MY474: Applied Machine Learning for Social Science

Lecture 4: Bias, Variance, Generalization Error, and Cross-Validation

Blake Miller

23 October 2019

Agenda

1. Generalization error
2. Bias-variance tradeoff review
3. Different measures of generalization error
4. Estimating generalization error with validation and cross-validation

Generalization error

Generalization error

- In machine learning our goal is often **prediction**: our goal is to get y as close to $\hat{y}$ as possible.
- But there are many models that will give us many possible $\hat{f}(\cdot)$
- We want to pick the$\hat{f}(\cdot)$that will **generalize** best to new data.
- We can do this by choosing the $\hat{f}(\cdot)$ that minimalizes **generalization error**
- **Generalization error** can be measured in many different ways, depending on the problem.

Two simple measures of generalization error

- **Mean-squared prediction error**: a measure used to evaluate performance of $\hat{f}(x)$ for regression problems
  - Training error: $MSE_{Tr} = Ave_{i \in Tr}[y_i - \hat{f}(x_i)]^2$ (biased when overfitting)
  - Test error: $MSE_{Te} = Ave_{i \in Te}[y_i - \hat{f}(x_i)]^2$ (mitigates bias by using **out of sample** data)
- **Misclassification error rate**: a measure used to evaluate performance of $\hat{C}(x)$ for classification problems
  - Training error: $Err_{Tr} = Ave_{i \in Tr} I[y_i \neq \hat{C}(x_i)]$
  - Test error: $Err_{Te} = Ave_{i \in Te} I[y_i \neq \hat{C}(x_i)]$

Training vs. test error

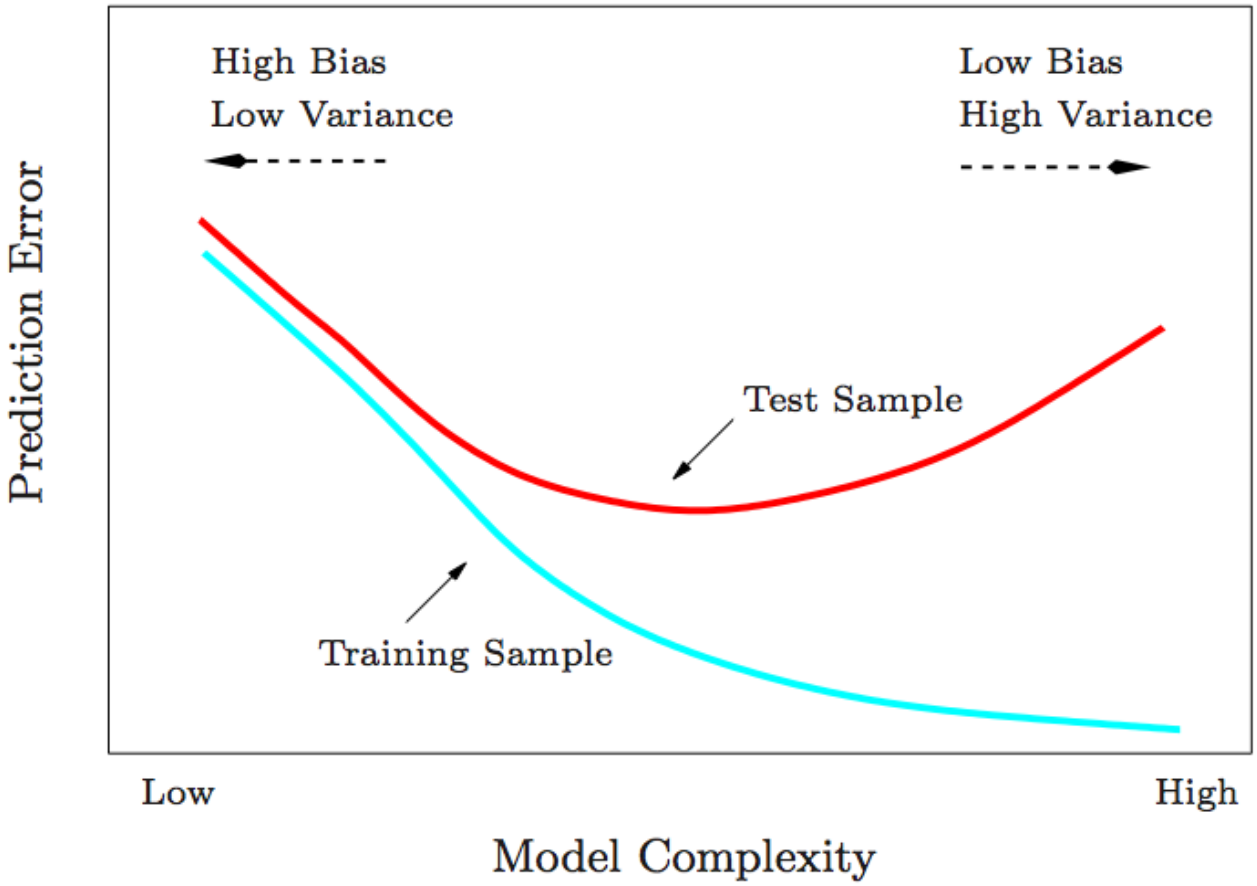
- For supervised learning, data can be divided into a **training** and **test** set
- Recall the distinction between the **test error** and the **training error**:
- The **test error** is the average error that results from using a statistical learning method to predict the response on a new observation, one that was not used in training the method.
- In contrast, the **training error** can be easily calculated by applying the statistical learning method to the observations used in its training.
- But the training error rate often is quite different from the test error rate, and in particular the former can dramatically underestimate the latter.
- Usually the **test set** is smaller. Why?

Training vs. test error

- It is important to estimate the true error rate to accurately assess the classifier
- An over-fitted classifier can achieve perfect training error, but will do poorly on test data
- The easiest way to assess true error is to hold out part of the data to be the test set; but that means reducing sample size n and therefore not training the classifier as well as possible
- Question: how can we assess the true error rate without wasting data?

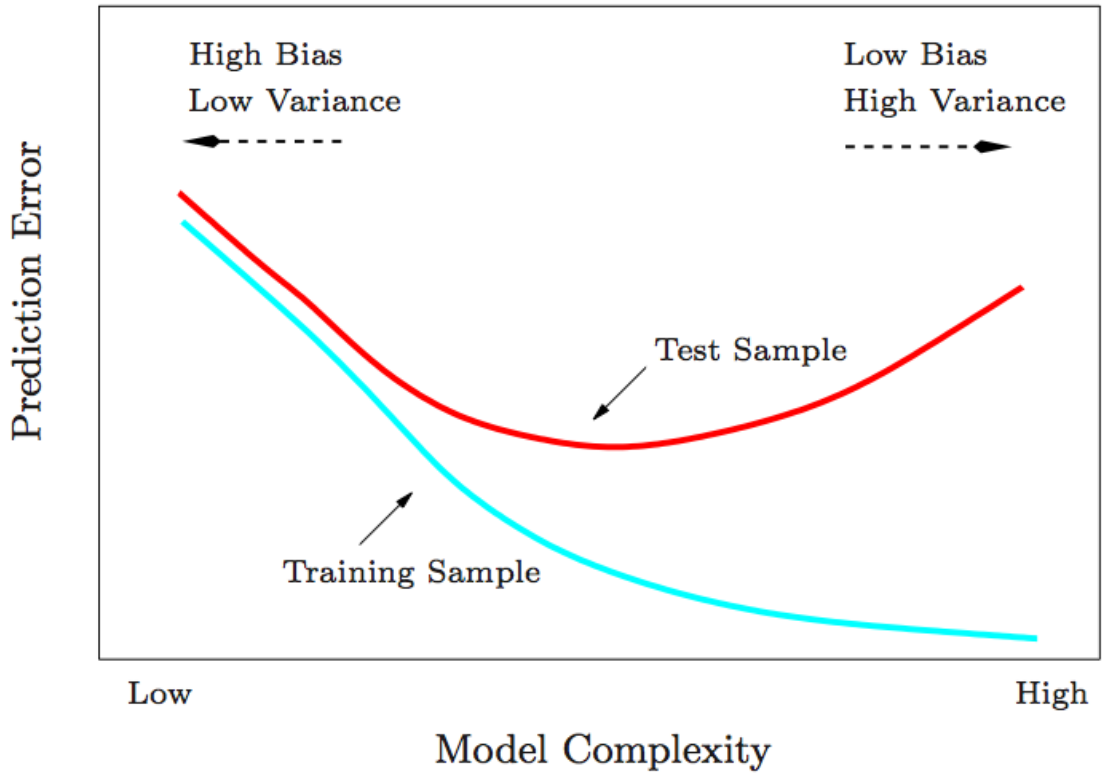
Bias-variance tradeoff review

The bias-variance tradeoff



- As flexibility of $\hat{f}$ increases, its **variance** increases and its **bias** decreases.
- Choosing the flexibility based on average **test error** amounts to a **bias-variance trade-off**.

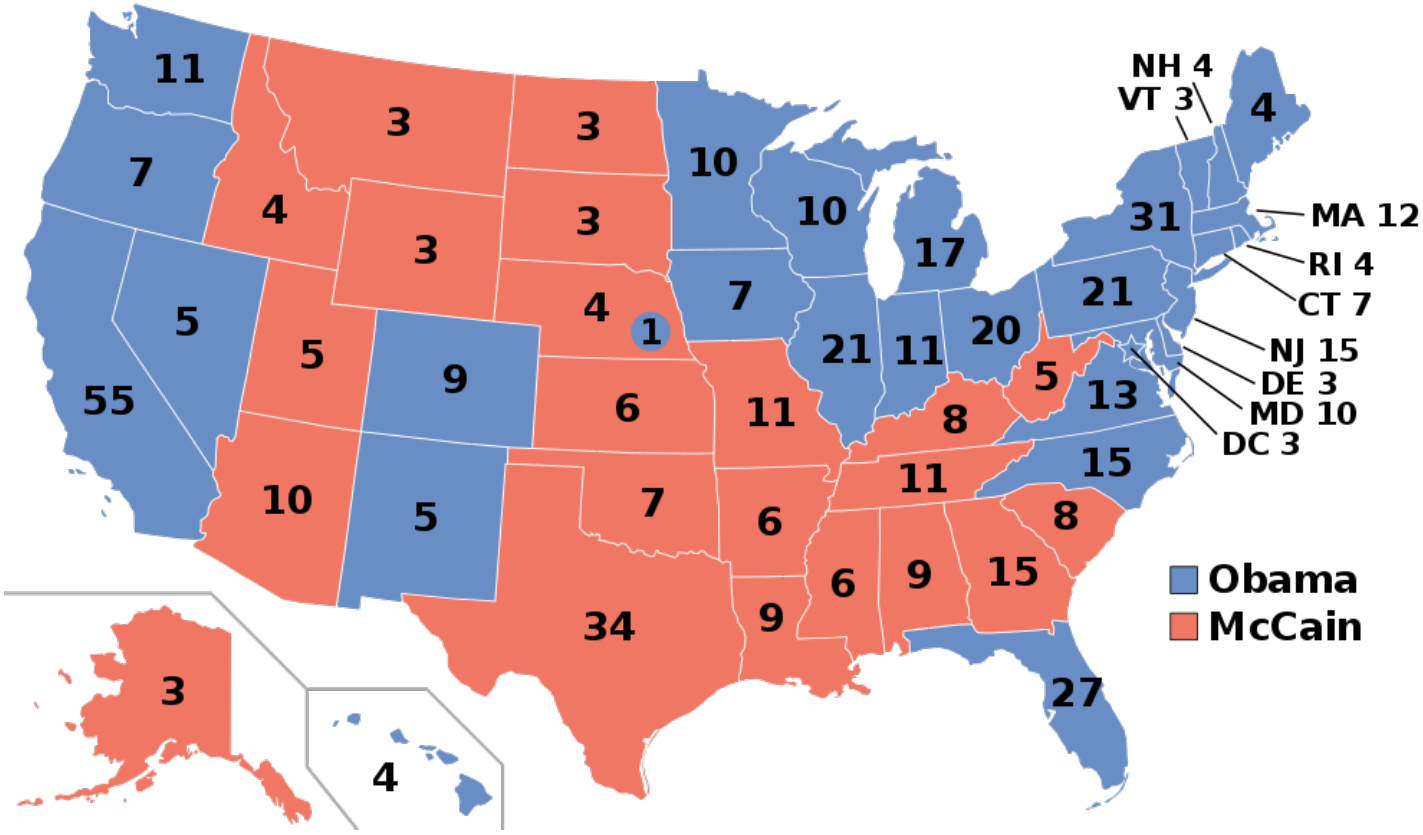
The bias-variance tradeoff



- Variance: the degree to which there is difference in fits across datasets
- Bias: The inability for a machine learning method (like linear regression) to capture the true relationship.
- The goal is to find the sweet spot between a simple model and a complex model such that we jointly minimize bias and variance.

Different measures of generalization error

Motivating example: 2008 US Presidential Election



President Obama wins with 365 electoral votes to McCain's 173.

- In this example we will go over a prediction task that can be formulated as a **classification** and a **regression** problem
- The goal of this exercise is to make various **performance measures** or **measures of generalization error** clear.

Motivating example: 2008 US Presidential Election

- A number of pundits accurately predicted the 2008 election outcome.
- Electoral College system makes predicting election outcomes challenging.
  - Must win an absolute majority of electoral votes to be elected president
  - Each of the 538 electors casts a single electoral vote.
  - Most states have "winner-take- all" system.
  - A winning presidential candidate must obtain at least 270 electoral votes.
- **Goal:** predict election outcome using public opinion polls conducted within each state.

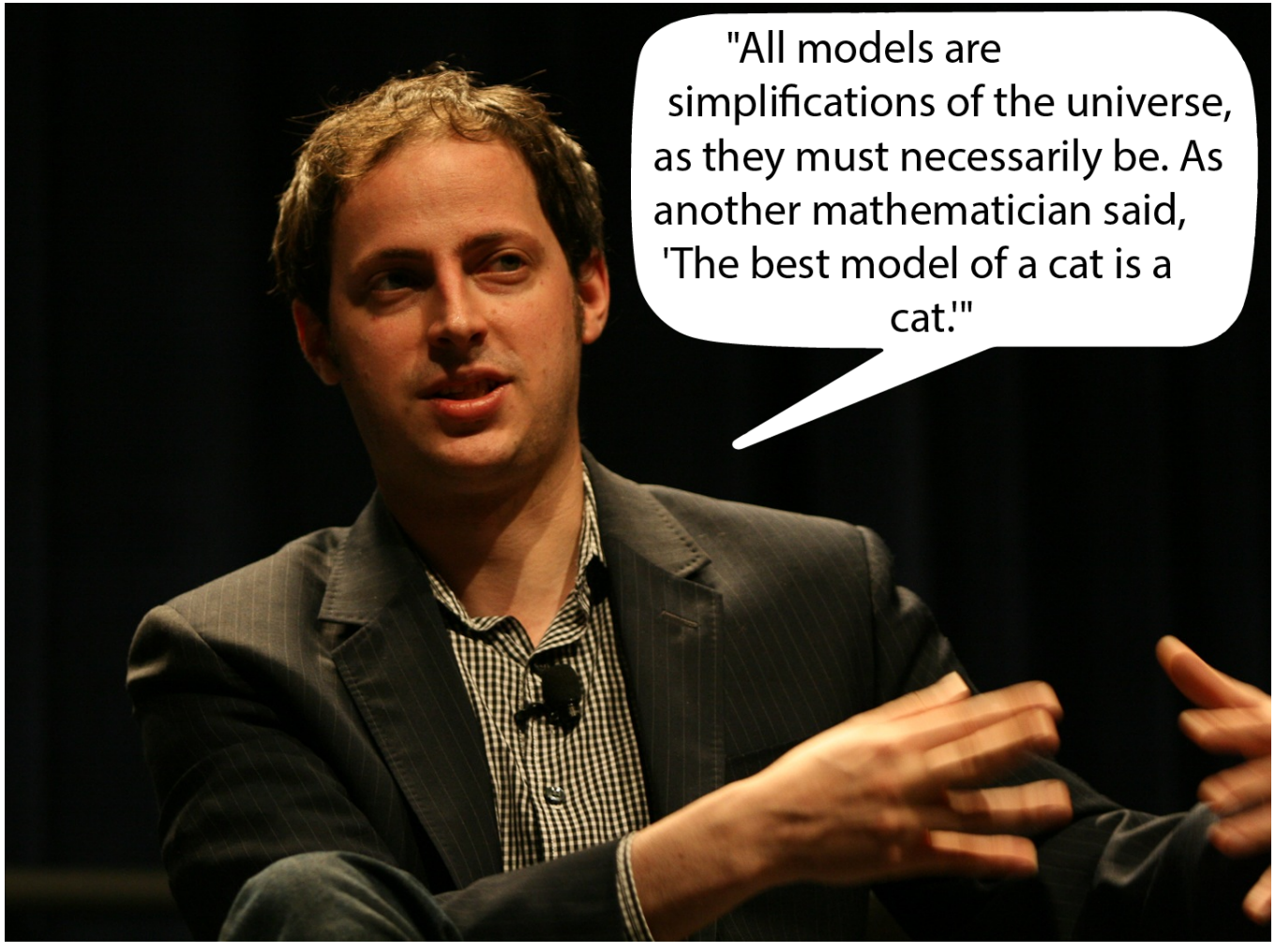
2008 US Presidential Election outcomes data (actual/y)

Variable	**Description**
`state`	abbreviated name of the state
`state.name`	unabbreviated name of the state
`Obama`	Obama's vote share (percentage)
`McCain`	McCain's vote share (percentage)
`EV`	number of Electoral College votes for the state
`margin`	Obama's margin

2008 US Presidential Election polling data (predicted/$\hat{y}$)

Variable	**Description**
`state`	abbreviated name of the state
`Obama`	predicted support for Obama (percentage)
`McCain`	predicted support for McCain (percentage)
`Pollster`	name of the organization conducting the poll
`middate`	middate of the period when the poll was conducted
`margin`	Obama's margin
`days`	# days ahead of election the poll was conducted

Our simple model for predicting the election outcome



The model: Mean margin of all the polls on the latest day of polling in each state.

Predicting vote margin from polls

Below we use the latest polls from each state to predict Obama's vote margin:

```
pres <- read.csv("pres08.csv")
polls <- read.csv("polls08.csv")

s_names <- unique(polls$state)
actual <- pres$margin
names(actual) <- as.character(s_names)
poll_pred <- rep(NA, 51)
names(poll_pred) <- as.character(s_names)

for (i in 1:51){
    s <- polls[polls$state == s_names[i], ]
    latest <- subset(s, days == min(days))
    poll_pred[i] <- mean(latest$margin)
}
```

Mean squared error of polls

```
head(poll_pred)
```

```
##    AL    AK    AZ    AR    CA    CO
## -25.0 -19.0  -2.5  -7.0  24.0   7.0
```

```
head(actual)
```

```
##  AL  AK  AZ  AR  CA  CO
## -21 -21  -9 -20  24   9
```

```
sq_err <- (poll_pred - actual)^2
head(sq_err)
```

```
##     AL     AK     AZ     AR     CA     CO
##  16.00   4.00  42.25 169.00   0.00   4.00
```

```
mean(sq_err)
```

```
## [1] 34.91558
```

Misclassification error rate of state predictions

```
pred <- ifelse(poll_pred > 0, "Obama", "McCain")
actual <- ifelse(pres$margin > 0, "Obama", "McCain")

mean(pred != actual)
```

```
## [1] 0.05882353
```

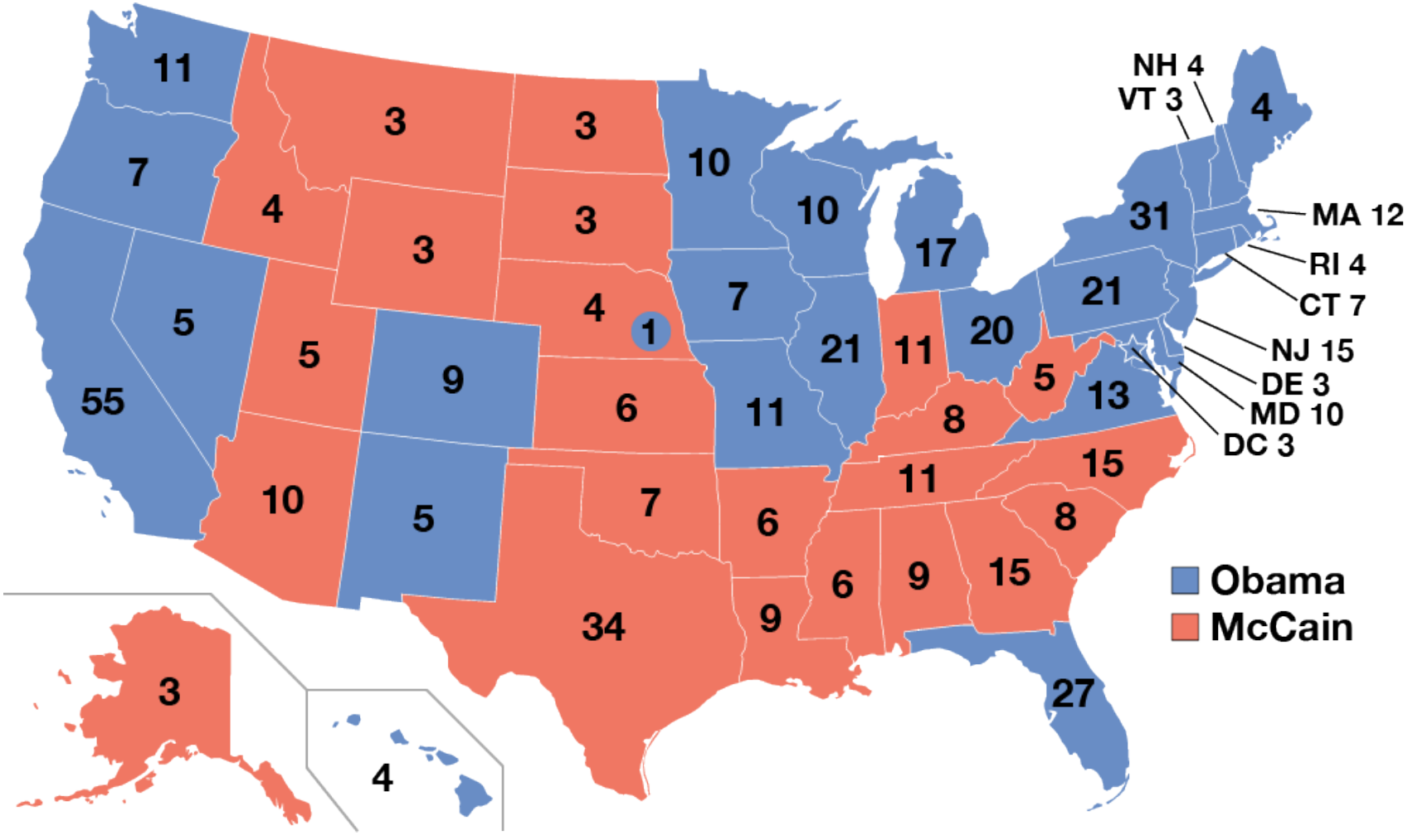
```
poll_pred[pred != actual]
```

```
## IN MO NC
## -5  1 -1
```

```
actual[pred != actual]
```

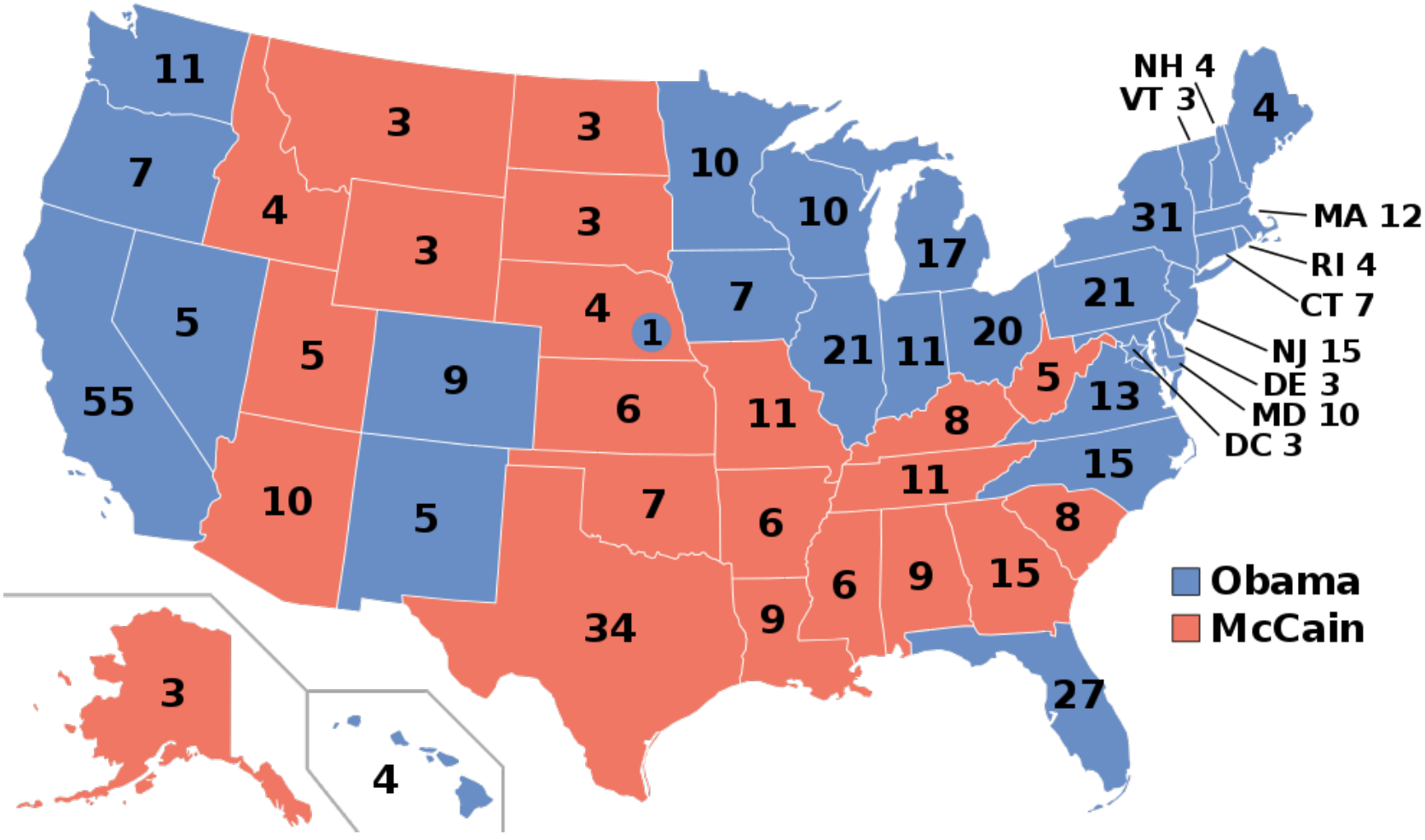
```
## [1] "Obama"  "McCain" "Obama"
```

Our model's state predictions



President Obama wins with 349 electoral votes to McCain's 189.

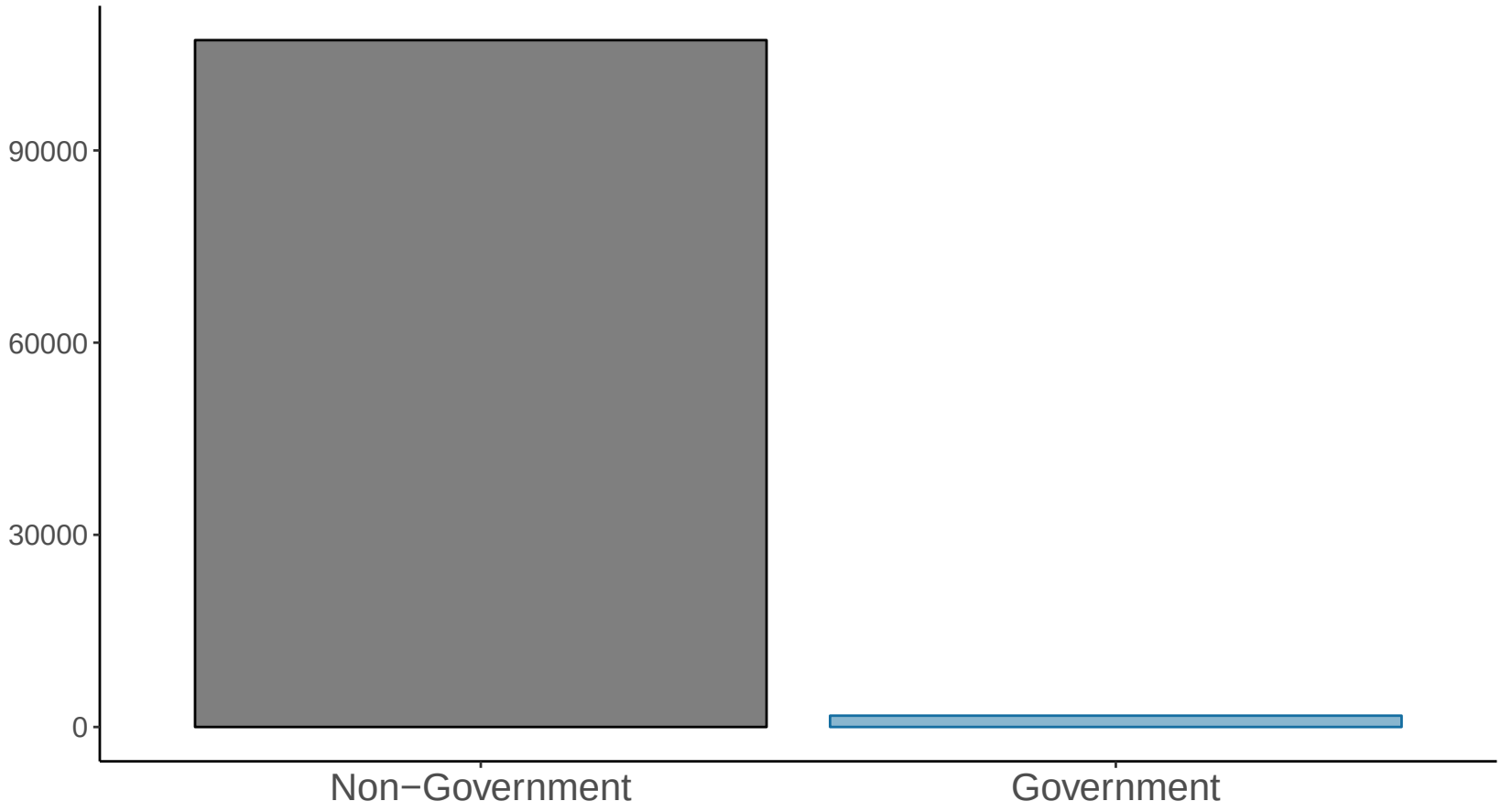
Actual election outcome



President Obama wins with 365 electoral votes to McCain's 173.

What if our classes were more imbalanced?

What if our classes were more imbalanced?



Confusion matrix of state predictions

```
table(pred, actual)
```

```
##          actual
## pred      Obama McCain
##   Obama      27       1
##   McCain      2      21
```

- Rows correspond to what the model predicted, i.e. $\hat{y}$
- Columns correspond to the known truth, i.e. y

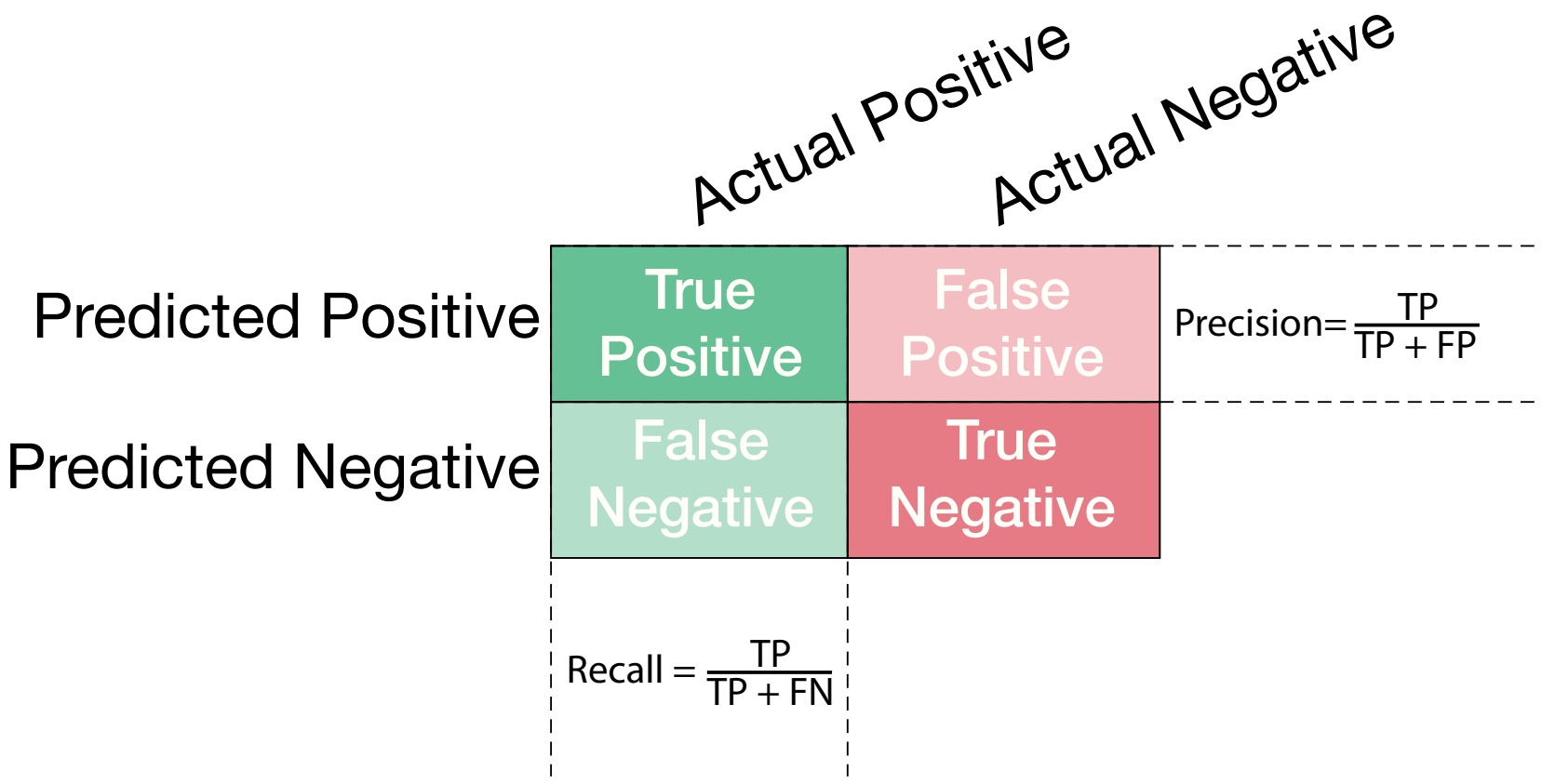
Confusion matrix of state predictions

```
table(pred, actual)
```

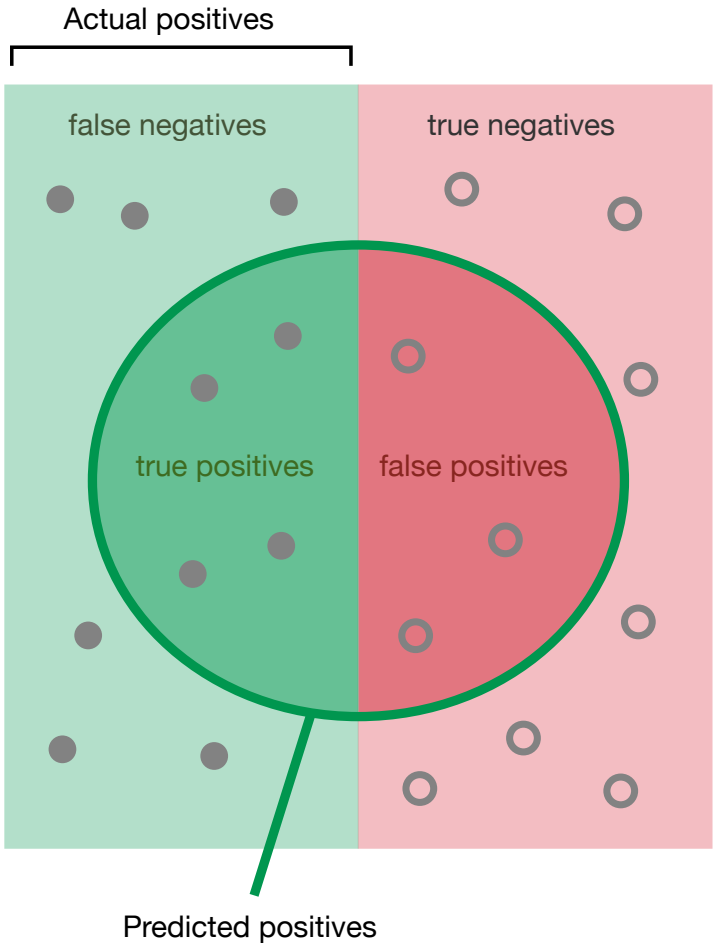
```
##          actual
## pred      Obama McCain
##   Obama      27       1
##   McCain      2      21
```

- Top left (**true positives**): predicted Obama states that were actually Obama states.
- Bottom right (**true negatives**): predicted McCain states that were actually McCain states.
- Top right (**false positives/Type I error**): predicted Obama states that were actually McCain states.
- Bottom left (**false negatives/Type II error**): predicted McCain states that were actually Obama states.

Confusion matrix



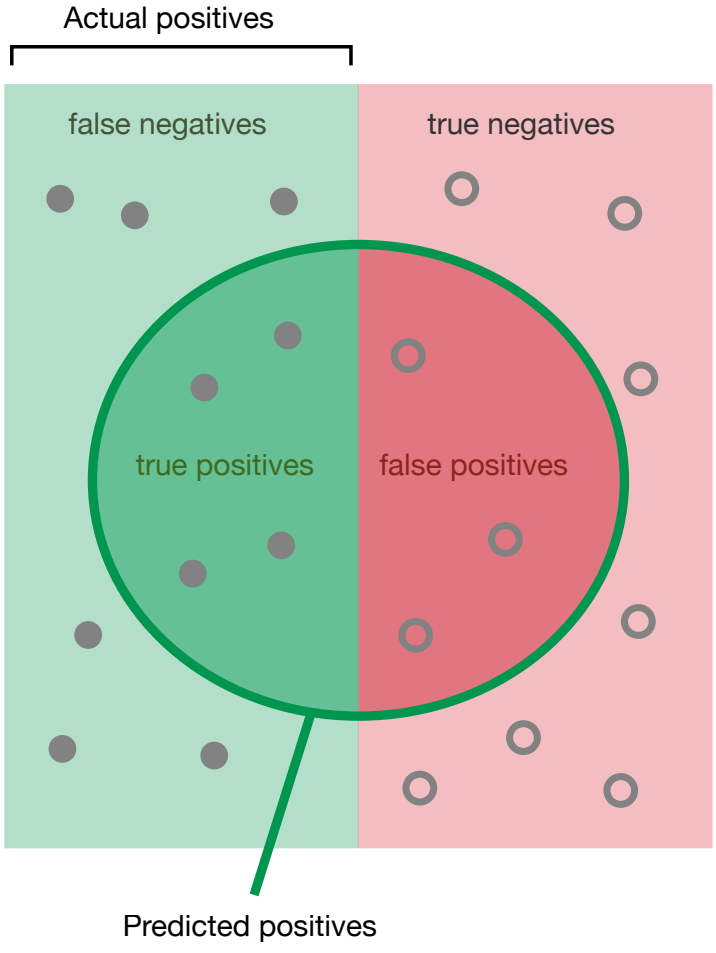
Performance measures: Precision



What proportion of predicted positives are in fact positive?

Precision =  = 4/7 = .57

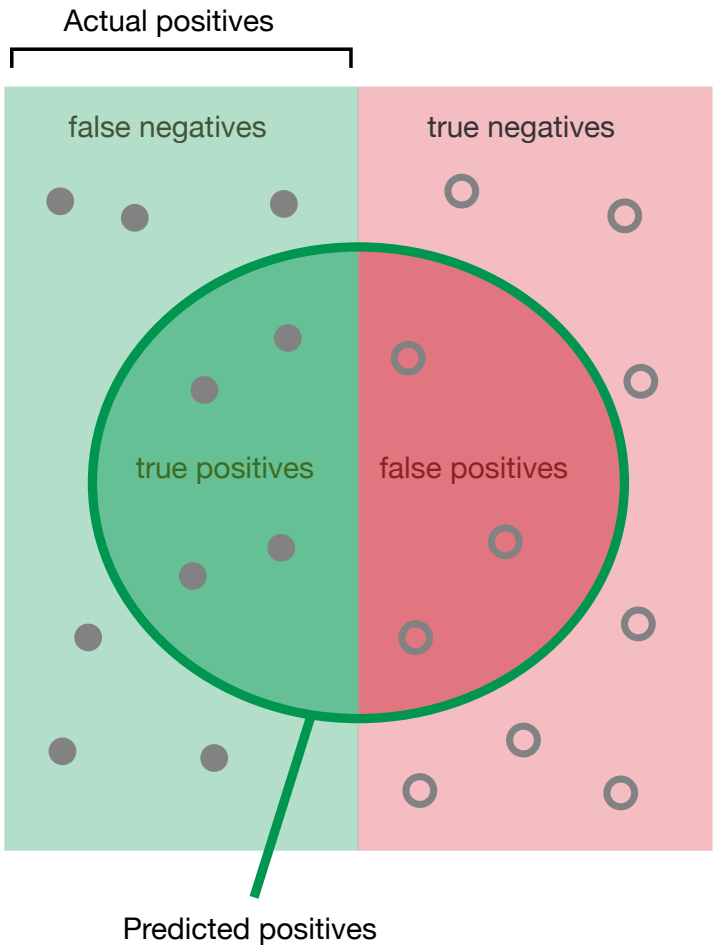
Performance measures: Recall/Sensitivity/True Positive Rate



What proportion of the actual positives are predicted to be positive?

$$\text{Recall} = \frac{\text{true positives}}{\text{actual positives}} = 4/10 = .4$$

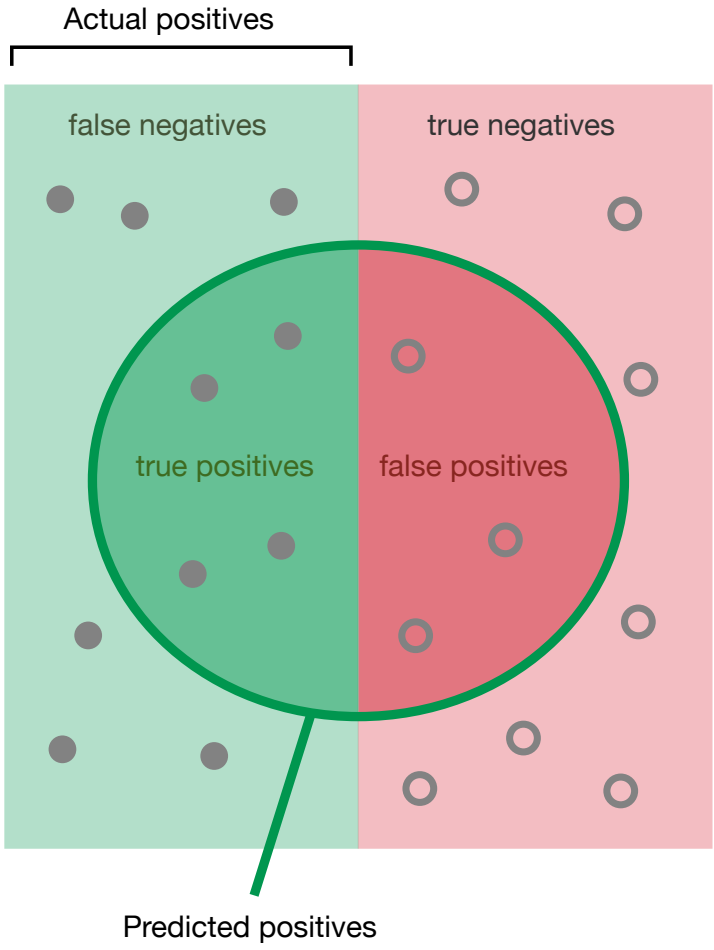
Performance measures: False Positive Rate



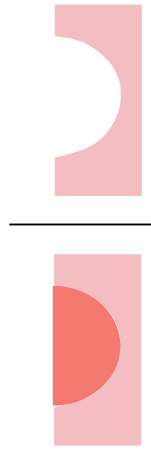
What proportion of actual negatives are predicted to be positive?

False Positive Rate=  /  = 3/10 = .3

Performance measures: Specificity



How often are actual negatives predicted to be negative?

Specificity =  = 7/10 = .7

Performance measures: F_1, F_k

- F_1 is a performance measure that balances **precision** and **recall**:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

- F1 is the **harmonic mean** of the two measures.
- The "1" in F_1 represents a parameter that can be adjusted to weight the relative importance of precision and recall:

$$F_k = (1 + k^2) \cdot \frac{\text{precision} \cdot \text{recall}}{k^2 \cdot \text{precision} + \text{recall}}$$

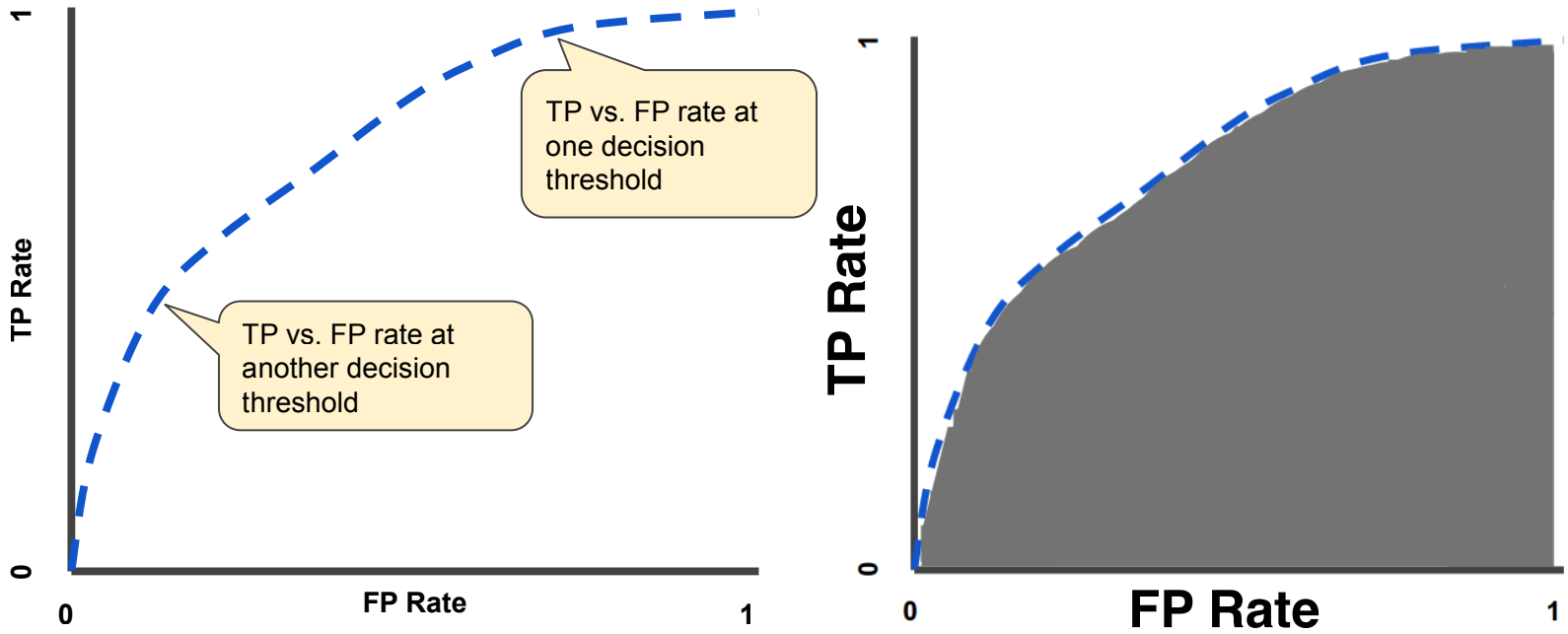
- A bigger k weights recall higher than precision; a smaller k weights precision higher than recall.

Area under the receiver operator curve (AUC)

Sailors operating a 285 gunnery radar. Photo by William H. Pugsley. National Archives of Canada, PA-139273.

- ROC was first used in World War II to aid in the detection of enemy submarines
- A true positive (enemy submarine) needed to be distinguished from a false positive (whales, fish, etc.)

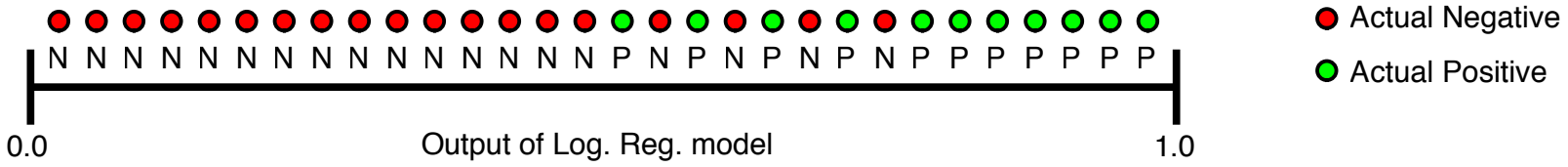
Area under the receiver operator curve (AUC)



- Left: True positive rate vs. false positive rate at different classification thresholds.
- Right: Area under the ROC Curve.
- The curve is then defined by the false positive rate and false negative rate if that predicted outcome were to represent the **classification threshold**.

[Source: developers.google.com](https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc)

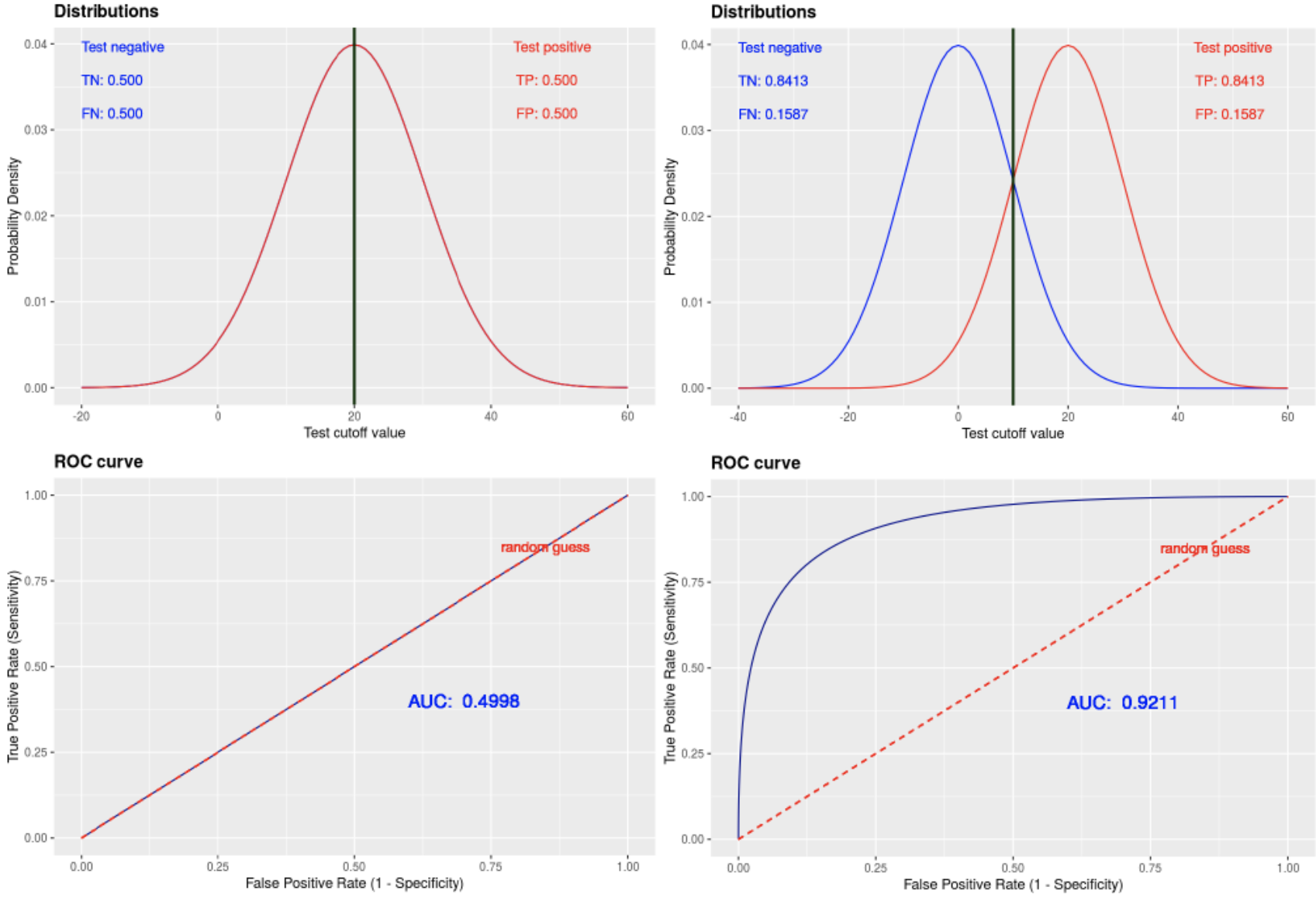
Area under the receiver operator curve (AUC)



Predictions from a logistic regression model ordered by predicted probability. Source: developers.google.com

- ROC curve ranks all observations in the order of their predicted outcome (for example predicted probabilities of a logistic regression model).
- AUC can be interpreted as the probability that a randomly drawn positive (green) observation is positioned to the right of a randomly drawn negative (red) observation.

Area under the receiver operator curve (AUC)



Click here for an interactive demo.

Estimating generalization error

Approaches to estimating generalization error

- Best solution: a large designated test set. Often not available
- An alternative: estimate the test error by holding out a subset of the training observations from the fitting process, and then applying the statistical learning method to those held out observations.

Some terms

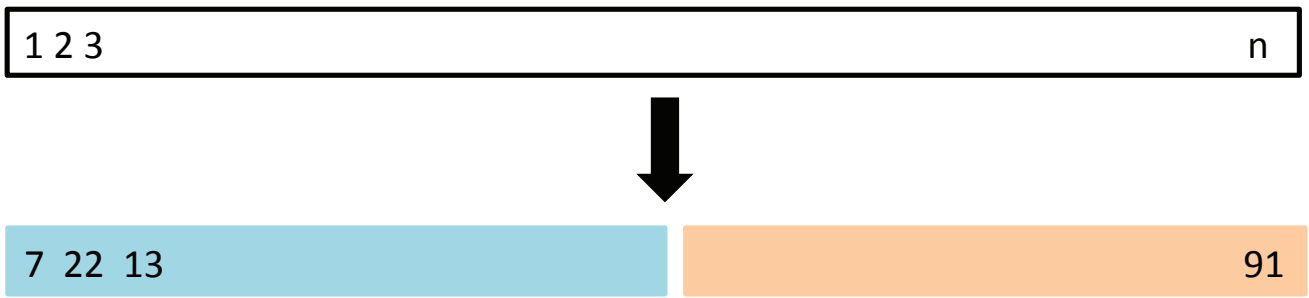
- **Training set**: A subsample used to fit a model (e.g. training set can be used to estimate model parameters).
- **Validation set**: A subsample used to tune **hyperparameters** (i.e. k in KNN).
- **Test set**: A subsample used only to measure the generalization error of a fully specified model.

Validation-set approach

- Here we randomly divide the available set of samples into two parts: a training set and a **validation** or **hold-out set**.
- The model is fit on the training set, and the fitted model is used to predict the responses for the observations in the validation set.
- The resulting validation-set error provides an estimate of the test error. This is typically assessed using MSE in the case of a quantitative response and misclassification rate in the case of a qualitative (discrete) response.

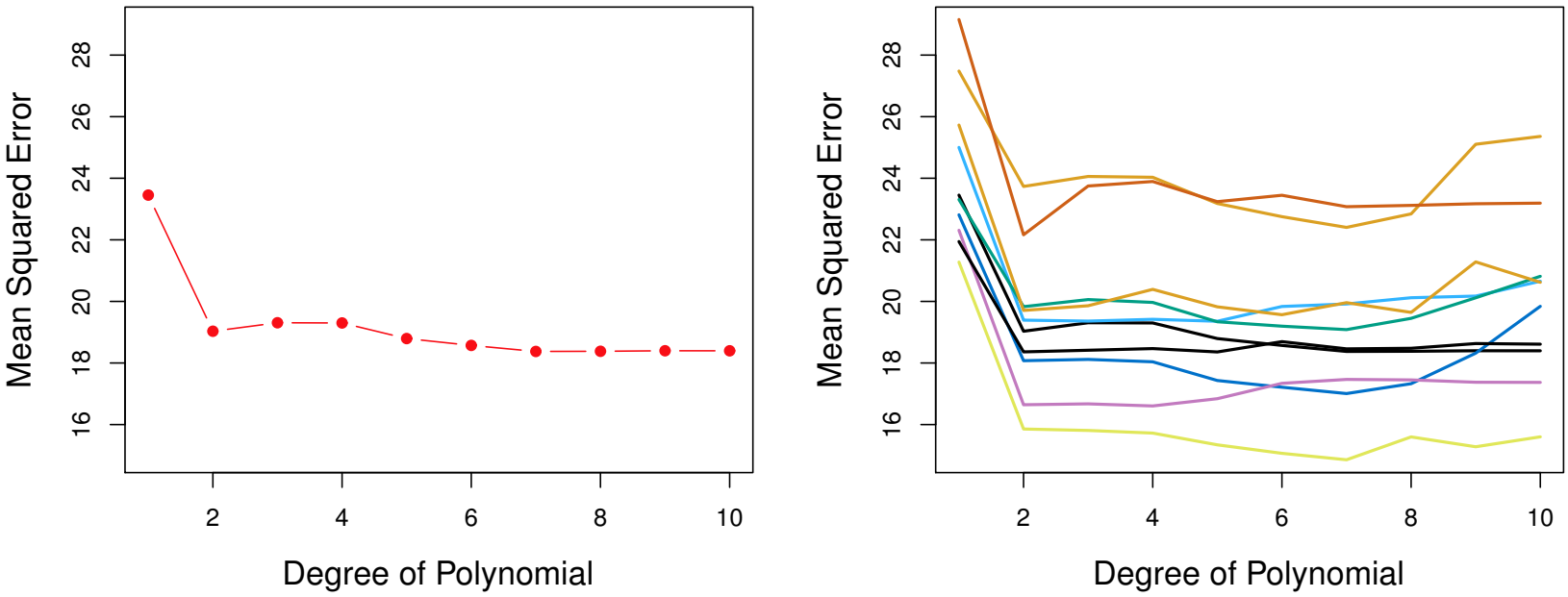
The validation process

A random splitting into two halves: left part is training set, right part is validation set



Example: automobile data

- Want to compare linear vs higher-order polynomial terms in a linear regression
- We randomly split the 392 observations into two sets, a training set containing 196 of the data points, and a validation set containing the remaining 196 observations.



Left panel shows single split; right panel shows multiple splits

Drawbacks of validation set approach

- the validation estimate of the test error can be highly variable, depending on precisely which observations are included in the training set and which observations are included in the validation set.
- In the validation approach, only a subset of the observations—those that are included in the training set rather than in the validation set — are used to fit the model.
- This suggests that the validation set error may tend to overestimate the test error for the model fit on the entire data set. Why?

K-fold cross-validation

- Estimates of generalization error from one train / validation split can be noisy, so shuffle data and average over K distinct validation partitions instead
- K-fold cross-validation is a widely used approach for estimating test error.
- Estimates can be used to select best model, and to give an idea of the test error of the final chosen model.

K-fold cross-validation

How it works:

1. Randomly divide the data into K parts of equal length
2. Hold out one of the parts to be the "test" data.
3. Use all remaining data to train the model.
4. Predict on the held-out part and compute the error rate.
5. Repeat steps 2-4 for each of the K parts.
6. Average all "test" errors to obtain the cross-validation error.

K-fold cross-validation in detail

Divide data into K roughly equal-sized parts ($K = 5$ here)

The details

- Let the K parts be $C_1, C_2, \ldots C_K$, where C_k denotes the indices of the observations in part k. There are n_k observations in part k: if N is a multiple of K, then $n_k = n/K$.
- Compute

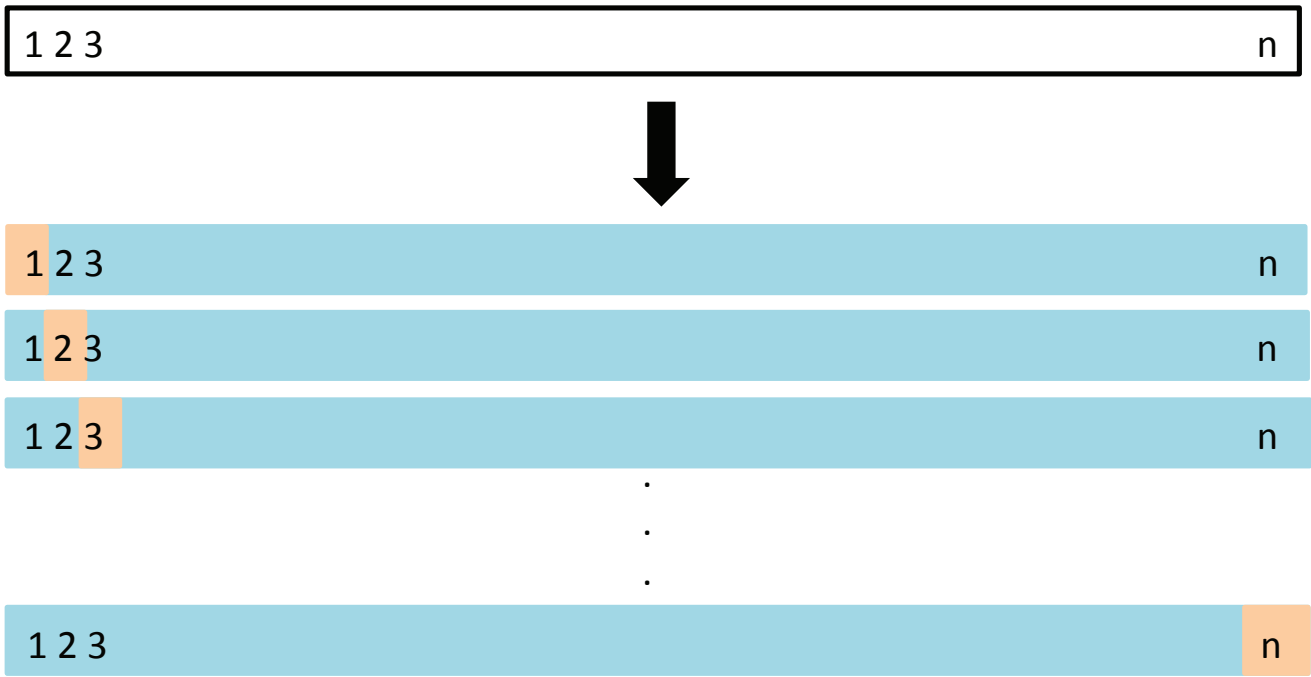
$$CV_{(K)} = \sum_{k=1}^{K} \frac{n_k}{n} MSE_k$$

where $MSE_k = \sum_{i \in C_k} (y_i - \hat{y}_i)^2 / n_k$, and $\hat{y}_i$ is the fit for observation i, obtained from the data with part k removed. - Setting $K = n$ yields n-fold or leave-one out cross-validation (LOOCV).

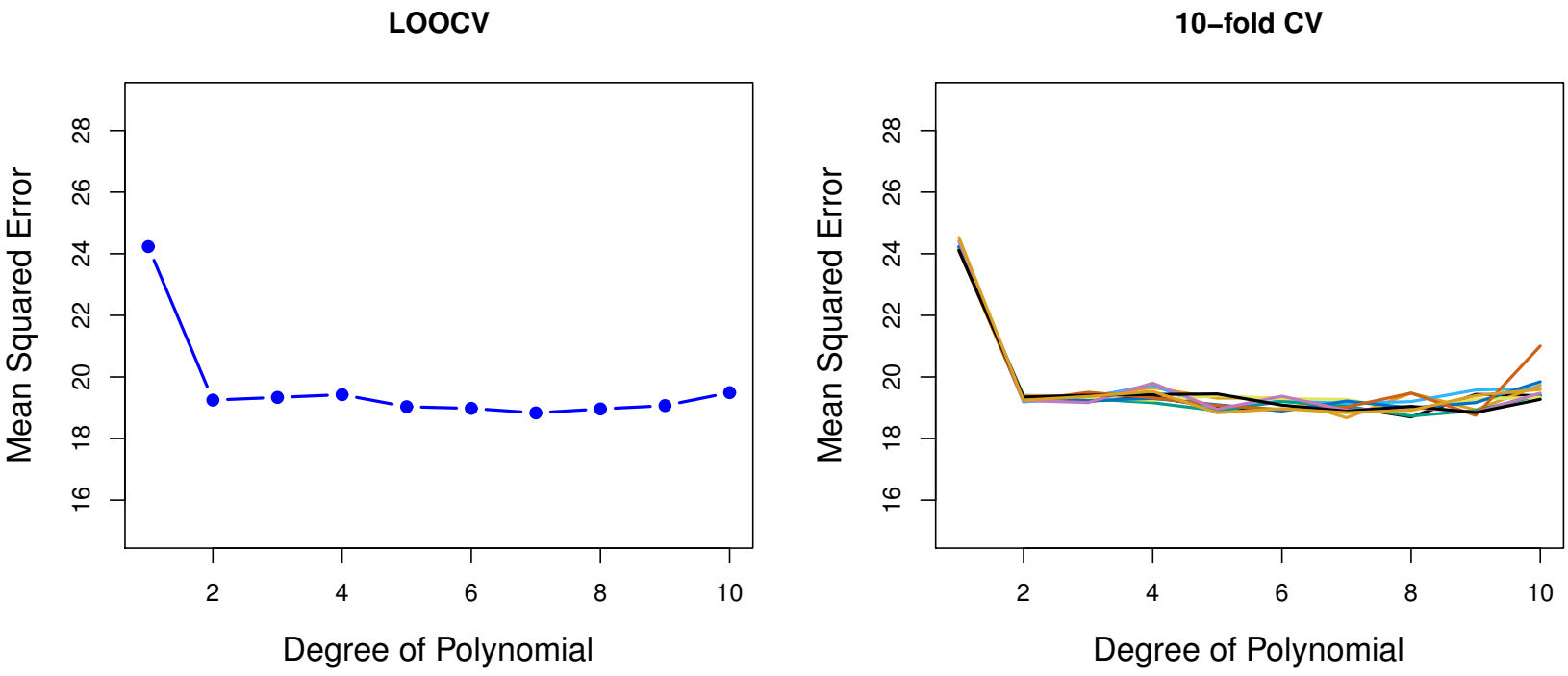
More about cross-validation

- K can be anything; popular values are 5, 10, n.
- The cross-validated error rate tends to be closer to the true error rate than to the apparent error rate.
- The computational cost can become a concern, particularly if there are many tuning parameters

Leave one out cross-validation



Auto data revisited



On the right, each line represents a different partition of the data.

Other issues with cross-validation

- Since each training set is only $(K-1)/K$ as big as the original training set, the estimates of prediction error will typically be biased upward. Why?
- This bias is minimized when $K = n(LOOCV)$, but this estimate has high variance, as noted earlier.
- $K = 5$ or 10 provides a good compromise for this bias-variance tradeoff.
- Like our choice of models, our CV decision involves a bias-variance tradeoff

Cross-validation for classification problems

- We divide the data into K roughly equal-sized parts $C_1, C_2, ...C_K$. C_k denotes the indices of the observations in part k. There are nk observations in part k: if n is a multiple of K, then $n_k = n/K$.
- Compute

$$CV_K = \sum_{k=1}^{K} \frac{n_k}{n} Err_k$$

  where $Err_k = \sum_{i \in C_k} I(y_i \neq \hat{y}_i)/n_k$.
- n_k accommodates folds of different sizes. If they are exactly the same size, the weight of the error will be $1/K$

Choosing hyperparameters using cross-validation

- Cross validation should be used to select a model when there are many **hyperparameters** to choose from or feature selection or feature preprocessing decisions to be made
- A **hyperparameter** is a value that can be tuned to improve the fit of a model. An example is k in KNN
- Usually, if you have a more complex model, you will need to do a **hyperparameter search**