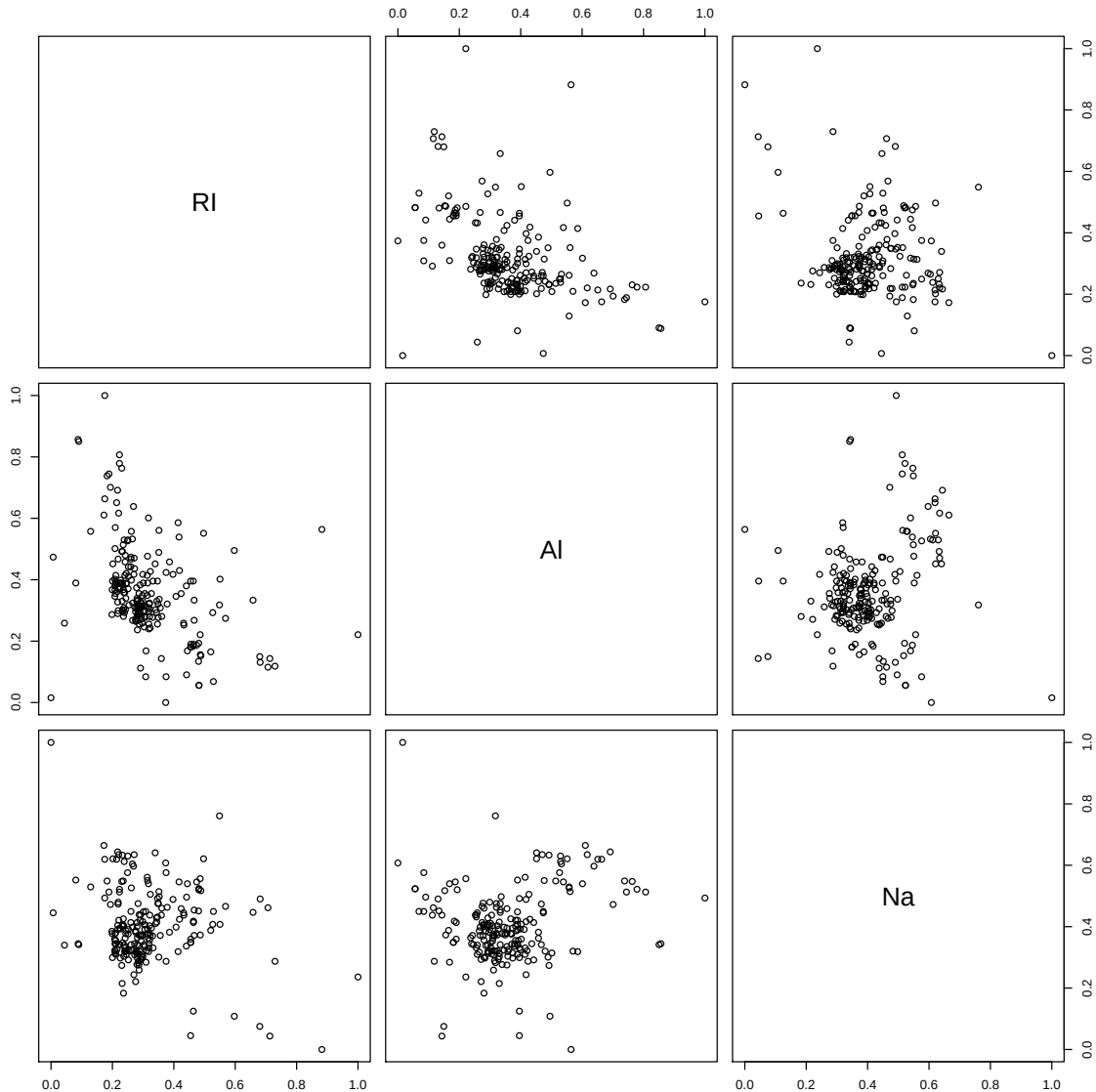
For big x_2 =AI, "Other" seems more likely for Y.

Plot y vs. each x.

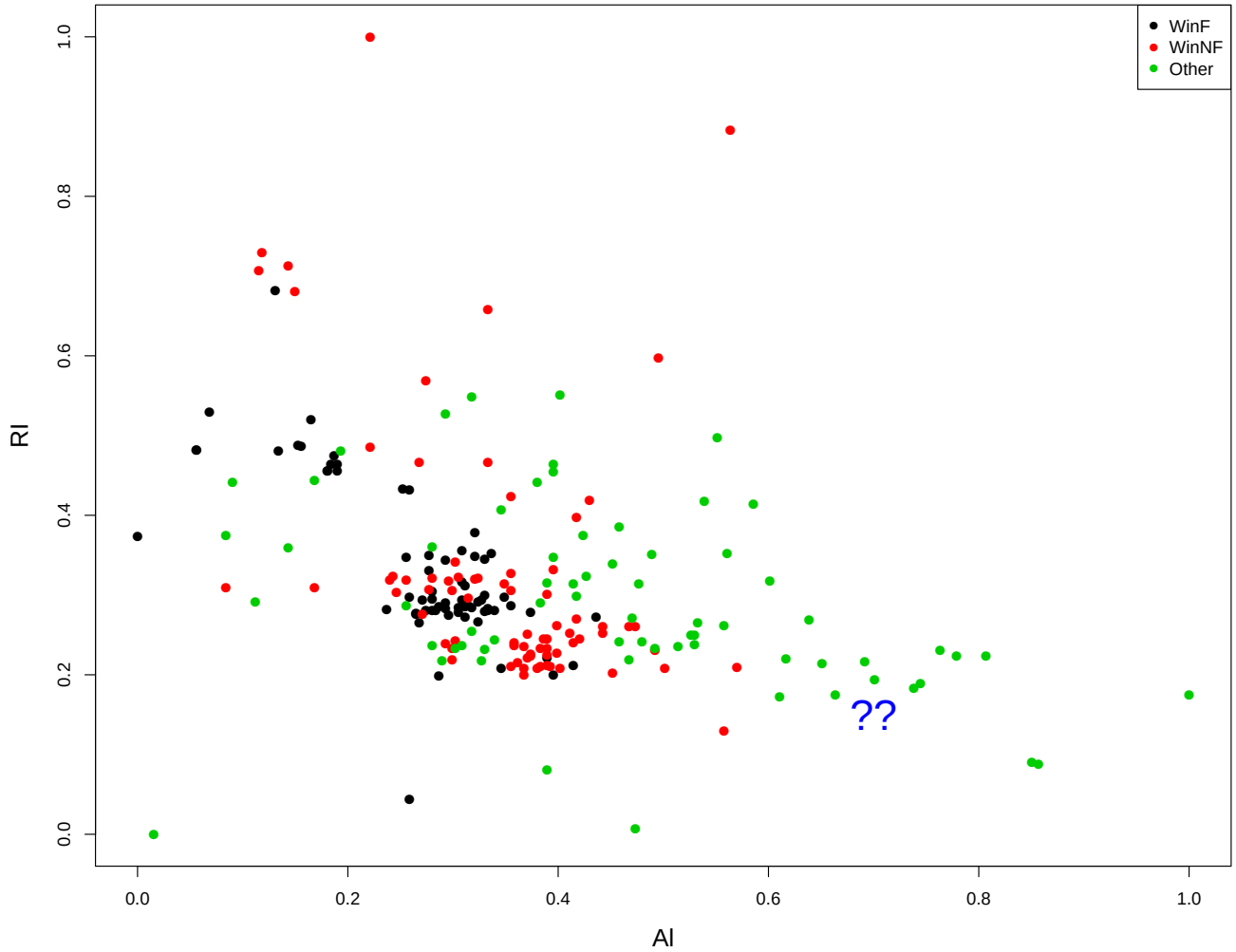


x_2 =Al looks like a winner, but there may also be information about whether Y =WinF in x_1 =RI, and information about whether Y =Other in x_3 =Na.

How does kNN use the x's to predict the categorical Y?



With just Al and RI:



If (Al,RI) = ??, what is prediction for Y?

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

We used kNN with $k = 10$ and stored the result in `near`.

```
near$fitted[1:50]
 [1] WinF  WinNF WinNF WinF  WinF  WinNF WinF  WinF  WinF  WinF  WinNF WinF
[13] WinNF WinF  WinF  WinF  WinF  WinF  WinF  WinNF WinNF WinF  WinF  WinF
[25] WinF  WinF  WinF  WinF  WinF  WinF  WinF  WinF  WinF  WinF  WinF  Other
[37] WinF  WinF  WinF  WinF  WinF  WinF  WinF  WinF  WinF  WinNF WinNF WinF
[49] WinF  WinF
Levels: WinF WinNF Other

> near$prob[1:5,]
     WinF WinNF Other
[1,]  0.6   0.3   0.1
[2,]  0.4   0.4   0.2
[3,]  0.1   0.9   0.0
[4,]  0.7   0.3   0.0
[5,]  0.8   0.2   0.0
```

The two-way table relating the observed Y with the predicted Y is called the *confusion matrix*.

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

So, there are 58+11+1 observations with Y =WinF.
Of those 11 were predicted to be WinNF.

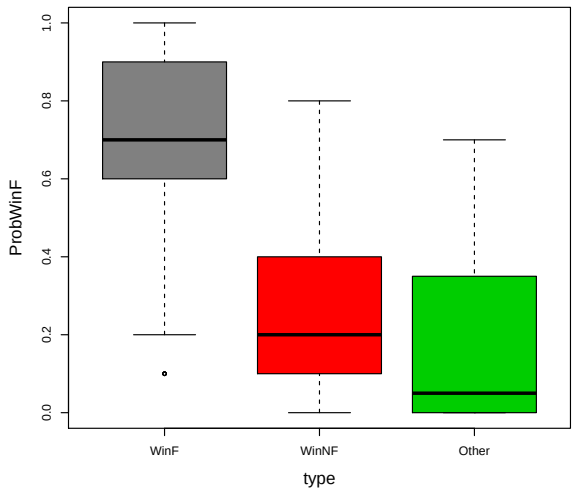
	WinF	WinNF	Other
WinF	58	13	14
WinNF	11	57	12
Other	1	6	42

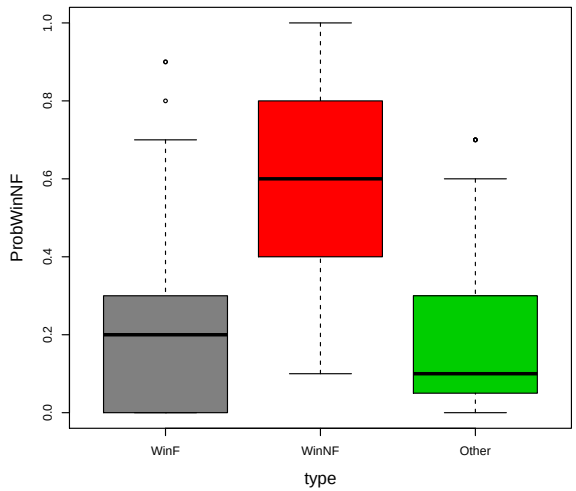
We like the diagonals big!
Missclassification rate: (214-(58+57+42))/214 = 0.27

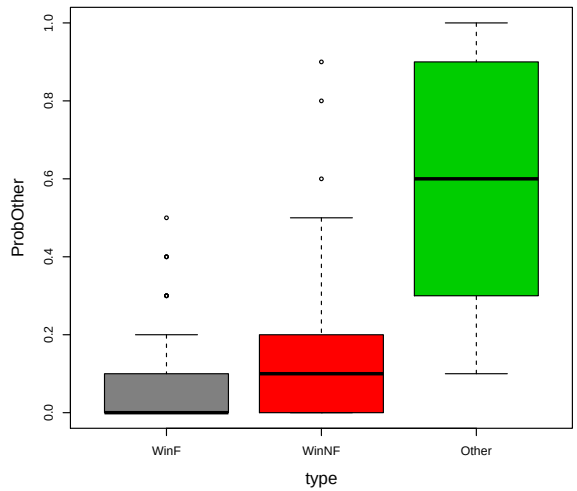
pretty good !!

How good are the probabilities ??

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 !!

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 $P(Y = y \mid x)$ 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.

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.

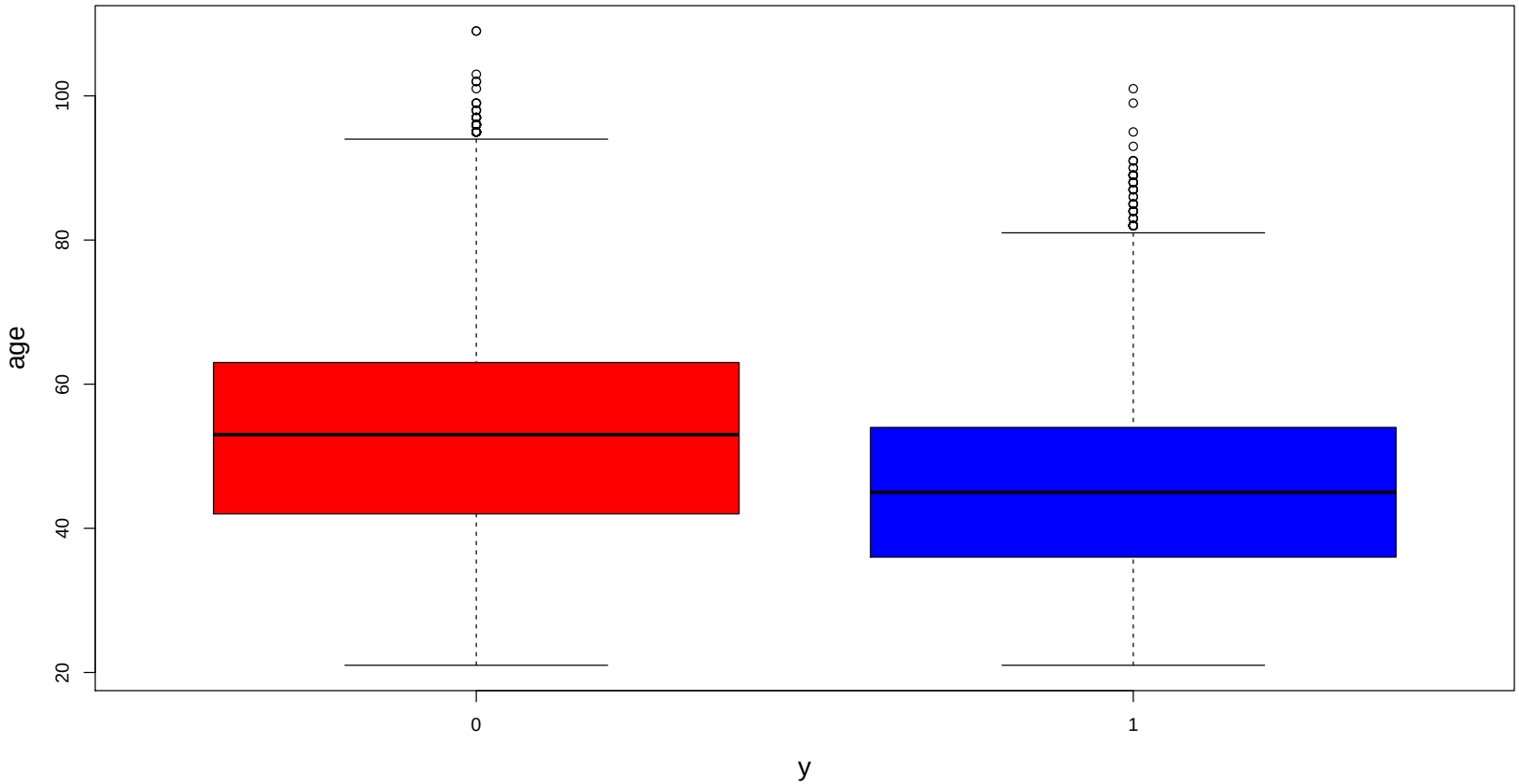
There are 9 x variables.

For example,

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

age:
Age of borrower in years: integer

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



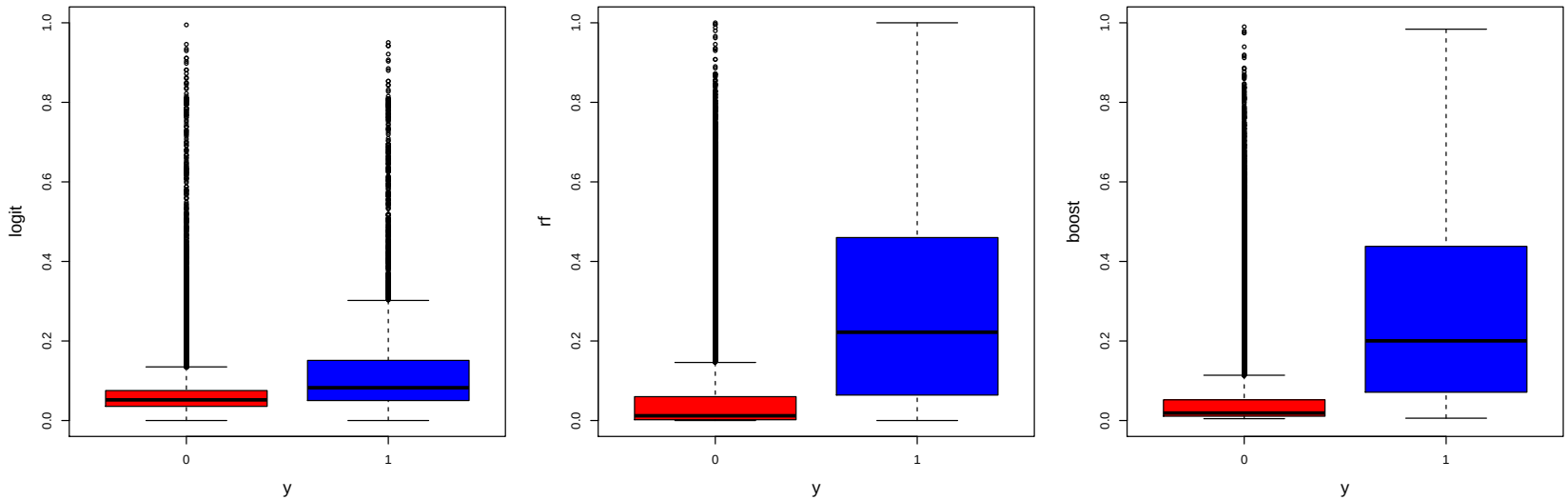
We ran boosting, random forests, and logit on train.

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.

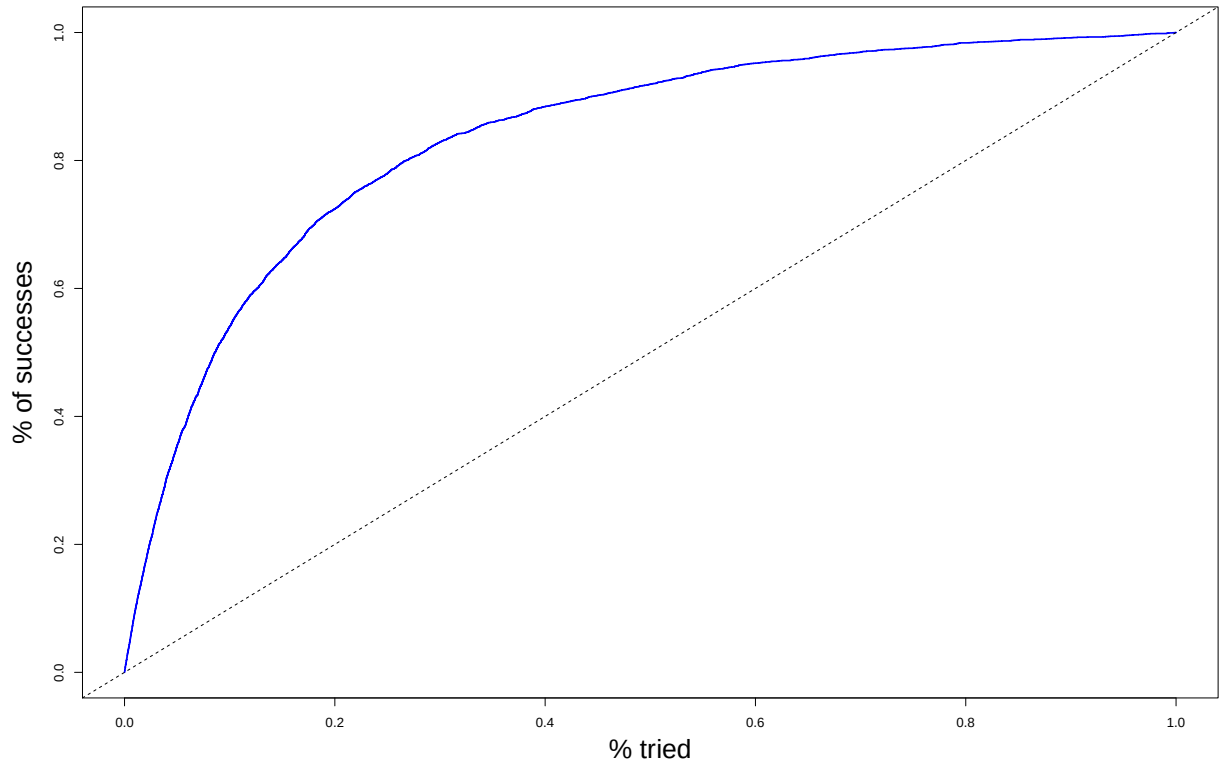
You get to choose observations, and the faster you find all the 1's the better!!

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).

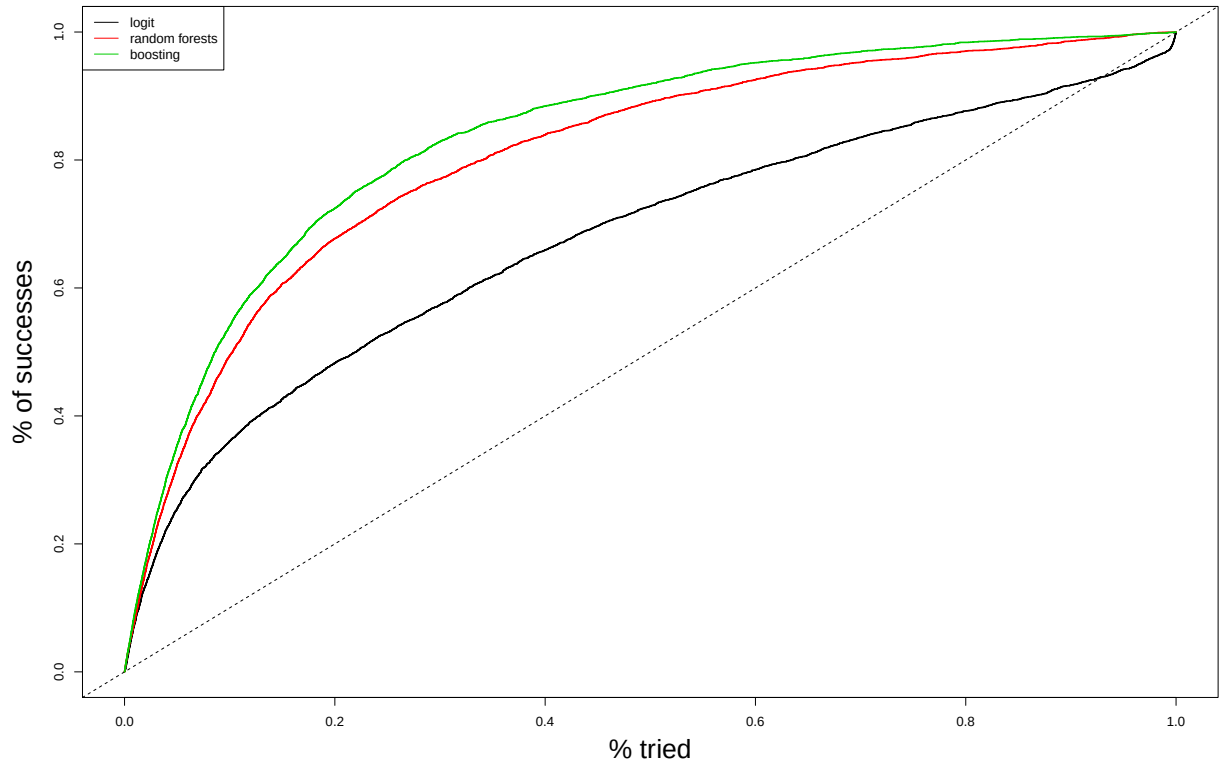
Here is the lift curve for the boosting $\hat{p}$'s.



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!!!

Boosting 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.

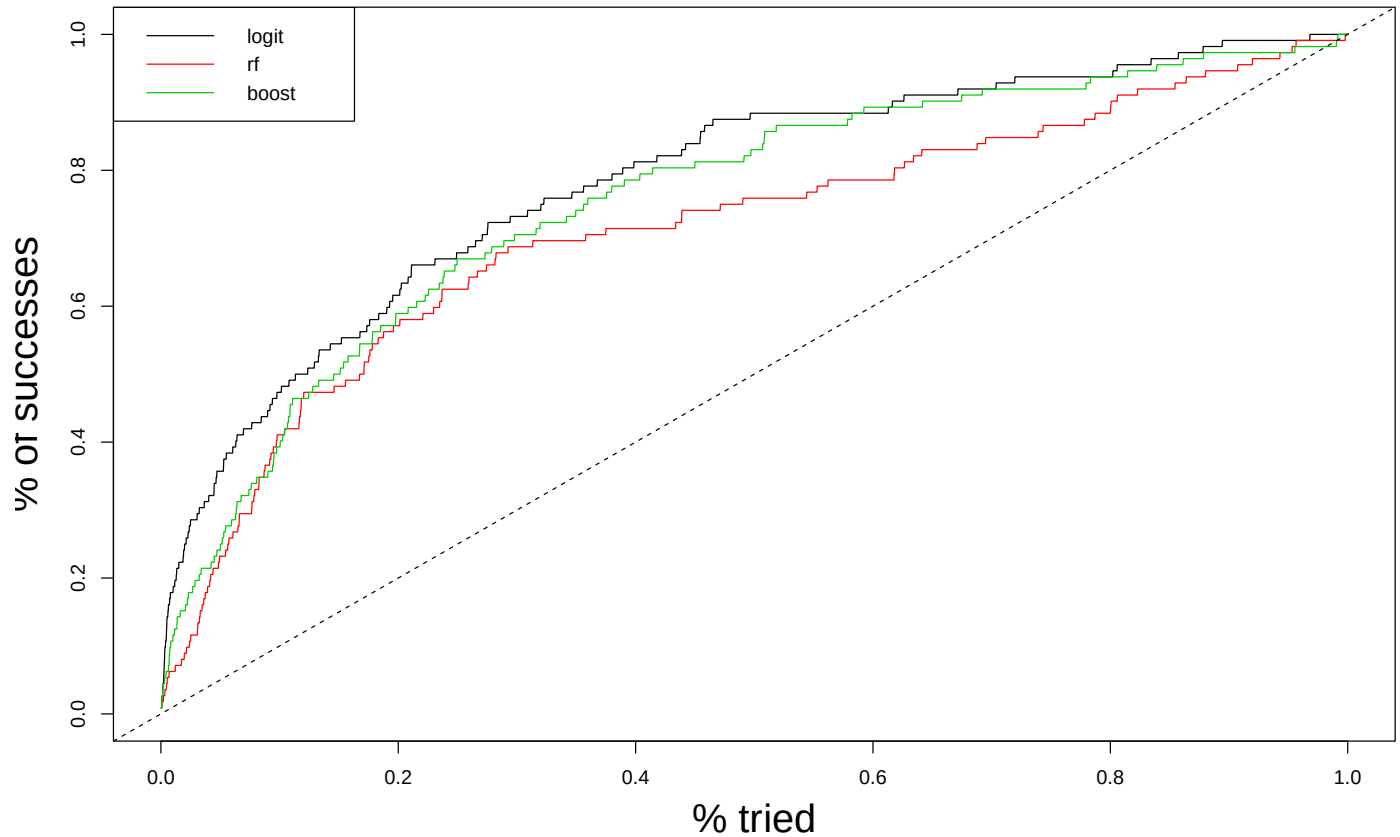
4 x's from the data base (selected from many other similar ones).

- 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 got 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.

While people mostly use RMSE and MAD ($|Y - \hat{Y}| = |Y - \hat{f}(x)|$) for numeric Y, choosing a loss for categorical Y can be tricky.

For example, the tree package gives you the option of choosing *deviance loss* or the *Gini criterion* when fitting trees.

We'll review RMSE and then discuss the missclassification rate and deviance as performance measures for categorical outcomes.

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 occurred.

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

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

is the fraction miss-classified.

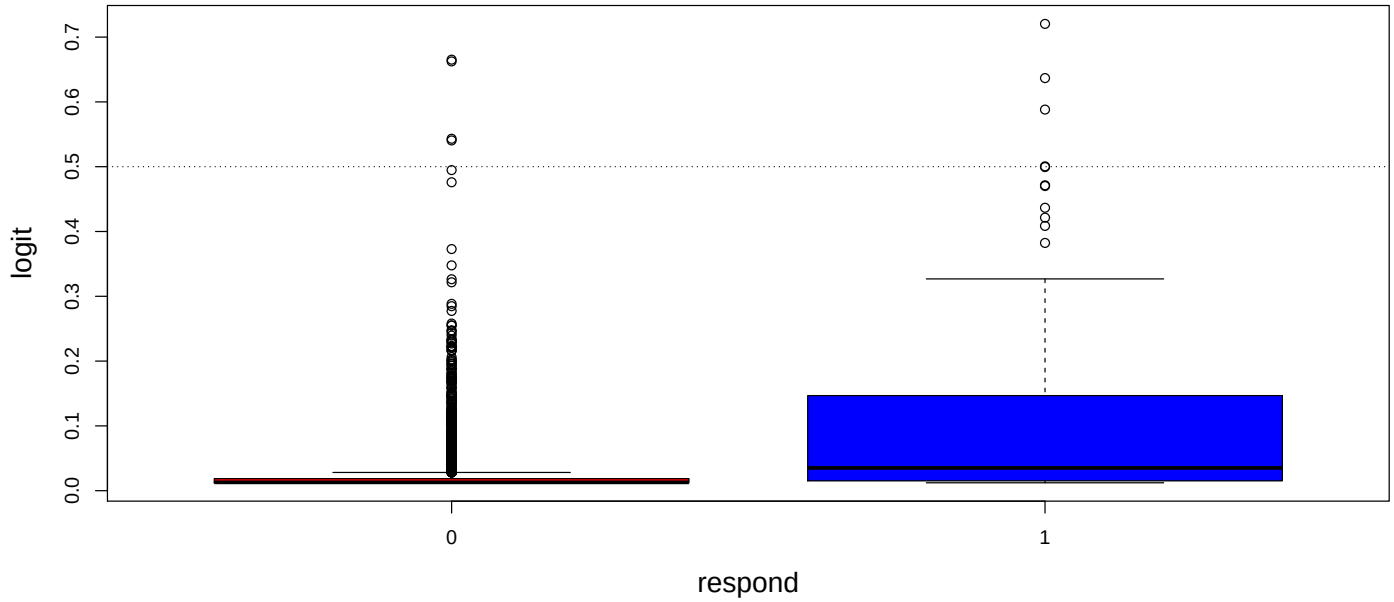
We saw this in the forensic glass example.

Deviance Loss:

For a lot of problems, missclassification 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 no-one responds the MR is 2%.

Suppose we have an (x, y) pair where x is a vector of predictors and y is the corresponding outcome for a categorical Y.

Our model gives us $P(Y = y \mid x)$.

We could say a measure of how well our model worked for this observation is just $P(Y = y \mid x)$.

If the model says what happened is likely, that is good !!!

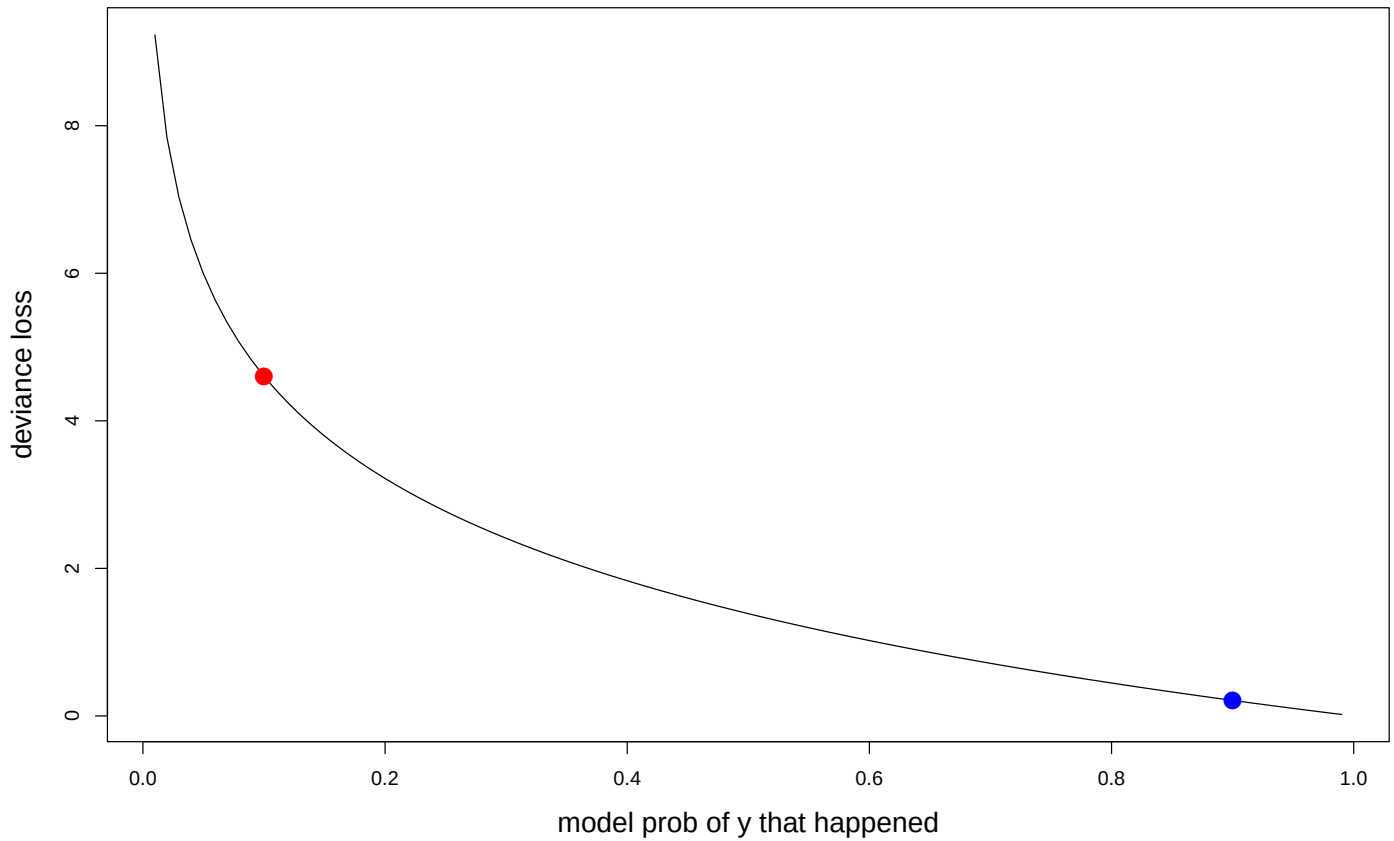
The *deviance* measure of how *bad* things are is

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

- 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.

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

If you say a certain y outcome is likely and then it happens, your loss is small.



If you say a certain y outcome is very unlikely and then it happens, your loss is big.

Then for data (x_i, y_i) (train) or (test)

Total loss is

$$\sum L(x_i, y_i) = \sum -2 \log(P(Y = y_i \mid x_i))$$

Example:

A tiny data set with 6 observations and model fits from model 1 ($P1(Y = y \mid x)$) and model 2 ($P2(Y = y \mid x)$).

```
     x y P1(Y=1|x) P1(Y=0|x)     dev1
[1,] 1 0       0.1       0.9 0.210721
[2,] 2 0       0.1       0.9 0.210721
[3,] 3 0       0.1       0.9 0.210721
[4,] 4 1       0.9       0.1 0.210721
[5,] 5 0       0.9       0.1 4.605170
[6,] 6 1       0.9       0.1 0.210721
```

```
     x y P2(Y=1|x) P2(Y=0|x)     dev2
[1,] 1 0       0.5       0.5 1.386294
[2,] 2 0       0.5       0.5 1.386294
[3,] 3 0       0.5       0.5 1.386294
[4,] 4 1       0.5       0.5 1.386294
[5,] 5 0       0.5       0.5 1.386294
[6,] 6 1       0.5       0.5 1.386294
```

Note: -2*log(.5) = 1.386294, -2*log(.1) = 4.60517, -2*log(.9) = 0.210721

Deviance under Model 2: 6*1.386294 = 8.317764

Deviance under Model 1: 5*0.210721 + 4.605170 = 5.658775

What happens if we fit a tree to this data set in R?

```
xydf = data.frame(x=1:6,y=as.factor(c(0,0,0,1,0,1)))
temp = tree(y~x,xydf,control=tree.control(6,mincut=3,minsize=6))
print(temp)
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

1) root 6 7.638 0 ( 0.6667 0.3333 )
  2) x < 3.5 3 0.000 0 ( 1.0000 0.0000 ) *
  3) x > 3.5 3 3.819 1 ( 0.3333 0.6667 ) *
```

The deviance from the left child is
3*(-2*log(1)) = 0.

The deviance from the right child is
2*(-2*log(2/3)) + (-2*log(1/3)) = 3.819085

If we print the R summary of the tree we get:

```
> print(summary(temp))

Classification tree:
tree(formula = y ~ x, data = xydf, control = tree.control(6,
    mincut = 3, minsize = 6))
Number of terminal nodes:  2
Residual mean deviance:  0.9548 = 3.819 / 4
Misclassification error rate: 0.1667 = 1 / 6
```

We get the missclassification rate and deviance as summaries.

The "average deviance" is obtained by dividing by
(n – number of bottom nodes) for reasons we skip.

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.

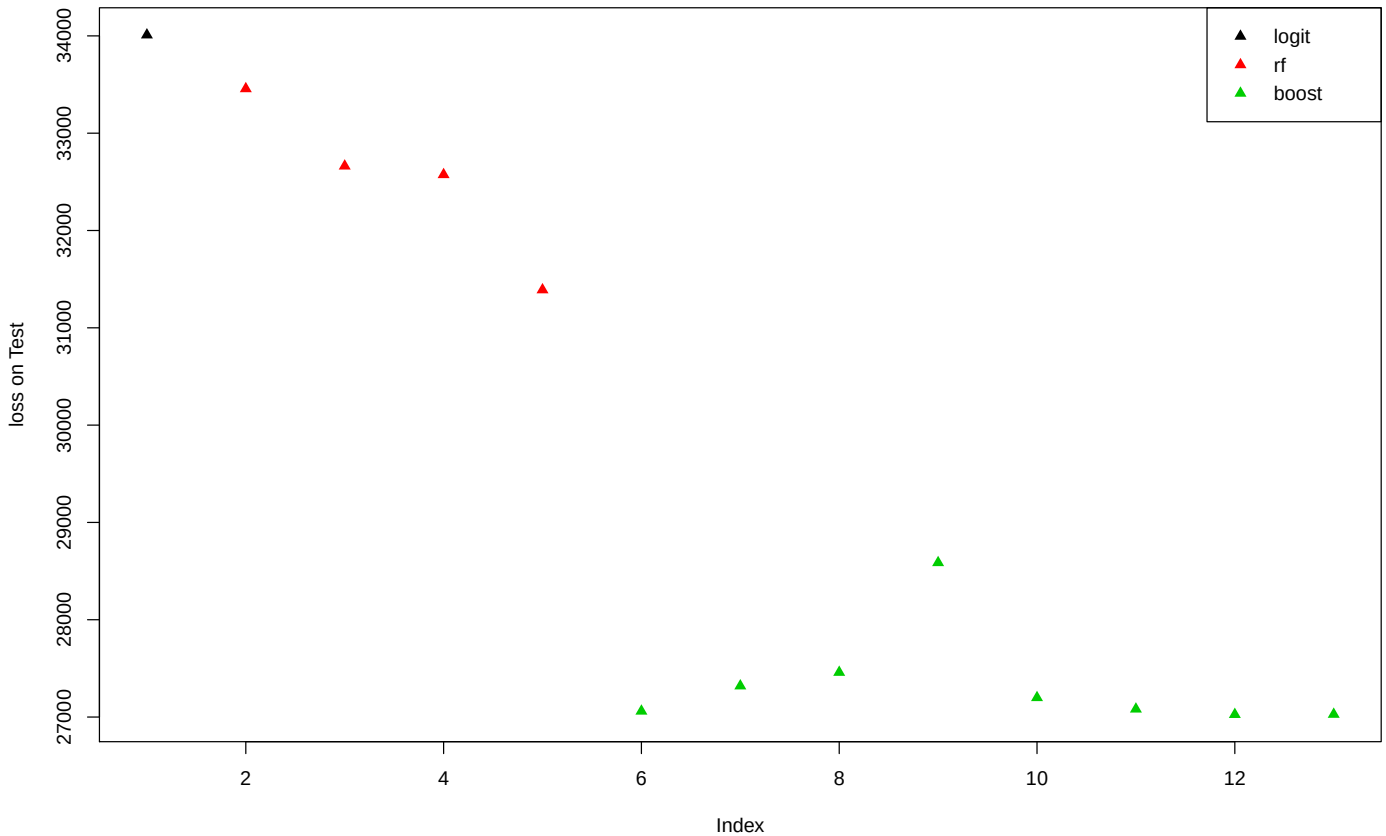
Random Forest settings:

	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

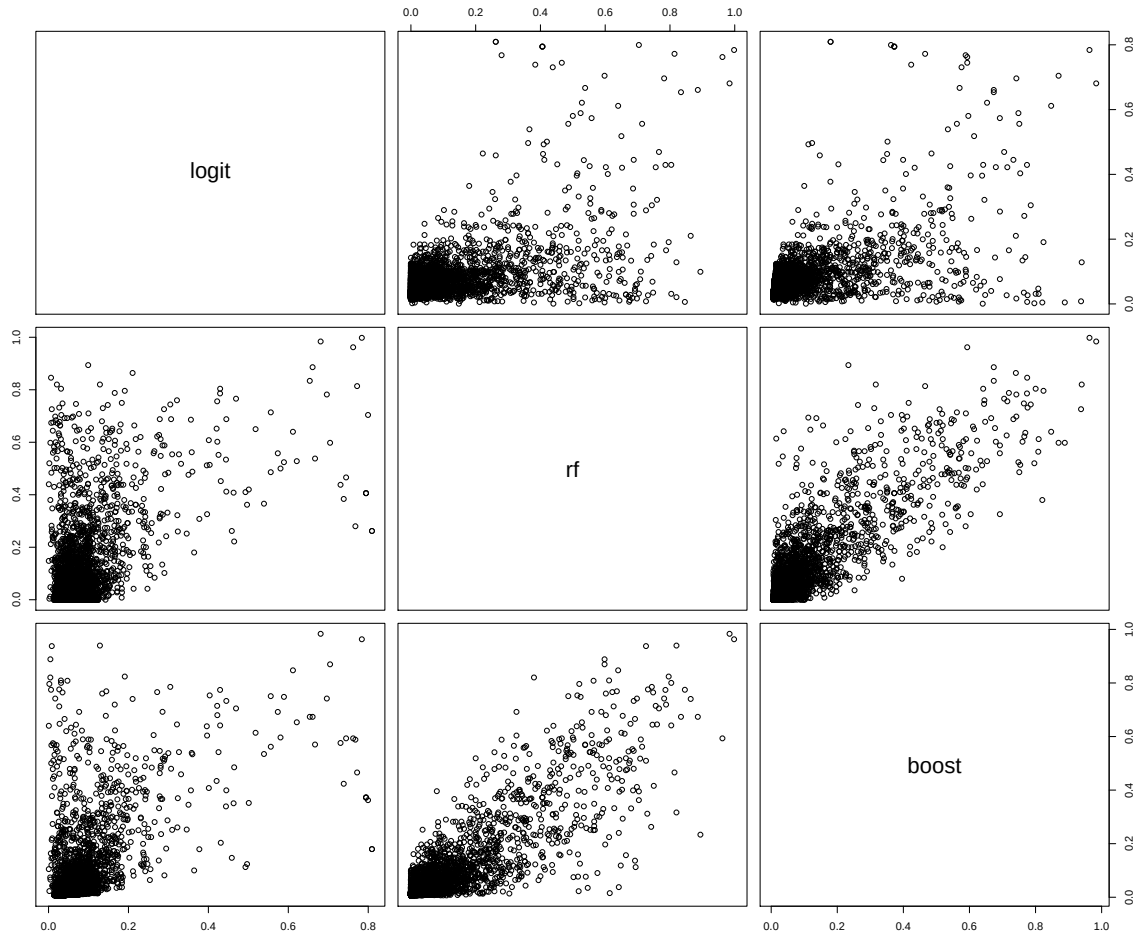
Here are the out-of-sample (test) deviance losses for the logit model, 4 random forest models, and 8 boosting models.



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 and Expected Utility

We have looked and ways to evaluate the predictive power of our models out-of-sample.

However, we may be able to use the estimated probabilities to make intelligent decisions.

Consider the target marketing example.

If, given x, the probability of a response is big, then it should make sense to target the customer.

Alternatively, if our model suggest there is a very low chance they will respond, it may be a waste of money.

How can we quantify this?

Rather than using a generic loss, we should think about our actual decision.

Let's suppose it costs \$.80 (80 cents) to mail the promotion (in our tabloid example).

Let's suppose that (on average) a customer spends \$40, **if** they respond.

If they respond we get 40-.8 = 39.20.

If they do not respond we get -.8.

Suppose the probability (given x) they respond is .05.

Let M denote the random variable which is our money payout.

The distribution of M is:

m	$Pr(M = m)$
-.80	.95
39.20	.05

Note that in thinking about our decision, it makes seems simpler to think about our gain rather than our loss.

m	$Pr(M = m)$
-.80	.95
39.20	.05

Decision theory says we should act if our expected utility is positive.

$$E(U(M)) = .95\,U(-.8) + .05\,U(39.20)$$

Often we keep things simple and use $U(m) = m$.

$$E(M) = .95\,(-.8) + .05\,(39.20) = 1.2 > 0.$$

So, if x give us a p of .05 we should do the targeting.

What is the probability cutoff such that if p is bigger than the cuttoff, we should target?

$-.8(1-p)+p(39.2)=0 \Rightarrow p=.8/40=.02.$

If $p>.02$, we should do the targeting.

Here is the confusion matrix where dotarget indicates whether $p > .02$ and the profit for logit, random forests, and boosting.

logit:

```
dotarget    0    1
   FALSE 3730   37
   TRUE  1158   75

75*40 - (75+1158)*(.8) = $2013.6
```

random forests:

```
dotarget    0    1
   FALSE 4223   59
   TRUE   665   53

53*40-(53+665)*.8 = $1545.6
```

boosting:

```
dotarget    0    1
   FALSE 3830   43
   TRUE  1058   69

69*40 - (69+1058)*.8 = $1858.4
```

As we saw with our other approaches, logit actually looks good for the tabloid problem.

Now we have an actual dollar amount for how much better it is!!

Boosting is not too bad and much better than random forests.

In the delinquency problem, boosting and random forests looked much better than logit.

It would be an interesting problem to try to put a dollar amount on how much!

5. The ROC Curve

Lift and ROC are two popular methods for assessing the quality of a classifier for a binary y.

Lift is particularly popular in Marketing.

ROC stands for the incomprehensible term "receiver operator characteristics".

Both look at missclassification rates for various values of s using the rule: classify $Y = 1$ if $P(Y = 1 \mid x) > s$.

If we classify $Y = 1$ if $P(Y = 1 \mid x) > .02$ this is our confusion matrix.

dotarget	0	1
FALSE	3830	43
TRUE	1058	69

ROC looks at:

- TP (true positive), % of y=1 correctly classified: 69/(43+69) = 0.6160714
- FP (false positive), % of y=0 incorrectly classified: 1058/(3830+1058) = 0.2164484

```
dotarget    0    1
   FALSE 3830   43
   TRUE  1058   69
```

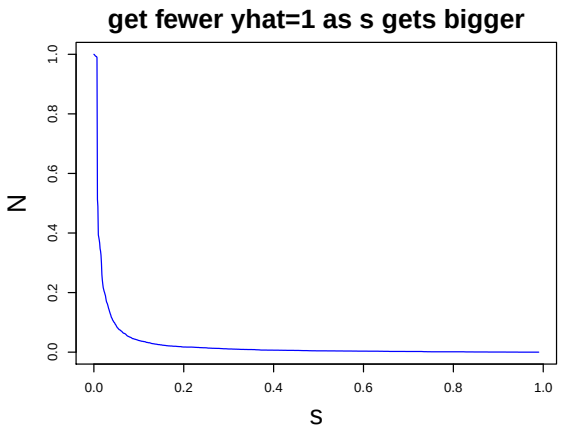
Lift looks at:

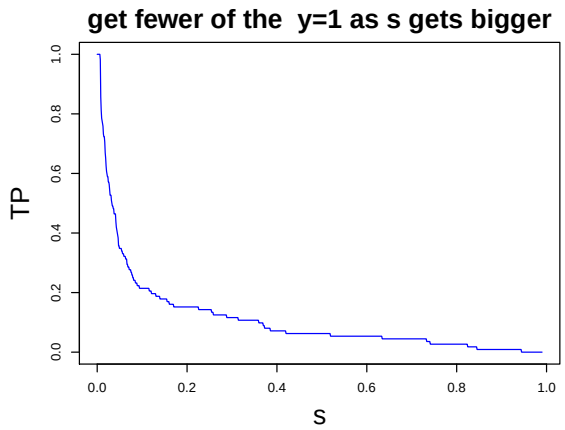
- TP (true positive), % of y=1 correctly classified: .62.
- N, % classified as 1: (1058+69)/(total number) = 1127/5000 = .2258.

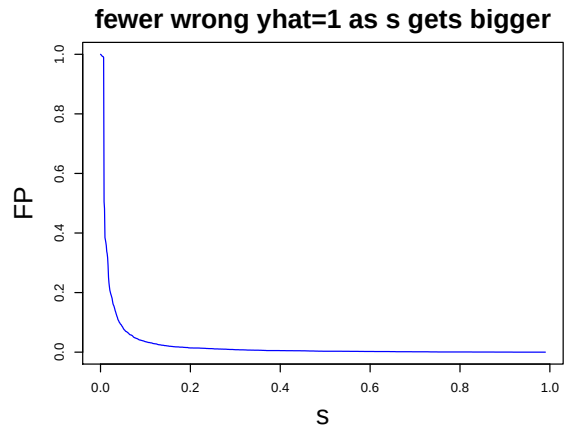
TP: fraction of 1's out there you got.

N: fraction of data you tried.

We then just compute these values for s between 0 and 1 and plot them.





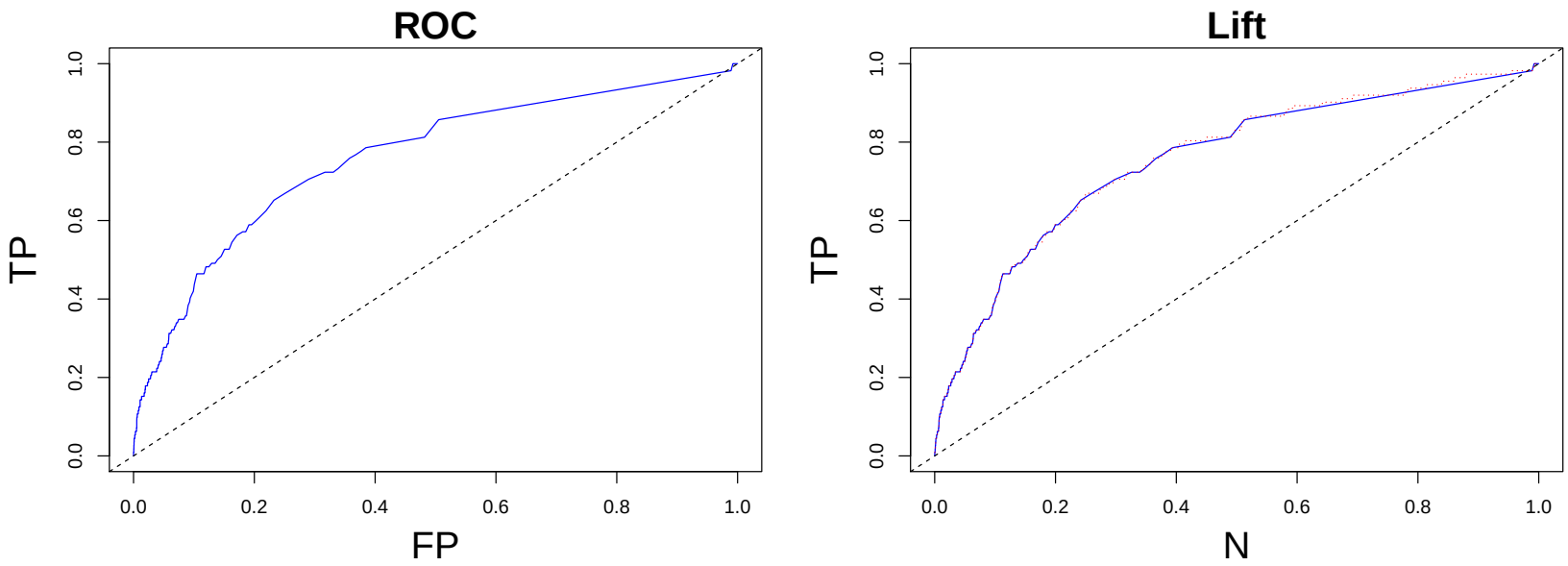


$$\text{classify as } 1 \Leftrightarrow P(Y = 1 \mid x) > s.$$

- TP (true postive), % y=1 correctly classified
- FP:(false positive), % y=0 incorrectly classified
- N, % classified as 1.

All three quantities go from 1 to 0, as s goes from 0 to 1.

ROC plots FP vs. TP, and lift plots N vs. TP.
Want TP big and FP small.



Note that as you go from left to right in these plots, s is decreasing.

The line is drawn at "y=x".
It represents the performance you would get if Y was independent of X.