Lecture 4

Last time

- Threshold predicted probabilities to get binary predictions
- Performance metrics like accuracy, sensitivity, and specificity can be calculated from a confusion matrix
- A threshold of 0.5 maximizes accuracy (in the population)
- As threshold increases, sensitivity decreases and specificity increases
- ROC curves plot the trade-off between sensitivity and specificity

Class activity

- Take some time to work through the class activity
- You are welcome to work in groups

https://sta712-f23.github.io/class_activities/ca_lecture_3.html

Class activity: dengue data

```
m1 <- glm(Dengue ~ Age + WBC + PLT, data = dengue,
          family = binomial)
m2 <- glm(Dengue ~  Age + WBC + PLT + BMI + HCT + Temperature,
          data = dengue, family = binomial)
```

How do I perform a LRT to compare the two models?

Class activity: dengue data

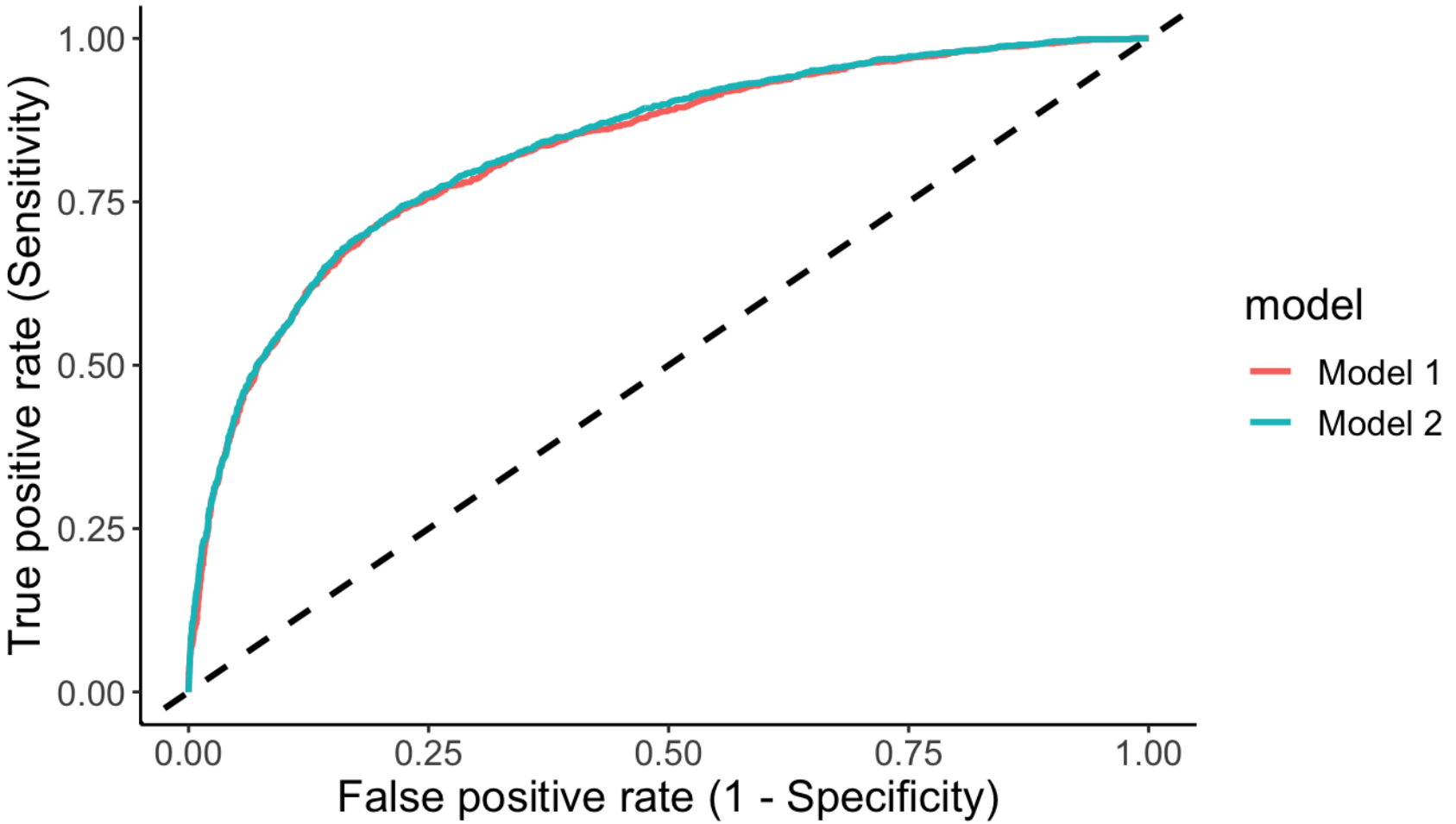
```r
m1 <- glm(Dengue ~ Age + WBC + PLT, data = dengue,
          family = binomial)
m2 <- glm(Dengue ~  Age + WBC + PLT + BMI + HCT + Temperature,
          data = dengue, family = binomial)

pchisq(m1$deviance - m2$deviance, 3, lower.tail=F)
```

```
[1] 1.742259e-16
```

Which model would the LRT choose?

Class activity: dengue data



Which model would you choose?

Class activity: simulated data

```r
m1 <- glm(y ~ X[,1:3], family = binomial)
m2 <- glm(y ~ X, family = binomial)

m1$deviance
```

```
[1] 1294.233
```

```r
m2$deviance
```

```
[1] 1292.818
```

Why *must* the second model have a smaller deviance (on the data used to fit the model)?

Class activity: simulated data

```
pred1 <- prediction(m1$fitted.values, m1$y)
pred2 <- prediction(m2$fitted.values, m2$y)

performance(pred1, "auc")@y.values
```

```
[[1]]
[1] 0.5808683
```

```
performance(pred2, "auc")@y.values
```

```
[[1]]
[1] 0.5847598
```

Why might model 2 have a greater AUC?

Class activity: new simulated data

```r
# predictions on new observations
phat_m1 <- exp(m1$coefficients[1] + X_new[,1:3] %*% m1$coefficients[2
  1 + exp(m1$coefficients[1] + X_new[,1:3] %*% m1$coefficients[2:4])
)

phat_m2 <- exp(m2$coefficients[1] + X_new %*% m2$coefficients[2:7])/(
  1 + exp(m2$coefficients[1] + X_new %*% m2$coefficients[2:7])
)

# new deviance for model 1
-2*sum(y_new*log(phat_m1) + (1-y_new)*log(1 - phat_m1))
```

```
[1] 1321.424
```

```r
# new deviance for model 2
-2*sum(y_new*log(phat_m2) + (1-y_new)*log(1 - phat_m2))
```

```
[1] 1321.842
```

Why are the deviances higher than for the training data?

Key take-aways

- When evaluated on the *training* data, a large model will have a lower deviance than any of its sub-models
  - Prediction metrics like AUC are also often higher, even if a reduced model is correct
- Often prefer the simpler model (easier to interpret, less variability, etc.) if model performance is similar, *even if* a hypothesis test would choose the larger model
- We expect model performance (deviance, AUC, etc.) to be better on training data than on a new test set

Training vs. testing data

- Models generally perform better on their *training* data (the data used to fit the model) than on new (*test*) data.
- When evaluated only on training data, larger models tend to look better

How should we assess and compare model performance if we can't sample new data (e.g., in the dengue scenario)?

Data splitting

How should we assess and compare model performance if we can’t sample new data (e.g., in the dengue scenario)?

- Randomly divide available data into two groups: training and test
  - E.g. 70% training, 30% test
- Fit the model on the training sample
- Evaluate the model on the test sample

Data splitting with the dengue data

```r
# create training and test splits
train_sample <- sample(1:nrow(dengue), 0.7*nrow(dengue), replace = F)
dengue_train <- dengue[train_sample,]
dengue_test <- dengue[setdiff(1:nrow(dengue), train_sample),]
```

```r
# fit the model
m1_train <- glm(Dengue ~ Age + WBC + PLT, data = dengue_train,
                family = binomial)

# predict on test data
test_predictions <- predict(m1_train, newdata = dengue_test,
                            type = "response")
pred <- prediction(test_predictions,dengue_test$Dengue)
performance(pred, "auc")@y.values
```

```
[[1]]
[1] 0.8252114
```

Are there any potential issues with this strategy?

Downsides of train/test splits

- We get less data for training
- Performance measure depends on the (random) split

Alternative: cross-validation

Cross validation

- Divide data into k groups (*folds*)
- For each fold $i = 1, \ldots, k$:
  - Train model on the remaining $k - 1$ folds
  - Evaluate on fold i
- Average performance across the k folds

Key take-aways

- Don’t choose a model based solely on training performance
  - Will bias towards more complex models
- Train/test splits and cross-validation give better estimates of model performance