STAT 1291: Data Science

Lecture 19 - Statistical modeling III: Classification and Model Evaluation

Sungkyu Jung

Where are we?

- *data visualization*
- *data wrangling*
- *professional ethics*
- *statistical foundation*
- *Statistical modeling: Regression*
- *Cause and effect: Causality and confounding*
- More statistical modeling: Machine learning

Classification

Classifiers are an important complement to regression models in the fields of machine learning and predictive modeling.

Whereas regression models have a quantitative response variable, classification models have a categorical response.

Example: The Obama-Clinton divide

- The following graphic, published in New York Times, by Amada Cox, in 2008 gained popularity for its beautiful use of graphics in displaying a statistical analysis.

  http://www.nytimes.com/imagepages/2008/04/16/us/20080416_OBAMA_GRAPHIC.html

- We see an entire story between Obama and Clinton - positions taken, counties won, and counties lost.
- This is an example of classification: For each given county, using measurements of the county (e.g. %black population, %high-school grad, geographic location, etc), you can predict whether Clinton wins or not.

The data: 2008 Democratic primaries

```
primary <- read.csv(url("http://www.stat.pitt.edu/sungkyu/c
primary <- as_tibble(primary)
```

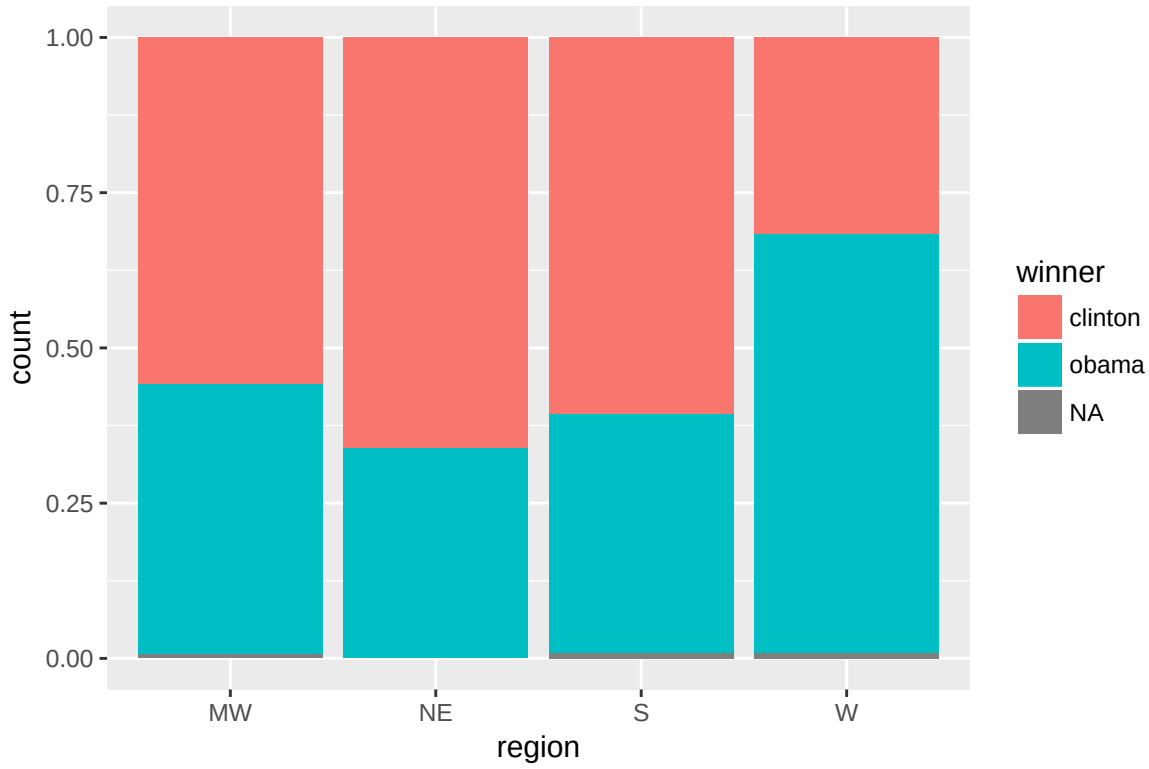
- The decision tree was built from a data set concerning all the counties in states where primaries had already been held.
- The unit of observation is a county and variables included various demographic measures (age and ethnic makeup, education level, religious breakdown), political measures (did the county go to Bush or Kerry in 04) and economic factors (unemployment rate, the amount of construction in the county), and so on.

Whenever we encounter a new dataset, it is a good practice to first visualize the data.

```
glimpse(primary) # demo only
```

- To see if the categorical variable `region` is a good indication of `winner`, we want to look at the conditional distribution of `winner`, given by different `region` levels:

```
primary %>% ggplot(aes(x = region, fill = winner)) +
  geom_bar(position = "fill")
```



Data transformation

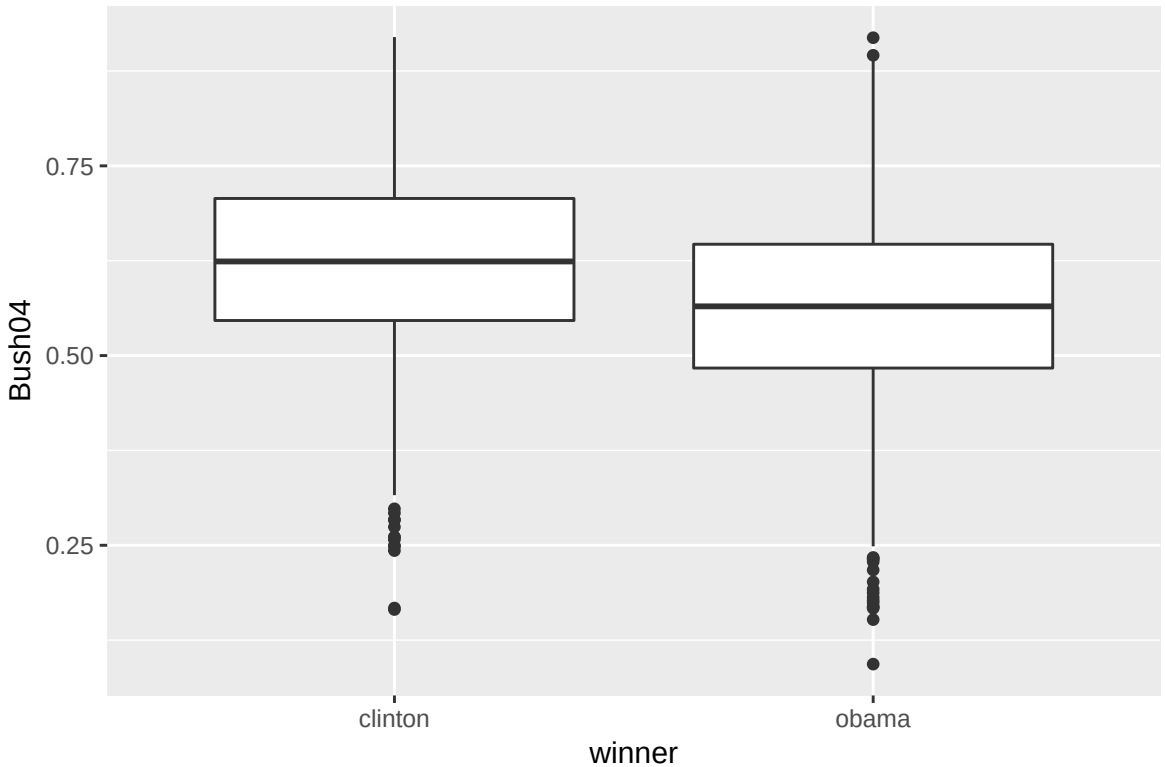
Some counties do not have a winner yet. Let us exclude these counties.

```
primary <- primary %>% filter(!is.na(winner))
```

- To see if the continuous variable `Bush04` is a good indication of `winner`, we want to look at the conditional distribution of `Bush04`, given by different `winner` levels.

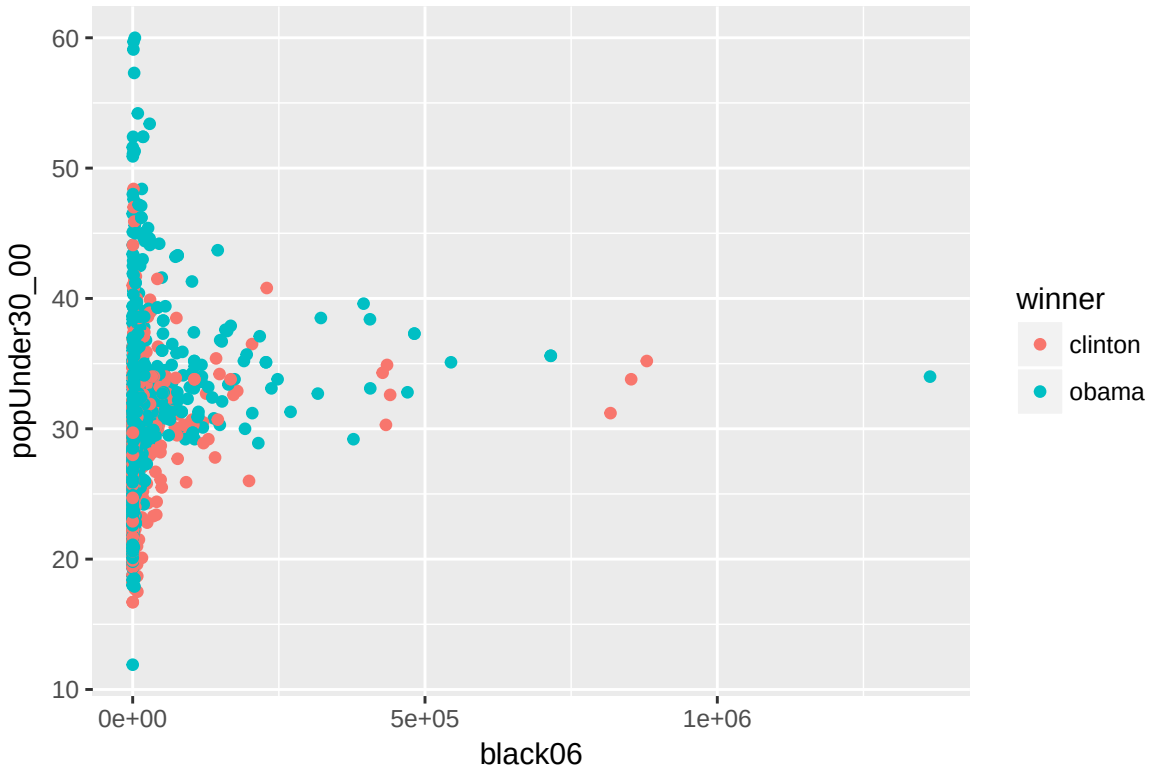
```
primary %>% ggplot(aes(x = winner, y = Bush04)) +
  geom_boxplot()
```

```
## Warning: Removed 1 rows containing non-finite values (st
```



- To see if the continuous variables `black06` and `popUnder30_00` are a good indication of `winner`, the best practice is to use the scatterplot with the `income` information given by the color cue.

```
primary %>% ggplot(aes(x = black06, y = popUnder30_00, colo
```



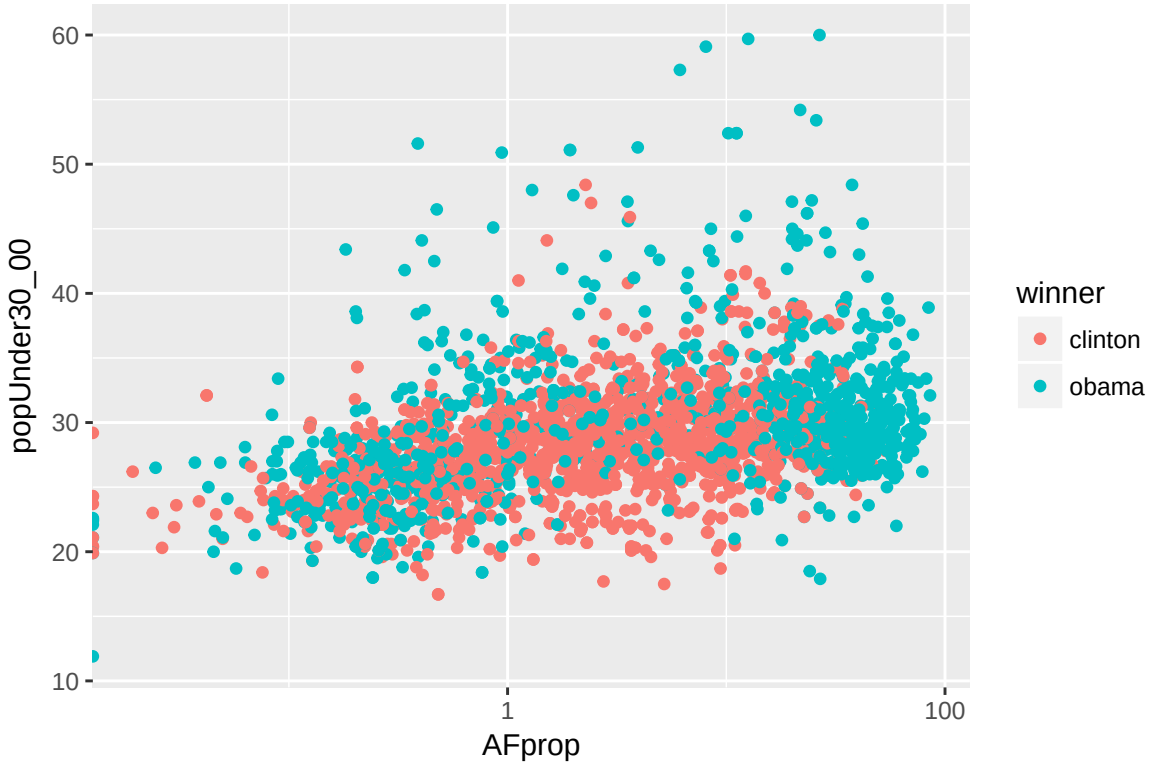
Data transformation

Notice that the coordinate scales do not commensurate; some counties have large populations.

Mutate a variable for the percent of African American population. Also remove some missing values

```
primary <-
  primary %>%
  mutate(AFprop = black06 / pop06 * 100) %>%
  filter(!is.na(AFprop))
```

```
primary %>% ggplot(aes(x = AFprop, y = popUnder30_00,
                       color = winner)) +
  geom_point() + scale_x_log10()
```



Data Science workflow

A typical data science project looks something like this:

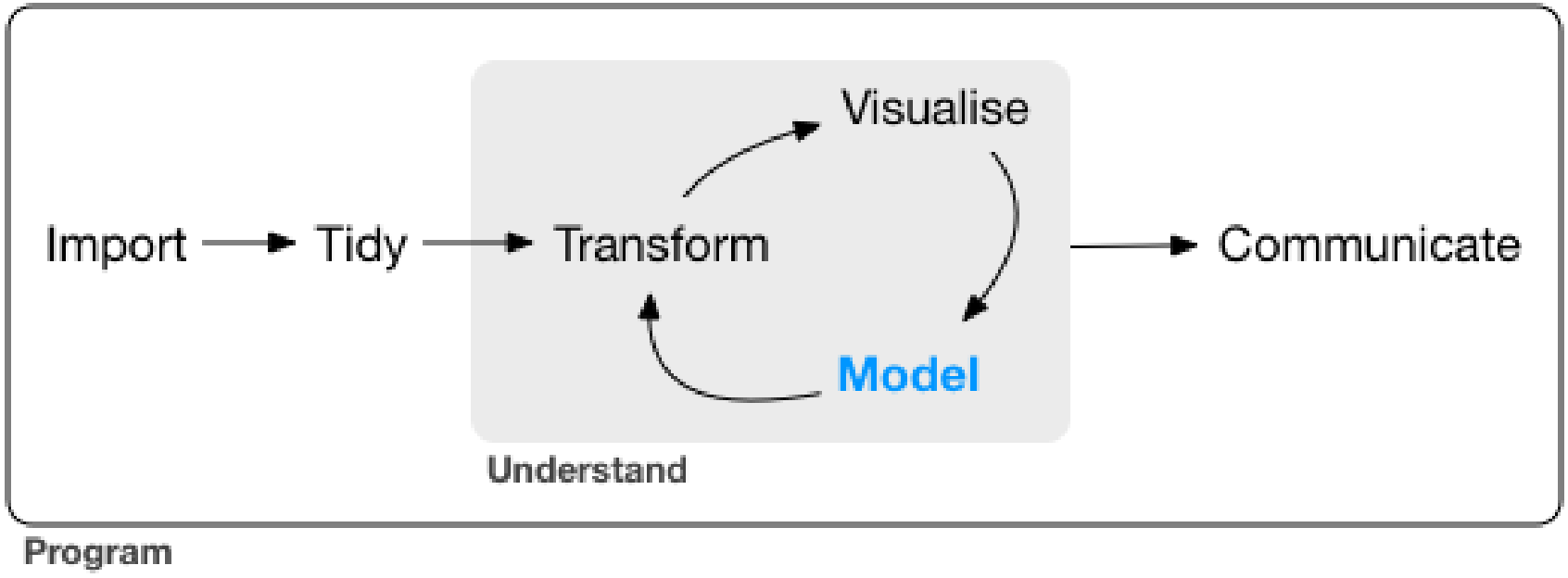


Figure 1: (from r4ds)

From exploratory analysis

- Some variables (when transformed) are logically predicting the winner
- Some make little sense to use.
- A combination of variables seems to work better than using a single variable.
- Overall, it seems reasonable to predict the winner using data.
- We will fit

$$\text{Winner} \sim f(\text{some variables})$$

using two families of models.

Classification methods

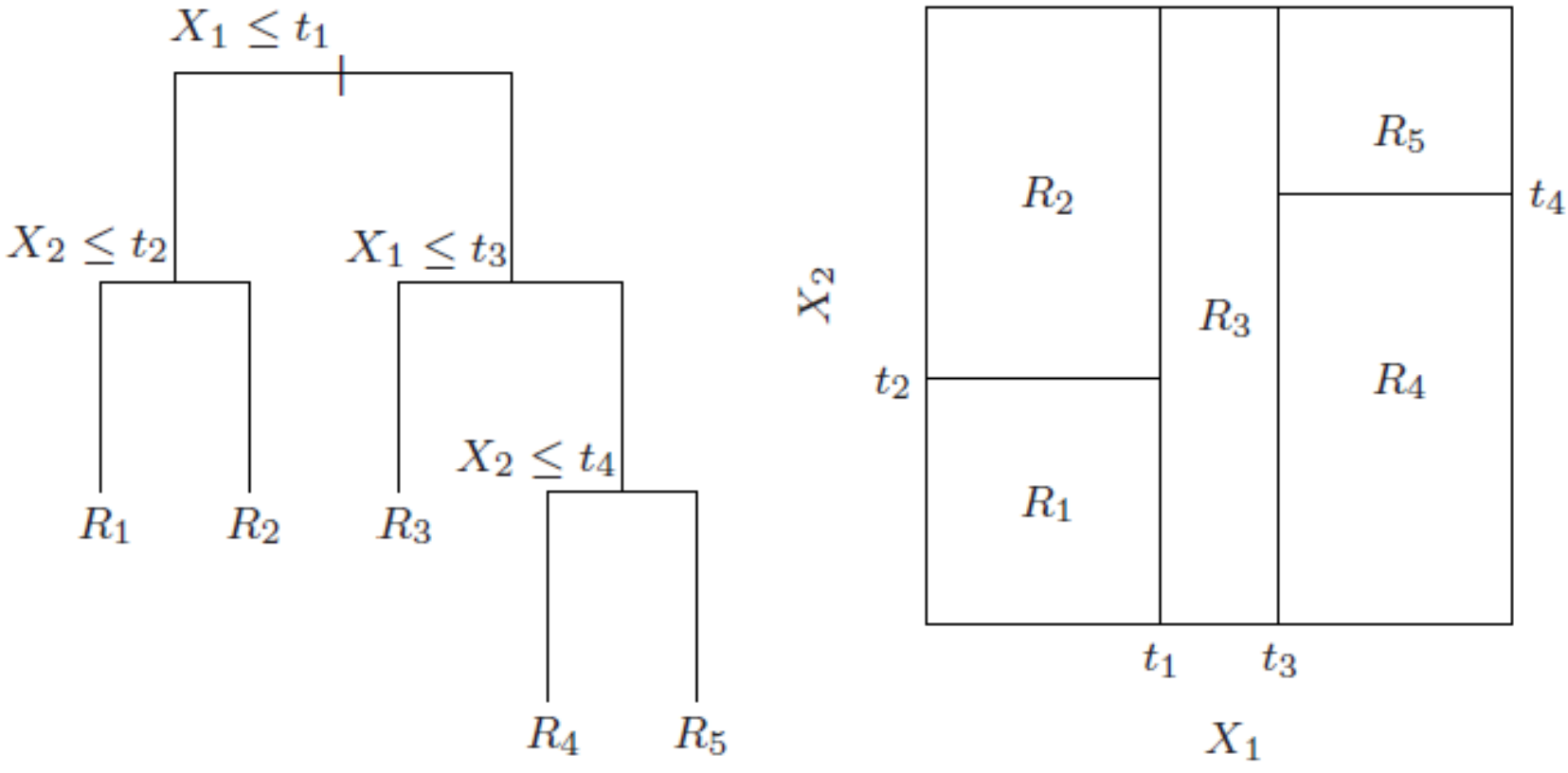
There are probably thousands of classification methods proposed to date. We will see a couple of popular methods.

1. Decision tree
2. Logistic regression

1. Decision tree

A decision tree is a tree-like flowchart that assigns class labels to individual observations. Each branch of the tree separates the records in the data set into increasingly "pure" (i.e., homogeneous) subsets, in the sense that they are more likely to share the same class label.

A tree with two numeric variables X_1 and X_2



Each subregion must be increasingly "pure".

How do we construct these trees?

- The number of possible decision trees grows exponentially with respect to the number of variables p.
- Computationally finding the optimal tree is impossible.
- There are several competing heuristics for building decision trees that employ greedy (i.e., locally optimal) strategies.
- We will simplify our presentation by restricting our discussion to *recursive partitioning* decision trees.
- The R package that builds these decision trees is accordingly called `rpart`.

Below, we simply use two variables AFprop and popUnder30_00 to predict winner.

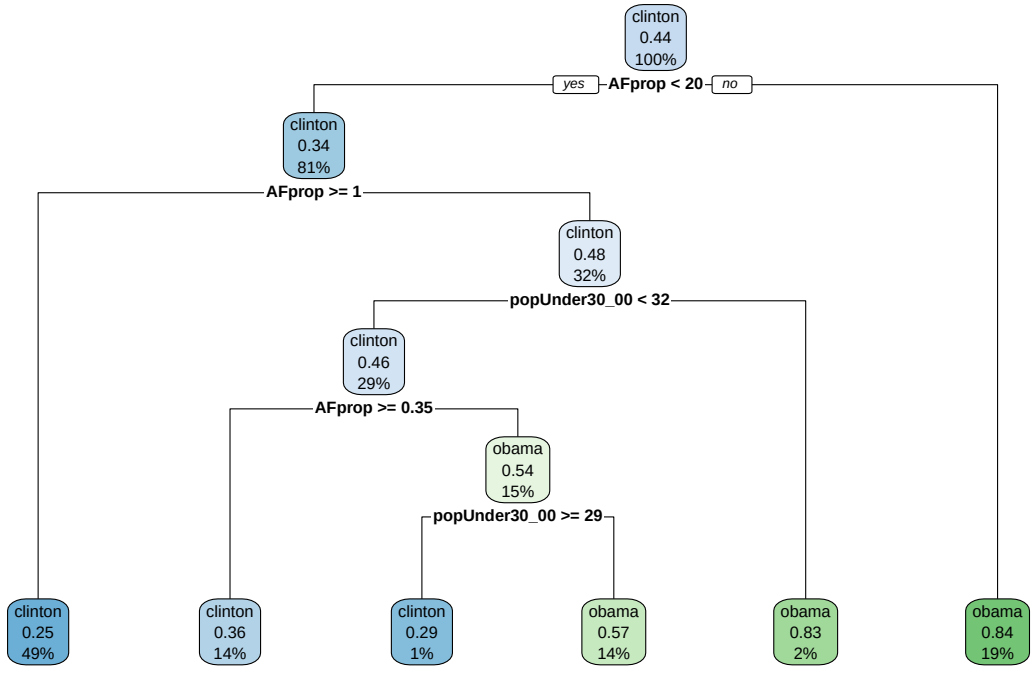
```
library(rpart)  # for computation of recursive partitioning
library(rpart.plot) # for plotting trees
```

```
## Warning: package 'rpart.plot' was built under R version
```

```
mod_tree1 <- rpart(winner ~ AFprop + popUnder30_00,
                   data = primary)
```

Use `rpart.plot()` to visualize the fitted decision tree.

```
rpart.plot(mod_tree1)
```

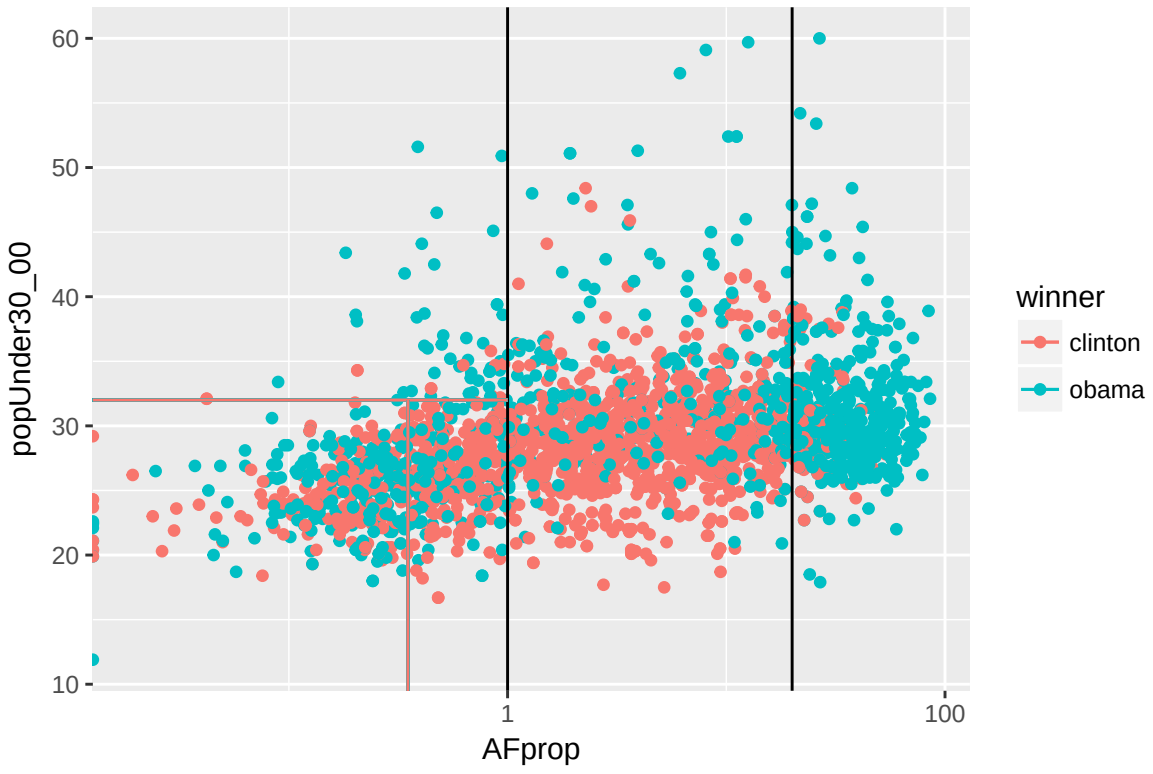


Each node shows

- the predicted class (Clinton or Obama),
- the predicted probability of Obama,
- the percentage of observations in the node.

Daughter nodes are branched by a decision involving a single variable.

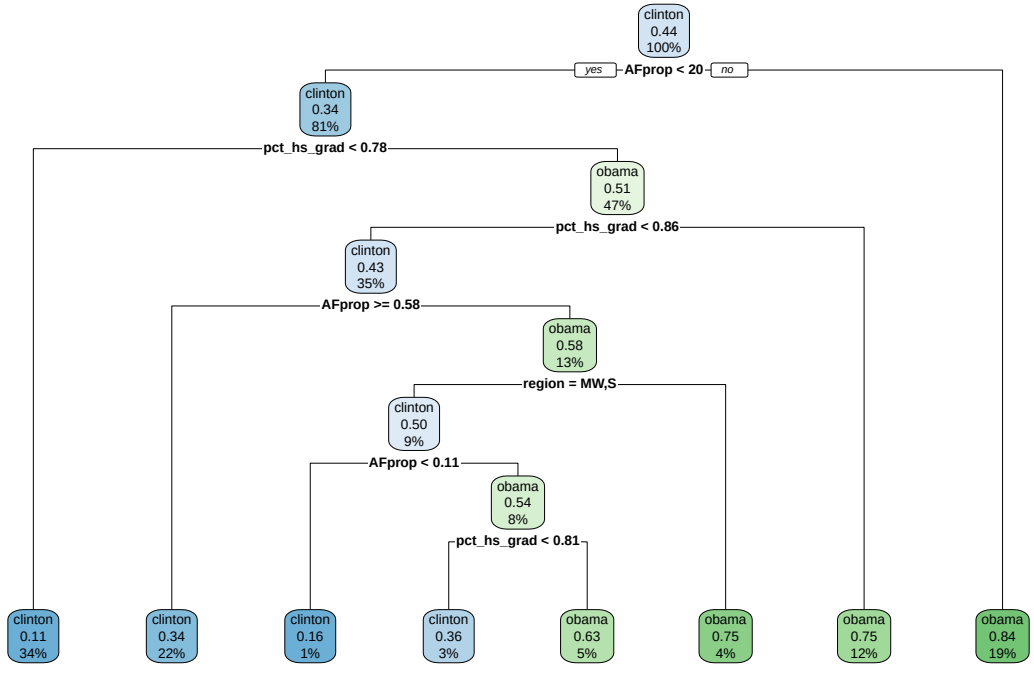
Decision tree with data



Another model

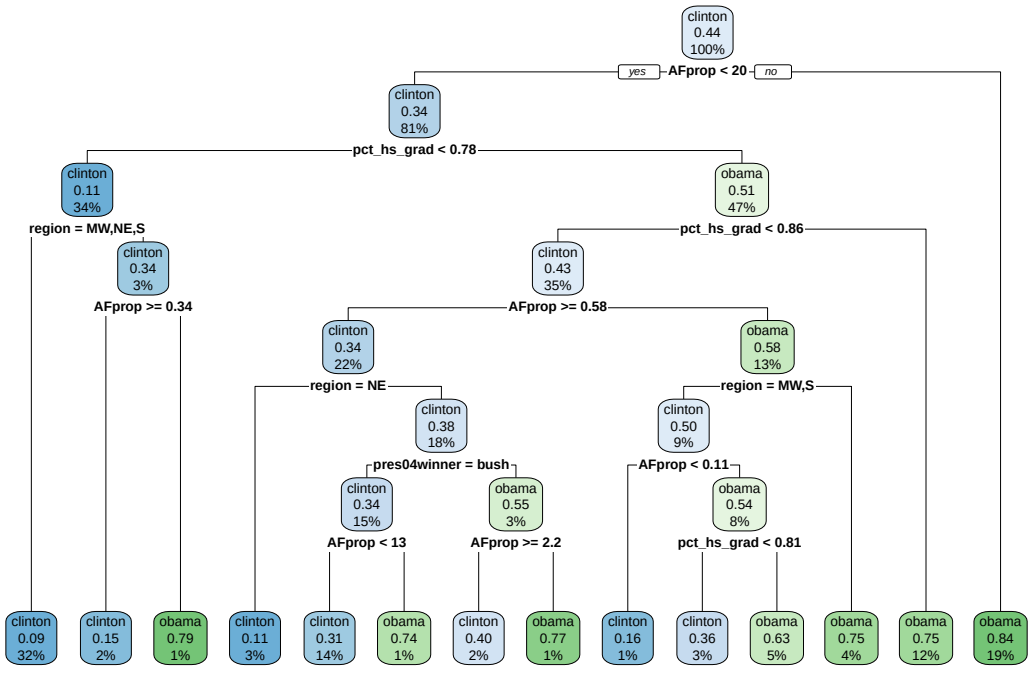
Note that the model formula is saved in an object `form`. We will reuse this object.

```
form <- as.formula("winner ~ AFprop + region + pres04winner
mod_tree2 <- rpart(form, data = primary)
rpart.plot(mod_tree2)
```



Yet another model

```
mod_tree3 <- rpart(form, data = primary, control = rpart.co
rpart.plot(mod_tree3)
```



Which model is better?

- Range of models is given by
  - sets of variables used
  - complexity of the model (depth of the tree)
- How do you assess each model?
  - By the classification performance (one of many possible choices)

Measuring classification performance

1. Confusion Matrix

```
Confusion_matrix3 <- primary %>%
  mutate(pred3 = predict(mod_tree3, type = "class")) %>%
  select(winner,pred3) %>% table()
Confusion_matrix3
```

```
##           pred3
## winner     clinton obama
##   clinton    1133   236
##   obama       234   826
```

2. Misclassification error rate

Model 3 gives the misclassification rate of 0.1934953.

Never use the same data to fit and assess the model

- Prone to overfit
- Model 2 gives the misclassification rate of 0.211198, Slightly worse than that of Model 3. But Model 2 is much simpler.

Model evaluation 1: Training and testing data

If the model works well on those training data, but not so well on a set of testing data, that the model has never seen, then the model is said to be *overfit*.

Perhaps the most elementary mistake in predictive analytics is to overfit your model to the training data, only to see it later perform miserably on the testing set.

In predictive analytics, data sets are often divided into two sets:

- **Training** The set of data on which you build your model
- **Testing** Once your model is built, you test it by evaluating it against data that it has not previously seen.

Training vs testing

- Randomly divide the data set into a training set and a testing set

```
library(dplyr)
set.seed(1)
train <- primary %>% sample_frac(size = 0.8)
test <- primary %>% setdiff(train)
```

Check that there is no overlap between the two data sets:

```
nrow(intersect(train,test))
```

```
## [1] 0
```

- Build models on training data

```
mod1 <- rpart(form2, data = train,
              control = rpart.control(cp = 0.005))
mod2 <- rpart(form2 , data = train,
              control = rpart.control(cp = 0.0001))
```

- Evaluate the misclassification rate on testing data

```
tibble(
  model = c("1","2"),
  testerror = c(mcr(test,mod1), mcr(test,mod2)),
  trainerror = c(mcr(train,mod1), mcr(train,mod2)))
```

```
## # A tibble: 2 x 3
##   model testerror trainerror
##   <chr>     <dbl>      <dbl>
## 1     1 0.1851852  0.1595471
## 2     2 0.1893004  0.1101390
```

- It is possible that training error is small, but the test error is large; an evidence of overfit

2. Logistic regression as a classifier

- A simple-minded understanding of logistic regression is to predict $y \in \{0, 1\}$ using $x_1, \ldots, x_p$ by a formula

$$y \sim \phi(x_1, \ldots, x_p) = \phi(\beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p)$$

- In fact, a logistic regression model estimates the probability p of $y = 1$, given the values of x.
- For prediction of y, we will round $0 \leq p \leq 1$ to either 0 or 1.

Logistic regression

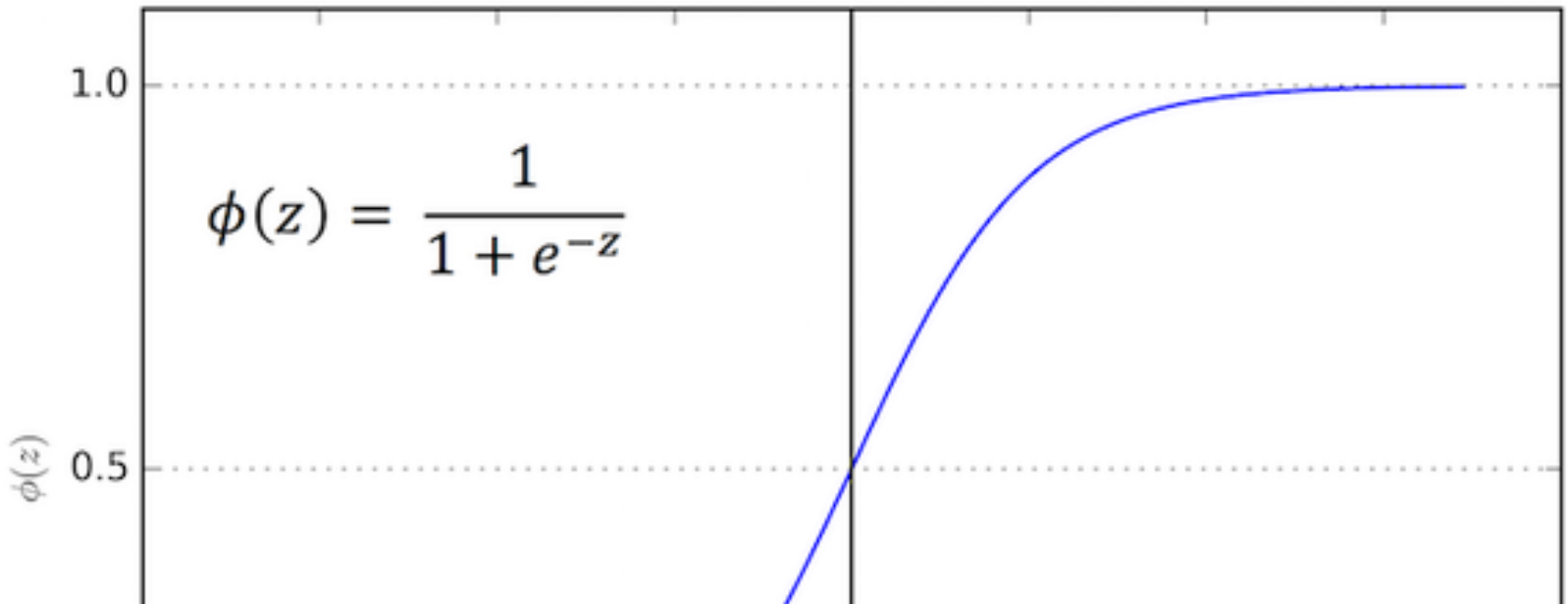
- Explanatory variables to an intermediate predictor z:

$$(x_1, \ldots, x_p) \rightarrow z = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$$

- z to estimated probability p:

$$p = \phi(z)$$

- Round p to either 0 or 1 to predict y (or classify into "0" or "1")



Example: Logistic regression

- Use `glmnet` package by Friedman, Hastie, Simon, Qian and Tibshirani to numerically compute the fitted coefficients (βs) of the logistic regression.
- `model.matrix()` transforms the categorical variables in the `data` into appropriate numeric variables (by creating *dummy* variables if needed).
- Logistic regression is a special case of Generalized Linear Models, where the response is binary (Clinton vs Obama, in this case). To indicate this, in the argument of `glmnet()`, we specify `family = "binomial"`.

```
library(glmnet)
form <- as.formula(
  "winner ~ AFprop + region + pres04winner")
predictors <- model.matrix(form, data = train)
fit1 <- glmnet(predictors, train$winner,
               family = "binomial",
               lambda = 0)
```

- We will discuss the argument `lambda = 0` shortly. Ignore for now.

Logistic regression fitted to primary data

```
fit1$beta
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                                s0
## (Intercept)          .
## AFprop               0.1040414
## regionNE            -1.1548498
## regionS             -1.6667265
## regionW              1.0794071
## pres04winnerkerry    0.2250285
```

In the print-out above, ignore the first two lines. They are related to the numerical problem of efficiently handling the fitted values.

- This logistic model gives the testing misclassification rate of 0.261316872427984.

Regularized logistic regression

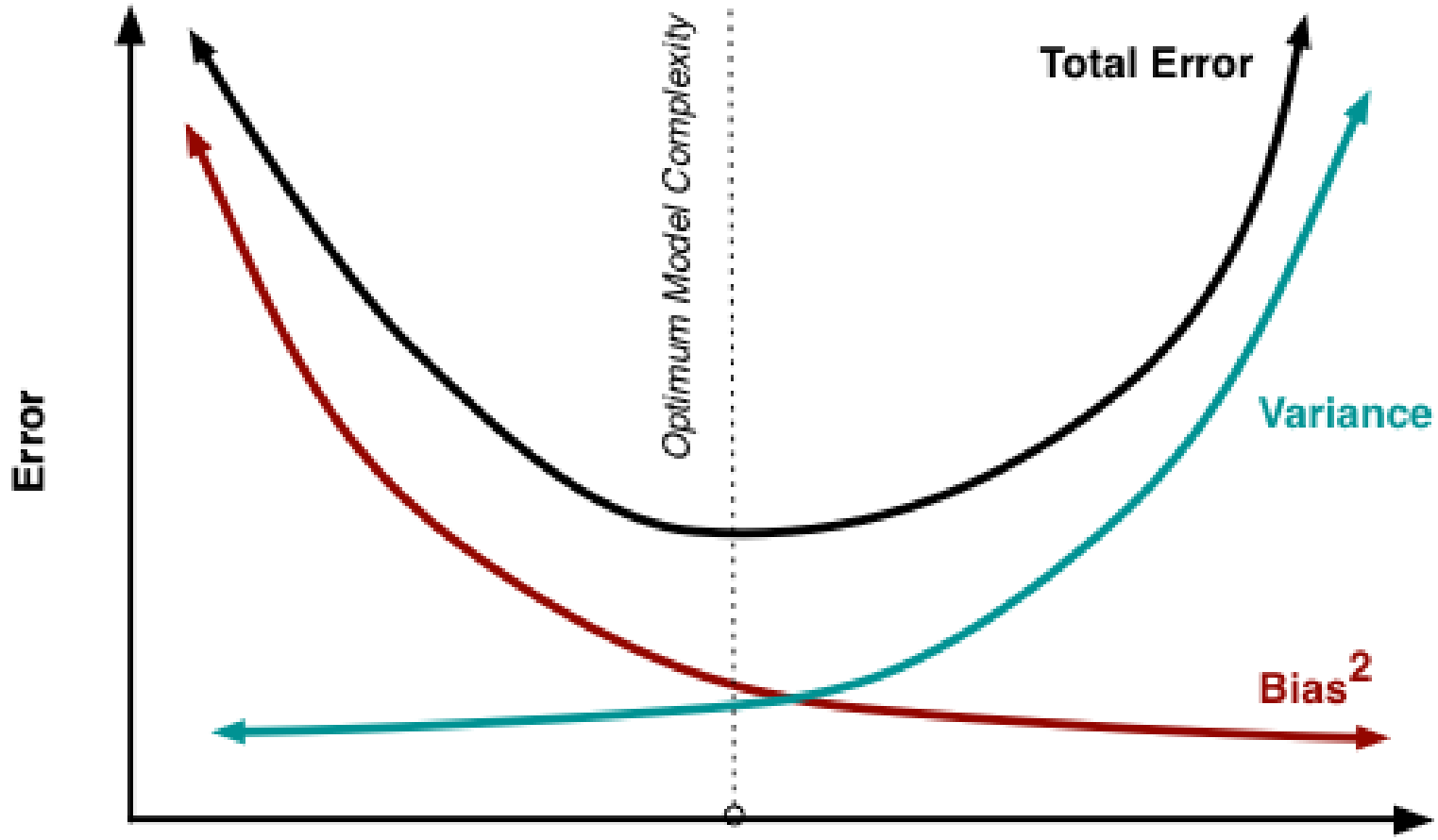
A regularized logistic regression finds the values of β's by minimizing the following:

$$\text{Variance} + \lambda \text{Bias}$$

This is often referred to as "Loss + Penalty". The latter is correct, and the former is a bit misleading, but let us focus on the idea for now.

Variance–Bias trade-off

- Ordinary logistic regression minimizes the "variance"
- Regularized logistic regression introduced a "bias" in hope for minimizing the total error



Elastic Net logistic regression

- the "Variance" is in fact a *negative goodness-of-fit measure*:

$$GF(\beta) = -\left[\frac{1}{N}\sum_{i=1}^{N} y_i \cdot (\beta_0 + x_i^T \beta) - \log(1 + e^{(\beta_0 + x_i^T \beta)})\right]$$

- the "Bias" is in fact the size of β:

$$Size(\beta) = \left[(1-\alpha)||\beta||_2^2/2 + \alpha||\beta||_1\right]$$

Elastic Net logistic regression amounts to the optimization of the form:

$$\min_{(\beta_0,\beta)\in\mathbb{R}^{p+1}} -GF(\beta) + \lambda Size(\beta)$$

What is λ?

- In the regularized regression, λ is called a **tuning parameter**:

  That is, the result of analysis is tunable by changing the value of λ.

- Large λ forces to set many β equal to 0:

```
fit2 <- glmnet(predictors, train$winner,
               family = "binomial", lambda = 0.05) # lambda
fit2$beta
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                              s0
## (Intercept)          .
## AFprop               0.04864433
## regionNE             .
## regionS             -0.49085299
## regionW              0.70316716
```

Classification by regularized logistic regression

- The question of which variables to model is now replaced by how to choose a value of λ?

lambda	Testing Misclassification Rate
0 (Ordinary logistic regression)	0.261316872427984
0.05 (Regularized logistic regression)	0.253086419753086

Model evaluation 2: Cross-validation

- So far, we set aside 80% of the observations to use as a training set, but held back another 20% for testing.
- Another approach is cross-validation.
- The idea of cross-validation is similar to that of bootstrap: Repeat the train-testing subsampling K times.

K-fold cross validation

- To perform a 2-fold cross-validation:

1. Randomly split your data (by rows) into two data sets with the same number of observations. Call them X1 and X2.
2. Build your model on the data in X1, and then run the data in X2 through your model. How well does it perform? Just because your model performs well on X1 (this is known as in-sample testing), does not imply that it will perform as well on the data in X2 (out-of-sample testing).
3. Now reverse the roles of X1 and X2, so that the data in X2 is used for training, and the data in X1 is used for testing.

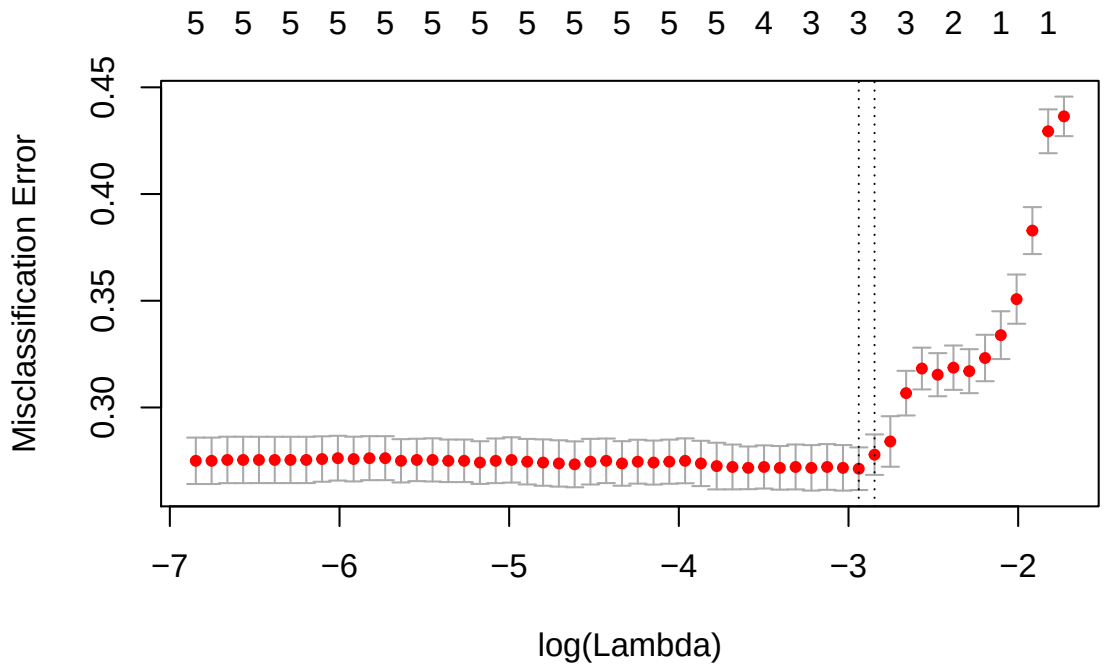
- Typically K is chosen to be 10, not 2.

Cross-validation of model fit against λ

- The package `glmnet` provides a handy cross-validation function `cv.glmnet()`.
- In the argument of `cv.glmnet()`, the option `type = "class"` is used to specify that the misclassification error is used as a measure of comparison.

```
predictors <-
  model.matrix(form, data = primary) # use whole data
cv.fit <-
  cv.glmnet(predictors, primary$winner,
                        family = "binomial", type = "class")
```

```
plot(cv.fit)
```


5 5 5 5 5 5 5 5 5 5 5 5 4 3 3 3 2 1 1
Misclassification Error
0.30
0.35
0.40
0.45
−7
−6
−5
−4
−3
−2
log(Lambda)

Summary: Classification and model evaluation

- Two popular classification methods: Decision tree, regularized logistic regression
- Both methods are *tunable*:
  - In decision tree, model complexity = the level of tree
  - In logistic regression, choice of lambda = variable selection
- To "tune" the tuning parameters, and to assess the fitted model, we use partitioned data: Training-Testing split or cross-validation
- Choice between decision tree and logistic regression can be similarly using the data.

Take away message

We have seen the tip of the iceberg. However, the idea is universal:

- Variance-bias trade off
  - Complex models: small bias, large variance
  - Simple models: large bias, small variance
- Compete for a goodness-of-fit measure (e.g. misclassification rate)
- Never use the same data for fitting and assessment