Resampling Methods

Resampling

Def: *Draw many samples from the training data and refit the model to each in order to learn about your model.*

Cross-Validation

Methods to estimate the test error rate (MSE, misclassification error) by

- holding out a subset of the training data from the fitting process
- using that model to predict on the subset

Three Flavors

1. Validation Set
2. Leave-one-out
3. k-fold

Validation Set

1. Randomly split data into a *training set* and a *validation set*.
2. Fit model to *training set*.
3. Use model to predict responses for *validation set*.
4. Compute validation set error rate as estimate of test error rate.

Let's look back on how we compared Logistic Regression to LDA . . .

Split data into training/validation

```r
library(ISLR)
data(Default)
set.seed(391)
test_ind <- sample(1:10000, size = 5000)
Default_test <- Default[test_ind, ]
Default_train <- Default[-test_ind, ]
```

Fit models to train

```
# Logistic
m1 <- glm(default ~ balance,
          data = Default_train,
          family = binomial)

# LDA
library(dplyr)
est <- Default_train %>%
  group_by(default) %>%
  summarize(n = n(),
            prop = n/nrow(Default_train),
            mu = mean(balance),
            ssx = var(balance) * (n - 1))
```

Fit models to train, cont.

Pull off estimates (and convert class).

```
pi_n <- as.numeric(est[1, 3])
pi_y <- as.numeric(est[2, 3])
mu_n <- as.numeric(est[1, 4])
mu_y <- as.numeric(est[2, 4])
sig_sq <- (1/(nrow(Default_train) - 2)) *
  sum(est$ssx)
```

Make predictions on validation

```
log_pred <- predict(m1,
                    newdata = Default_test,
                    type = "response")

my_lda <- function(x, pi, mu, sig_sq) {
  x * (mu/sig_sq) - (mu^2)/(2 * sig_sq) + log(pi)
}

d_n <- my_lda(Default_test$balance,
              pi_n,
              mu_n,
              sig_sq)
d_y <- my_lda(Default_test$balance,
              pi_y,
              mu_y,
              sig_sq)

my_log_pred <- ifelse(log_pred < 0.5, "No","Yes")
my_lda_pred <- ifelse(d_n > d_y, "No", "Yes")
```

Misclassification Rates

```r
conf_log <- table(my_log_pred,
                  Default_test$default)
conf_lda <- table(my_lda_pred,
                  Default_test$default)
(conf_log[2, 1] + conf_log[1, 2])/
  nrow(Default_test)
```

```
## [1] 0.0238
```

```r
 (conf_lda[2, 1] + conf_lda[1, 2])/
  nrow(Default_test)
```

```
## [1] 0.0244
```

Logistic has the lower test error rate.

Let's do that again

How would have our conclusions changed if we had used a different partition into training and validation sets?

```
test_ind <- sample(1:10000, size = 5000)
Default_test <- Default[test_ind, ]
Default_train <- Default[-test_ind, ]
```

(Re-run all that code)

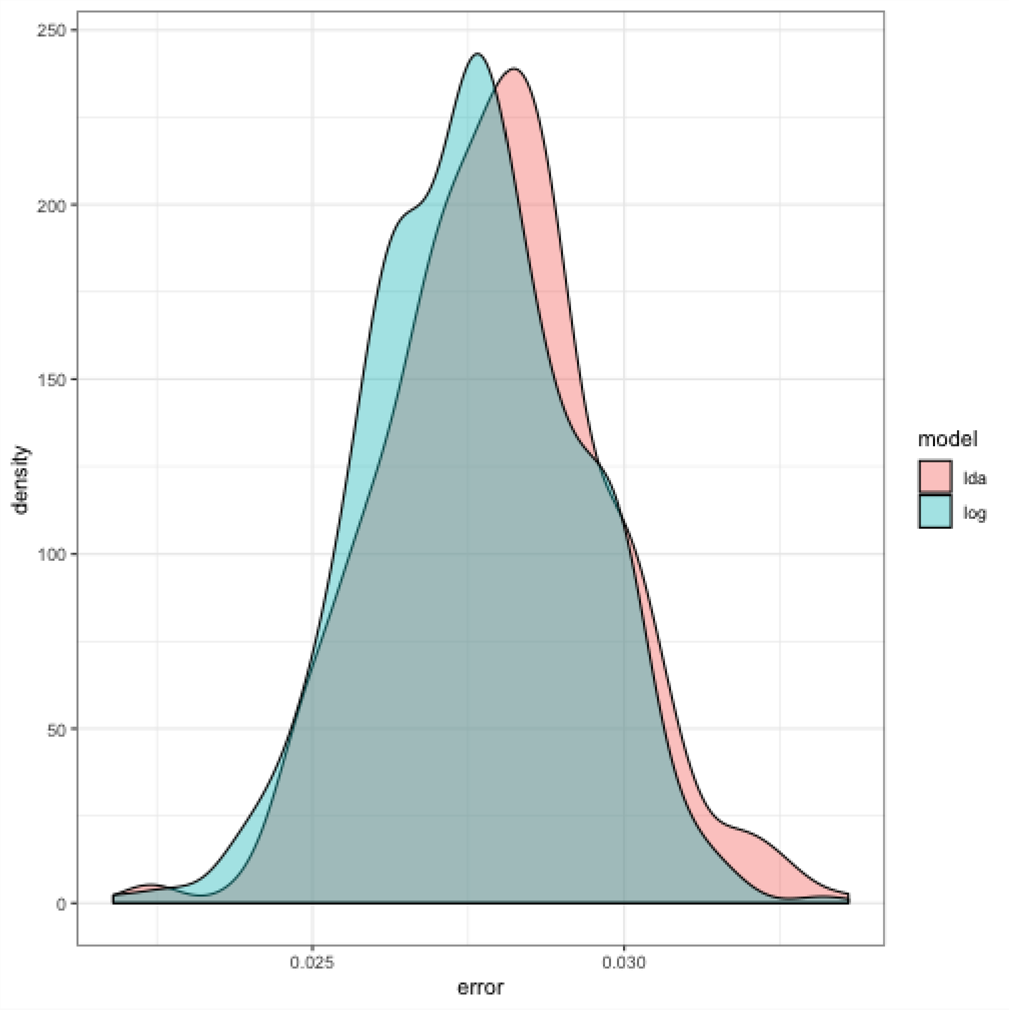
```
(conf_log[2, 1] + conf_log[1, 2])/
  nrow(Default_test)
```

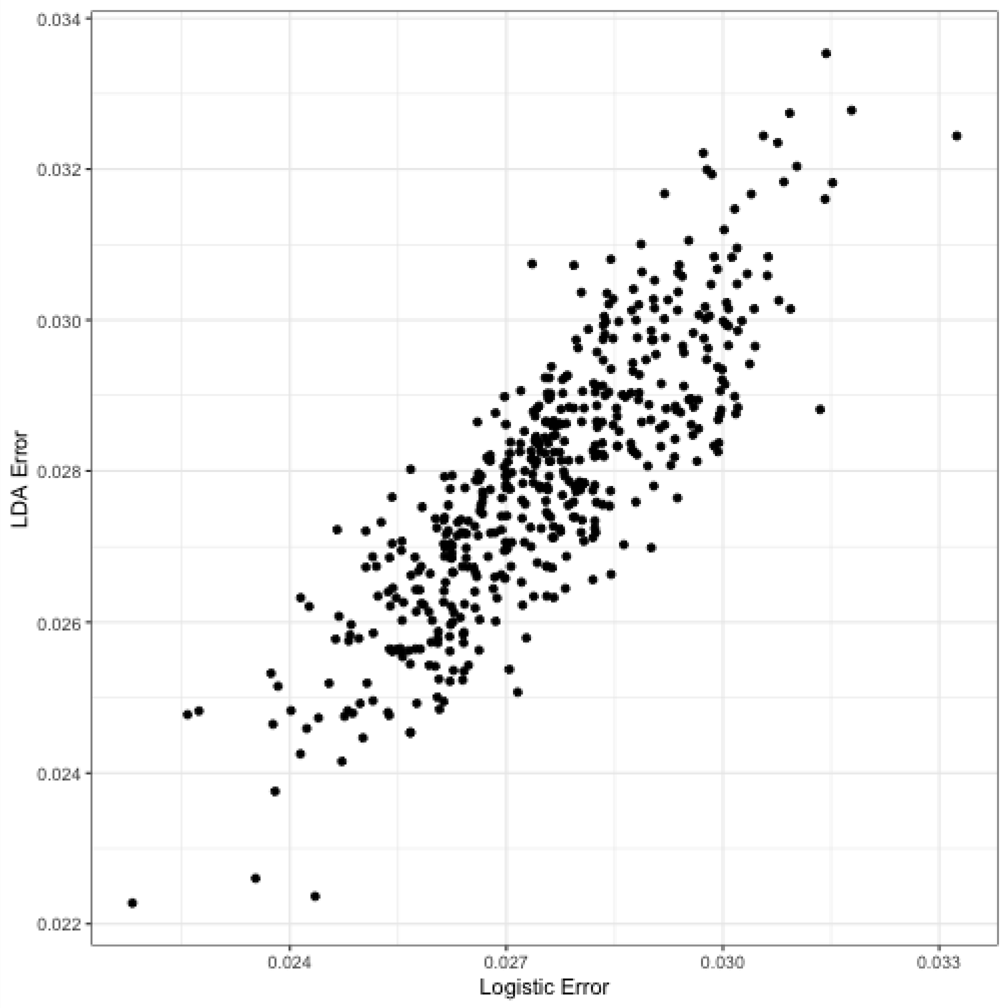
```
## [1] 0.0294
```

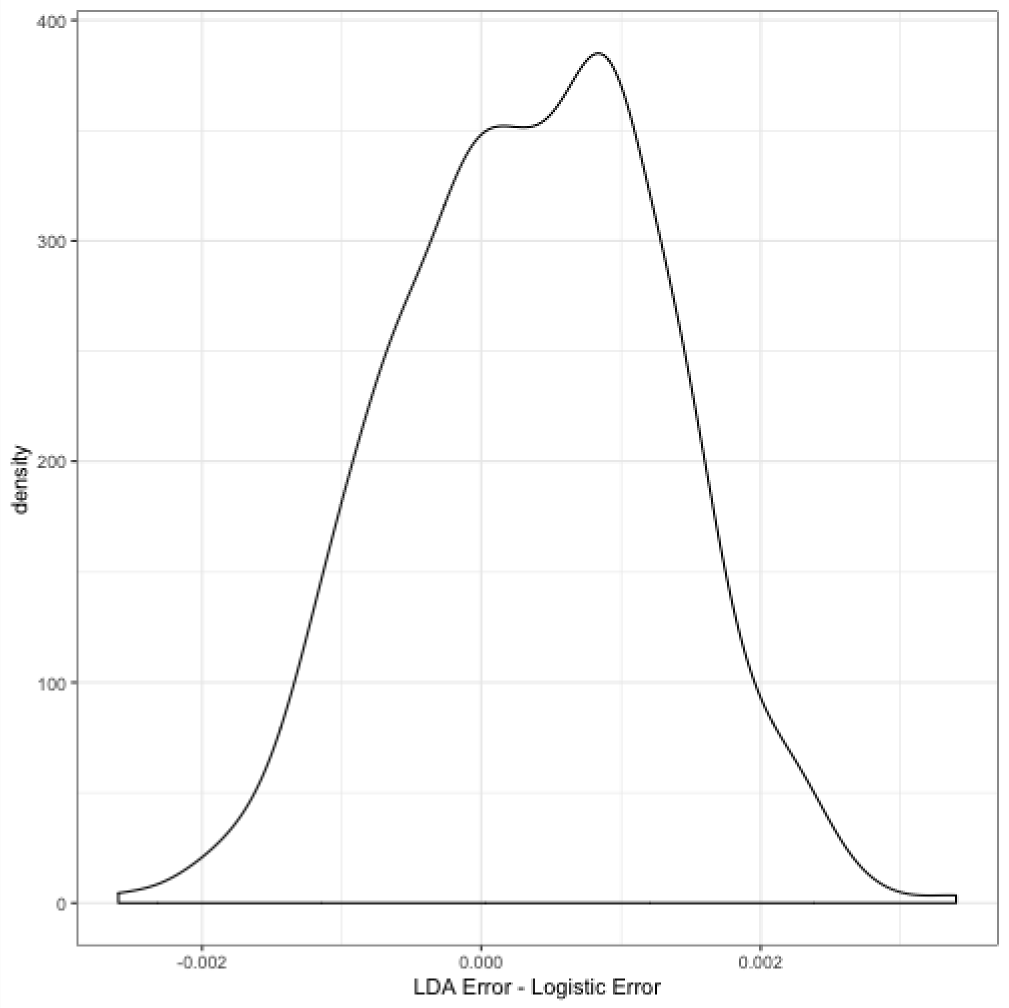
```
 (conf_lda[2, 1] + conf_lda[1, 2])/
  nrow(Default_test)
```

```
## [1] 0.0298
```

Let's do that many times




0.034
0.032
0.030
0.028
0.026
0.024
0.022
LDA Error
0.024
0.027
0.030
0.033
Logistic Error



400
300
200
100
0
density
-0.002
0.000
0.002
LDA Error - Logistic Error

Validation Set

Downsides

1. Estimate of test error rate can be highly variable based on the partition.
2. You only use a fraction of the data to fit the model > overestimating test error rate.

Leave-one-out