Lecture 19: Cross-Validation for Logistic Regression
STAT 632, Spring 2020

Cross-Validation

- Cross-validation is way to evaluate how well a statistical model performs at making new, or future, predictions of the response variable.
- One simple strategy for cross-validation is to randomly divide the data set into two parts: a **training set** and a **test set**.[1] The model is fit using the training set, and then the fitted model is used to predict the responses on the withheld, test set.
- For logistic regression the performance of the model on the test set can be quantified using classification metrics such as accuracy, sensitivity, and specificity.
- In the literature, you will often see people dividing 70% of the data for training and 30% testing.

[1] Note that the test set is also sometimes called the **validation set**

Example: 2012/16 Election Data

- To illustrate cross-validation for logistic regression we will use the 2012/16 election data set for US counties discussed in HW 7.
- The response variable is `trump_win`, an indicator variable that is 1 if Trump won the county, and 0 otherwise.
- The predictor variable we will consider for this demonstration is `obama_pctvotes`, the percent of votes cast for Obama in each county in 2012.

```
# read in data from URL
> county_votes16 <- readRDS(url("https://ericwfox.github.io/data/county_votes16.rds"))

> head(county_votes16)
  state         county clinton_pctvotes trump_pctvotes obama_pctvotes pct_pop65 pct_black
1    AL Autauga County            23.96          73.44          26.58      13.8      18.7
2    AL Baldwin County            19.57          77.35          21.57      18.7       9.6
3    AL Barbour County            46.66          52.27          51.25      16.5      47.6
4    AL    Bibb County            21.42          76.97          26.22      14.8      22.1
5    AL  Blount County             8.47          89.85          12.35      17.0       1.8
6    AL Bullock County            75.09          24.23          76.31      14.9      70.1
  pct_white pct_hispanic pct_asian highschool bachelors income trump_win
1      77.9          2.7       1.1       85.6      20.9 53.682         1
2      87.1          4.6       0.9       89.1      27.7 50.221         1
3      50.2          4.5       0.5       73.7      13.4 32.911         1
4      76.3          2.1       0.2       77.5      12.1 36.447         1
5      96.0          8.7       0.3       77.0      12.1 44.145         1
6      26.9          7.5       0.3       67.8      12.5 32.033         0

> dim(county_votes16)
[1] 3112   14
```

```
> set.seed(999) # set seed for reproducibility
> n <- nrow(county_votes16); n
[1] 3112
> floor(0.7*n)
[1] 2178

# randomly sample 70% of rows for training set
> train <- sample(1:n, 2178)

# fit model using training data
> glm_train <- glm(trump_win ~ obama_pctvotes, data=county_votes16,
                   subset = train, family = binomial)

> summary(glm_train)
Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)    19.50723    1.18411   16.47   <2e-16 ***
obama_pctvotes -0.36136    0.02252  -16.05   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
# subset data frame for testing observations
> county_votes16_test <- county_votes16[-train, ]

# make predictions for probabilities on test set
> probs_test <- predict(glm_train, newdata = county_votes16_test,
                        type = "response")
```

We can use a 0.5 probability threshold to classify points in the test set. If the predicted probability is greater than 0.5, classify as a Trump win (1). Otherwise, if the predicted probability is less than 0.5, classify as a Trump loss (0).

```
> length(probs_test)
[1] 934

> preds_test <- rep(0, 934)
> preds_test[probs_test > 0.5] <- 1

> head(probs_test)
        3         4         5        10        11        16
0.7285607 0.9999560 0.9999997 0.9999912 0.9999962 0.9999732
> head(preds_test)
[1] 1 1 1 1 1 1
```

Next make the so-called **confusion matrix**. The confusion matrix tabulates the predicted versus actual results on the test set. There are 934 counties in the test set. So we see that the model correctly predicted Trump losing in 129 counties, and correctly predicted Trump winning in 758 counties. There are also some misclassifications: the model predicted Trump losing in 28 counties that he actually won, and the model predicted Trump winning in 19 counties that he actually lost.

```
# make confusion matrix
> tb <- table(prediction = preds_test,
              actual = county_votes16_test$trump_win)

> addmargins(tb)
          actual
prediction   0   1 Sum
       0   129  28 157
       1    19 758 777
       Sum 148 786 934
```

We can use the confusion matrix to compute various classification metrics (accuracy, sensitivity, and specificity) on the test set.

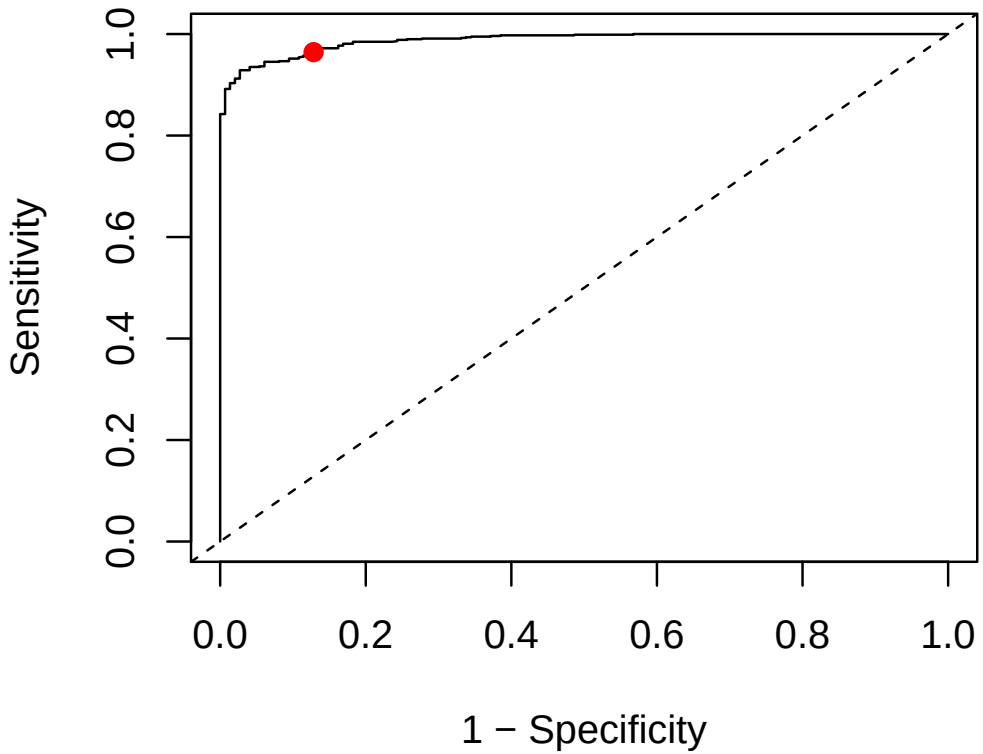
```
> tb <- table(prediction = preds_test,
              actual = county_votes16_test$trump_win)
> addmargins(tb)
          actual
prediction   0   1 Sum
       0   129  28 157
       1    19 758 777
       Sum 148 786 934

# Accuracy (percent correctly classified)
> (129 + 758) / 934
[1] 0.9496788

# Sensitivity (percent of Trump wins (1) correctly classified)
> 758 / 786
[1] 0.9643766

# Specificity (percent of Trump losses (0) correctly classified)
> 129 / 148
[1] 0.8716216
```

ROC Curve

- One issue with the classification metrics we have just considered is that they depended on using a 0.5 probability threshold.
- One popular graphical method to evaluate how well the model does at classifying, regardless of the choice of threshold, is the **receiver operating characteristic (ROC)** curve. The name comes from communication theory.
- The ROC curve is a plot of `sensitivity` versus `1-specificity` for a variety of probability thresholds between 0 and 1.
- In the context of the election data, the `sensitivity`, or true positive rate, is the percent of Trump wins (1) correctly classified. Whereas, `1-specificity`, or false positive rate, is the percent of Trump losses (0) that are misclassified as Trump wins (1).

ROC Curve

ROC curve for the logistic regression model for predicting Trump winning counties in the test set data. Note that the red point corresponds to the 0.5 probability threshold ($\texttt{sensitivity} = 758/786 = 0.964$, and $1 - \texttt{specificity} = 19/148 = 0.128$).



ROC Curve

- The point (0,1) in the top-left corner represents perfect classification. That is, it corresponds to correctly classifying all the response data in the test set.
- In general, the closer the ROC curve is to the top-left corner, or (0,1) point, the better the model performs at classification.
- On the other hand, a model that makes random classifications would have an ROC curve that lies close to the diagonal, or 1-1 line.
- For the election data, the ROC curve is close to the top-left corner, and thus the logistic regression model appears to be performing well at classification.

ROC Curve

I used the `pROC` package to plot the ROC curve (there are other R packages one could use). Here's the code:

```
> library(pROC)
> roc_obj <- roc(county_votes16_test$trump_win, probs_test)
> plot(1 - roc_obj$specificities, roc_obj$sensitivities, type="l",
      xlab = "1 - Specificity", ylab = "Sensitivity")
# plot red point corresponding to 0.5 threshold:
> points(x = 19/148, y = 758/786, col="red", pch=19)
> abline(0, 1, lty=2) # 1-1 line
```

Note that the second argument of `roc()` are the predicted **probabilities** on the test set.

AUC

- Another popular classification metric is the **area under the ROC curve (AUC)**.
- The closer the AUC is to 1 the better the model performs at classification. While a model that performs no better than random guessing would have an AUC close to 0.5.
- One reason the AUC is such a popular metric is because it does not depend on a probability threshold.
- For the election data, the logistic regression model has an AUC that is close to 1. We can compute the AUC using the `pROC` package:

```
> auc(roc_obj)
Area under the curve: 0.9876
```