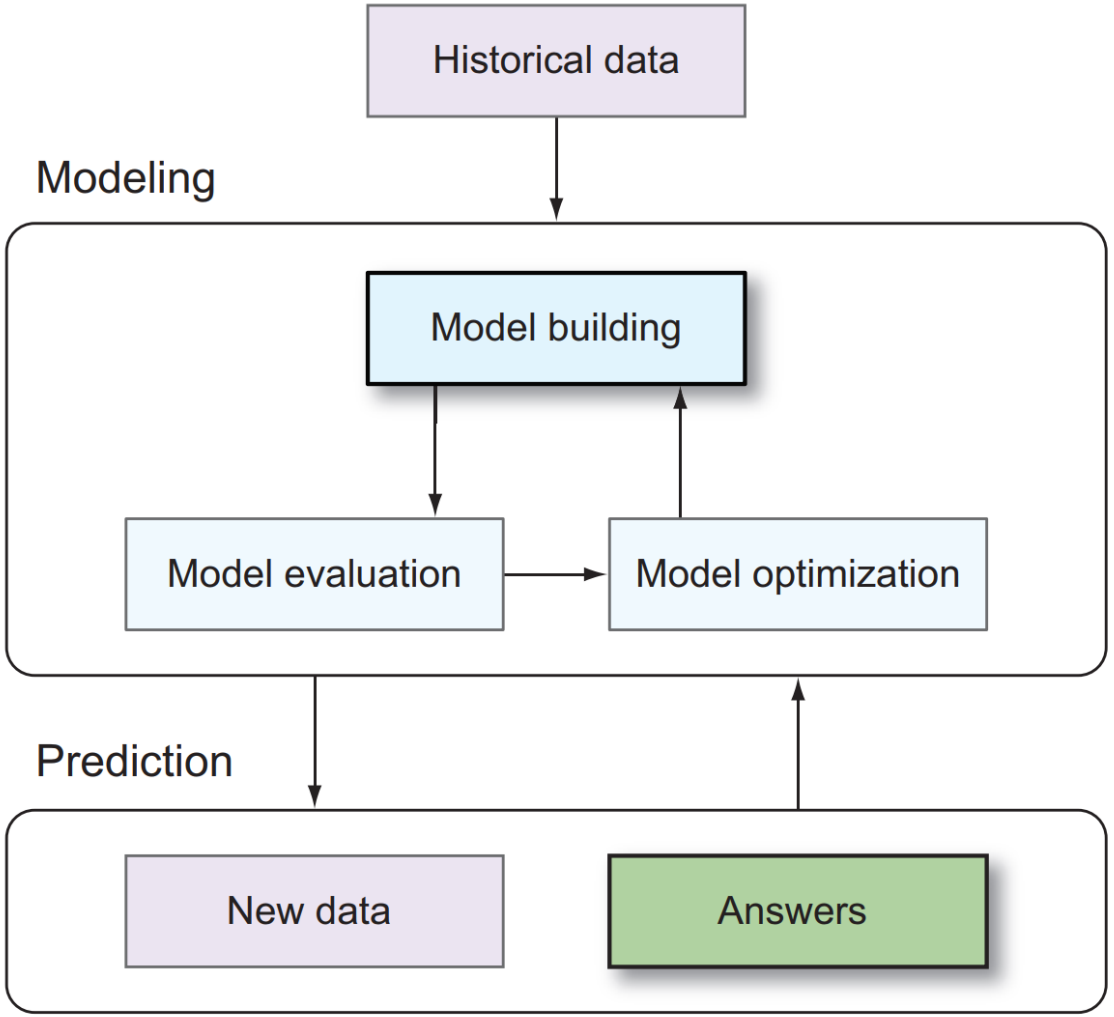
CSE 445
Lecture 11

Model evaluation & selection

Pattern modeling

- Different modeling approaches share a common workflow
  - Use of historical data to build and optimize a model
  - The model is used to make predictions based on new data

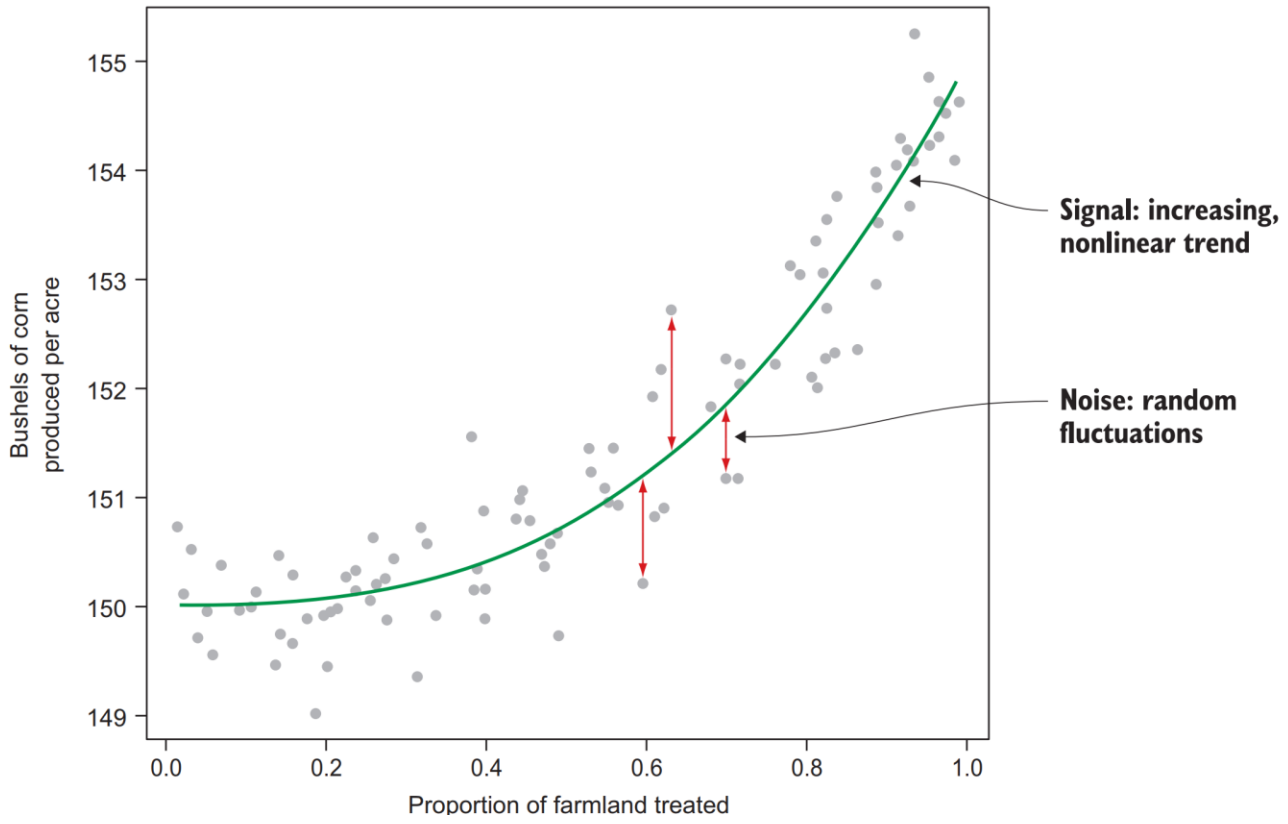


Why model evaluation?

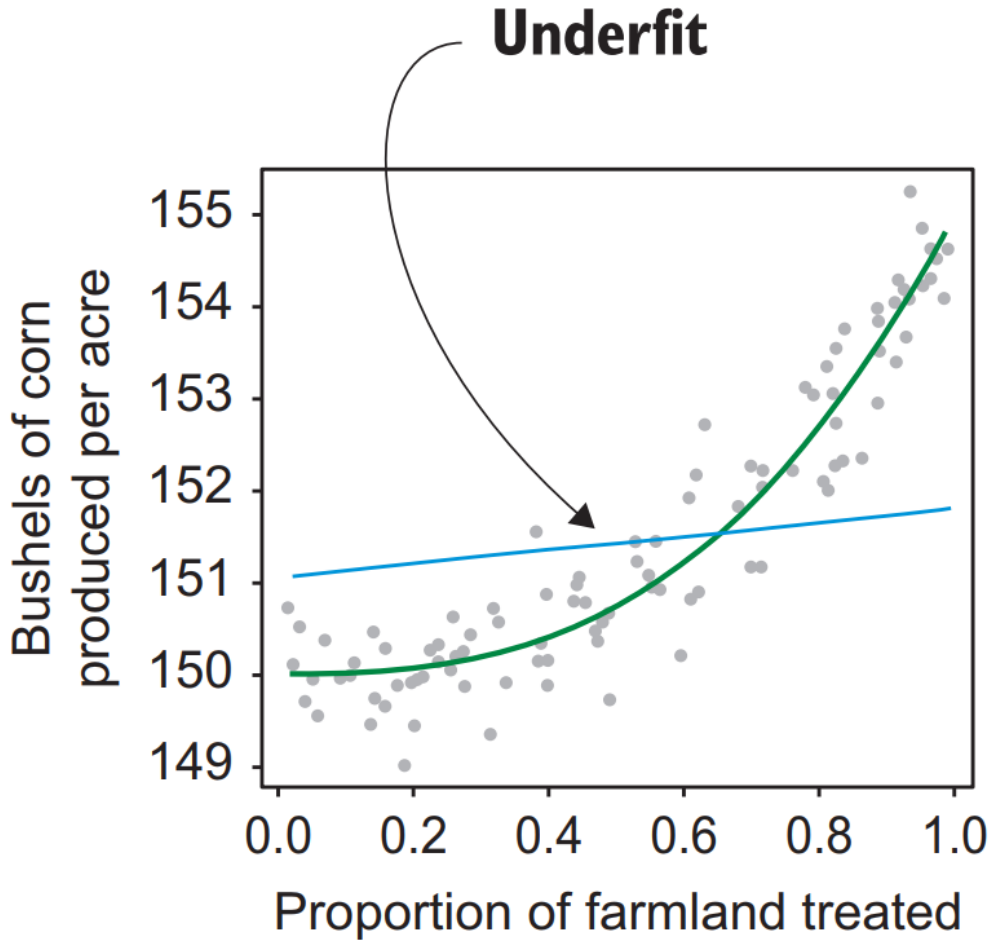
- The primary goal of supervised machine learning is accurate prediction
- We want our model, which has been built from training data, to generalize well to new data
- That way, when we deploy the model in production, we can be assured that the predictions generated are of high quality
- Therefore, when we evaluate the performance of a model, we want to determine *how well that model will perform on new data*
- This seemingly simple task is wrought with complications and pitfalls that can befuddle even the most experienced ML users
- Model selection discusses bunch of methods for feature extraction and selection
  - However we need to be familiar with dimensionality reduction

Problem 1: Overfitting

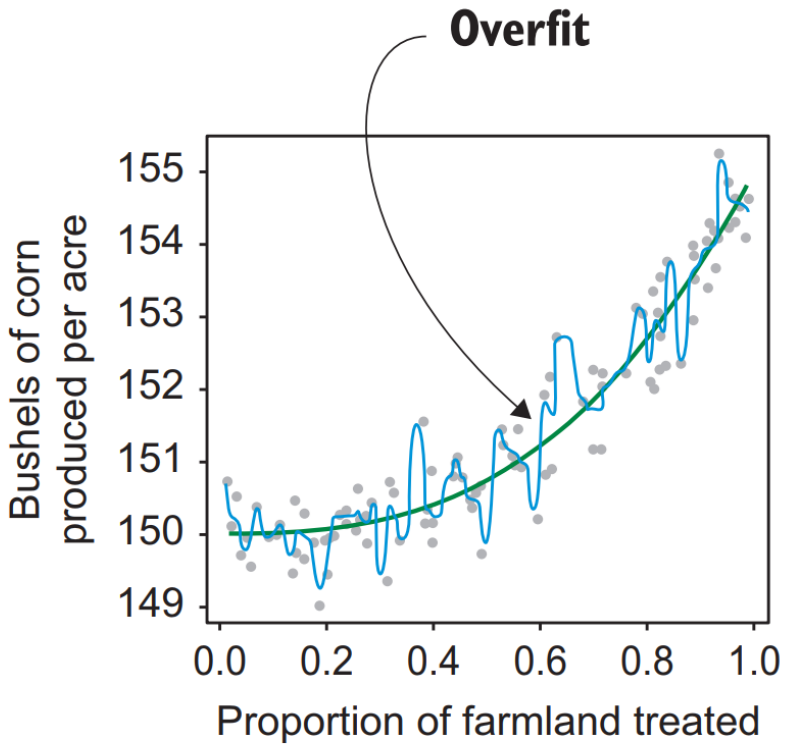
- We want to predict the production of bushels of corn per acre on a farm as a function of the proportion of that farm's planting area that was treated with a new pesticide
- It is clear that an increasing, nonlinear relationship exists, and that the data also has random fluctuations

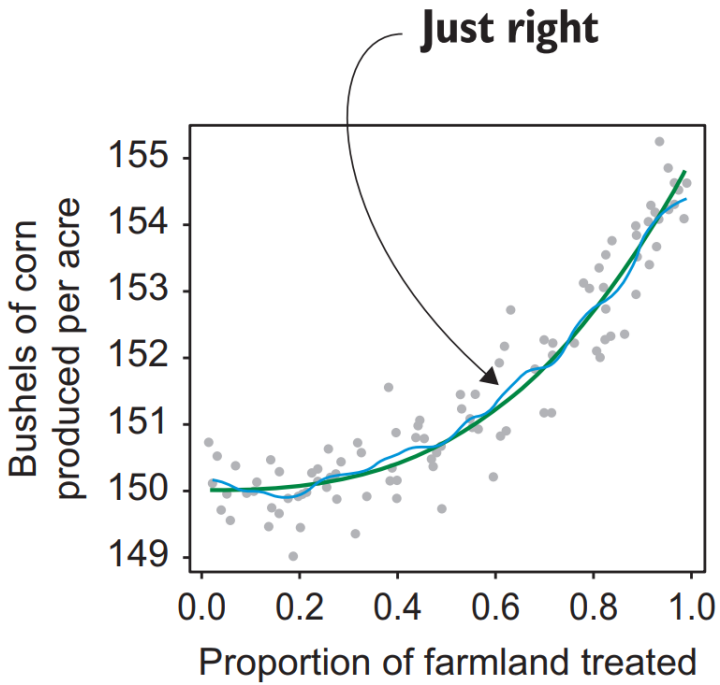


- We will use kernel smoothing technique
- For large values of the bandwidth, almost all of the training data is averaged together to predict the target, at each value of the input parameter
- This causes the model to be flat and to underfit the obvious trend in the training data



- for small values of the bandwidth, only one or two training instances are used to determine the model output
- Therefore, the model effectively traces every bump and wiggle in the data
- This susceptibility to model the intrinsic noise in the data instead of the true signal is called *overfitting*

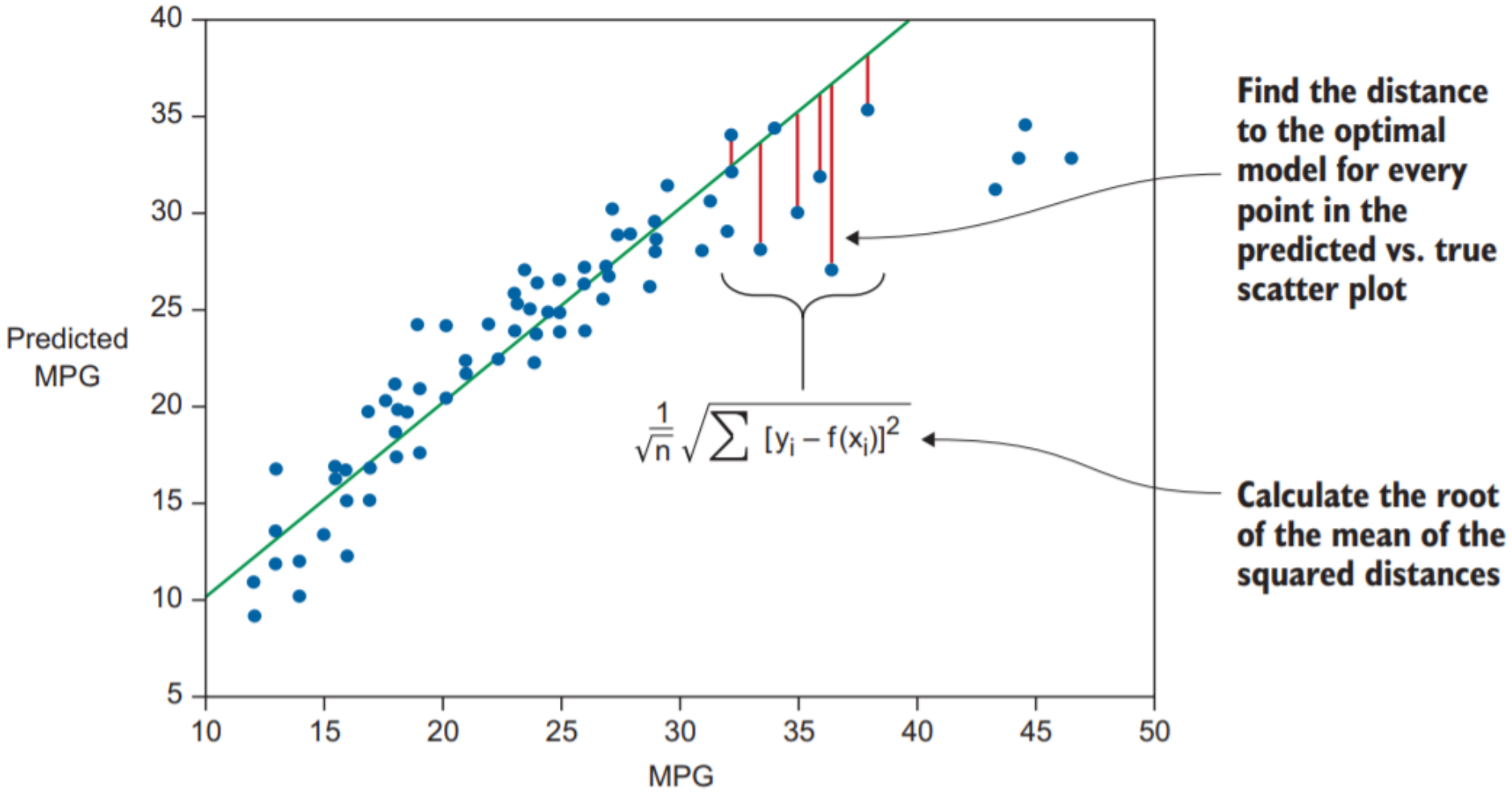




So what do we do now?

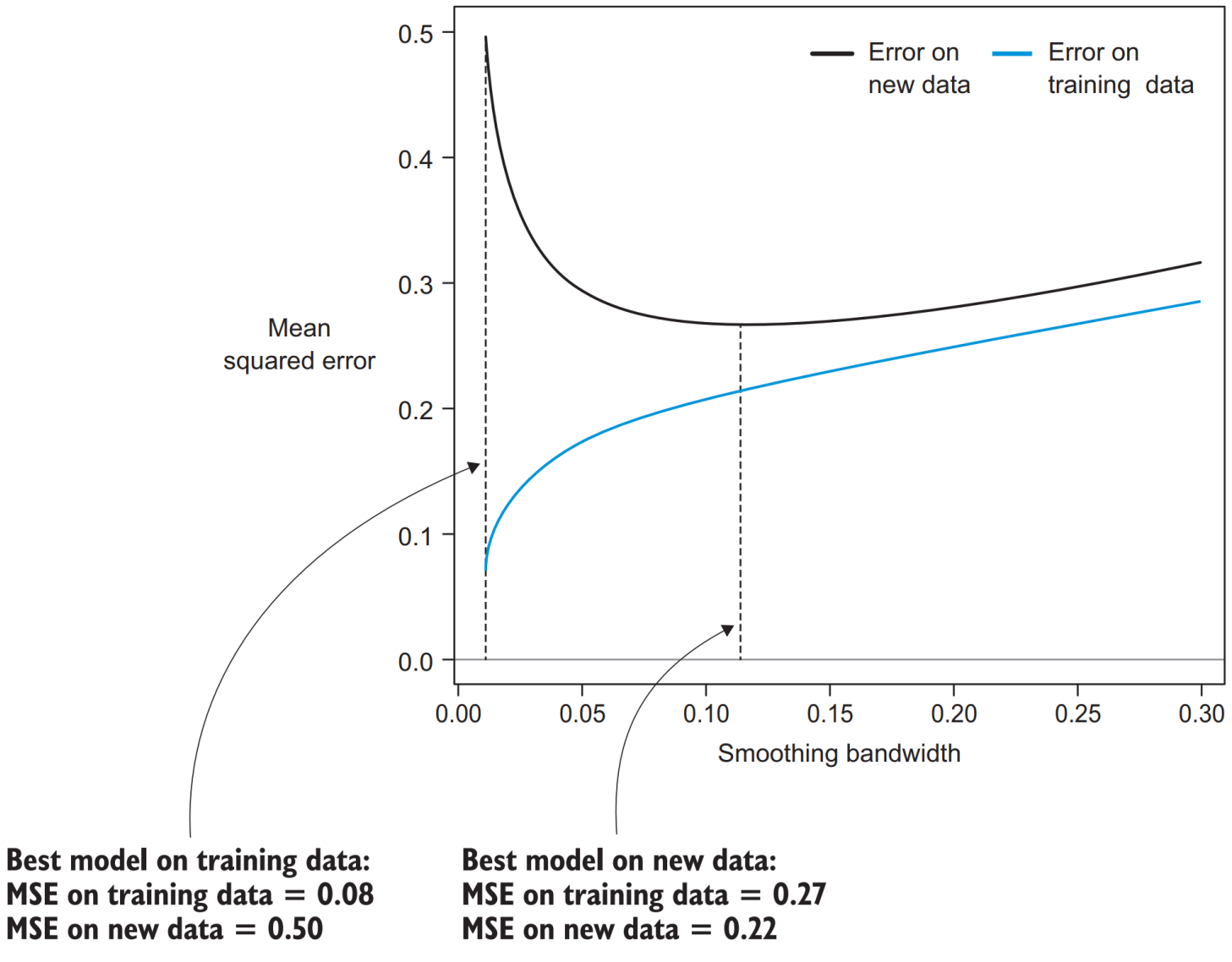
- Determining how well ML model will generalize to predict the corn output from data on different farms
- The first step in this process is to select an evaluation metric that captures the quality of the predictions
- The simplest form of performance measurement of a regression model is the root mean-square error, or RMSE
- This estimator looks at the difference from each of the predicted values to the known values, and calculates the mean in a way that's immune to the fact that predicted values can be both higher and lower than the actual values

RMSE


Predicted MPG
MPG
40
35
30
25
20
15
10
5
10
15
20
25
30
35
40
45
50
$\frac{1}{\sqrt{n}}\sqrt{\sum [y_i - f(x_i)]^2}$
Find the distance to the optimal model for every point in the predicted vs. true scatter plot
Calculate the root of the mean of the squared distances

Pitfall – improve RMSE on training

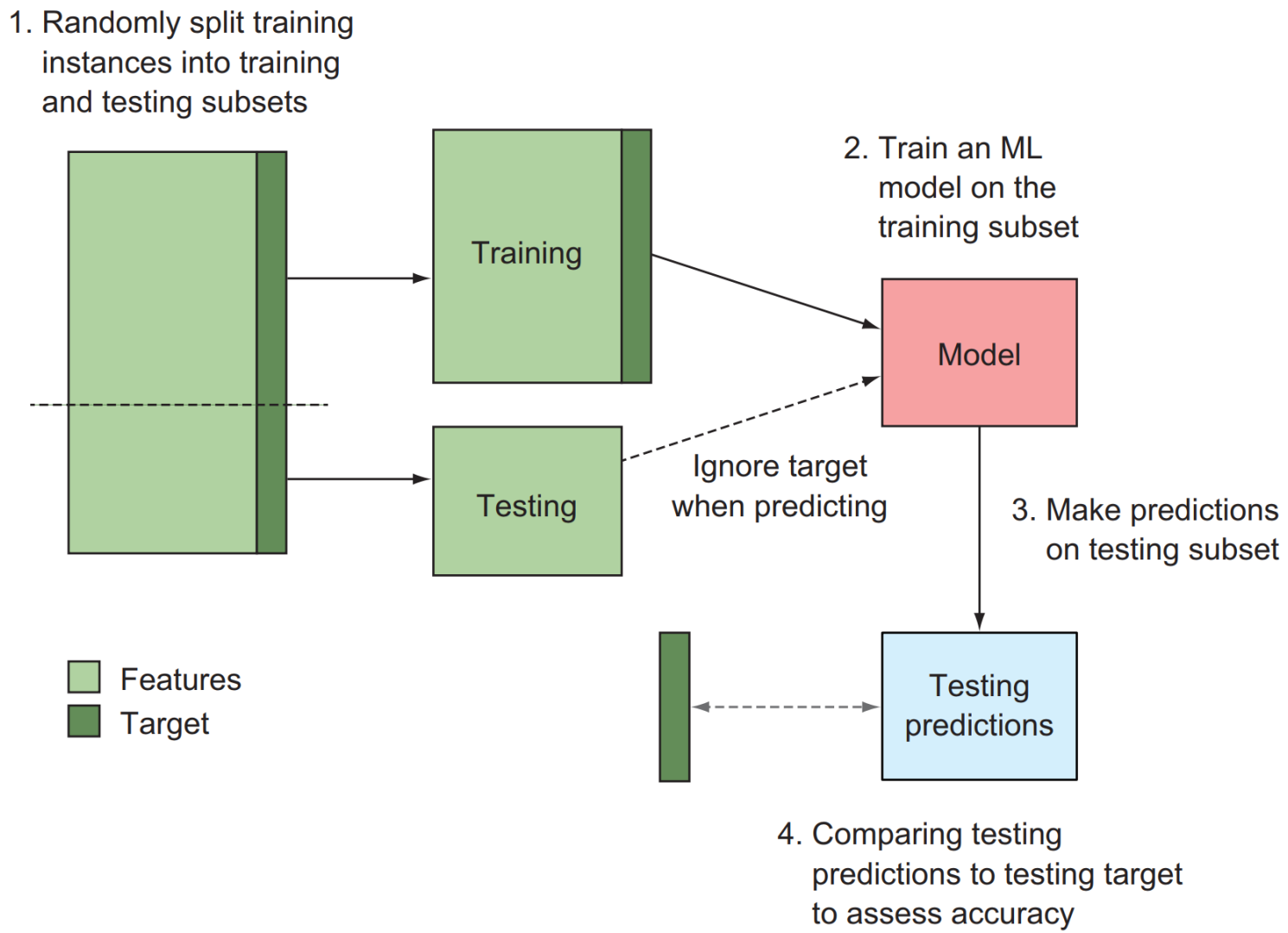
- For small values of the bandwidth parameter, the MSE evaluated on the training set is extremely small
- Whereas the MSE evaluated on new data (in this case, 10,000 new instances) is much larger
- Simply put, the performance of the predictions of a model evaluated on the training set isn't indicative of the performance of that model on new data



Solution – cross validation

- We've diagnosed the challenge in model evaluation: the training set error isn't indicative of the model error when applied to new data
- To get a good estimate of what error rate will be for new data, we must use a more sophisticated methodology called *cross-validation* (often abbreviated *CV*) that rigorously employs the training set to evaluate what the accuracy will be on new data
- There are two kinds of cross validation methods
  - Hold out method
  - K-fold method

Hold out cross validation



Hold out cross validation

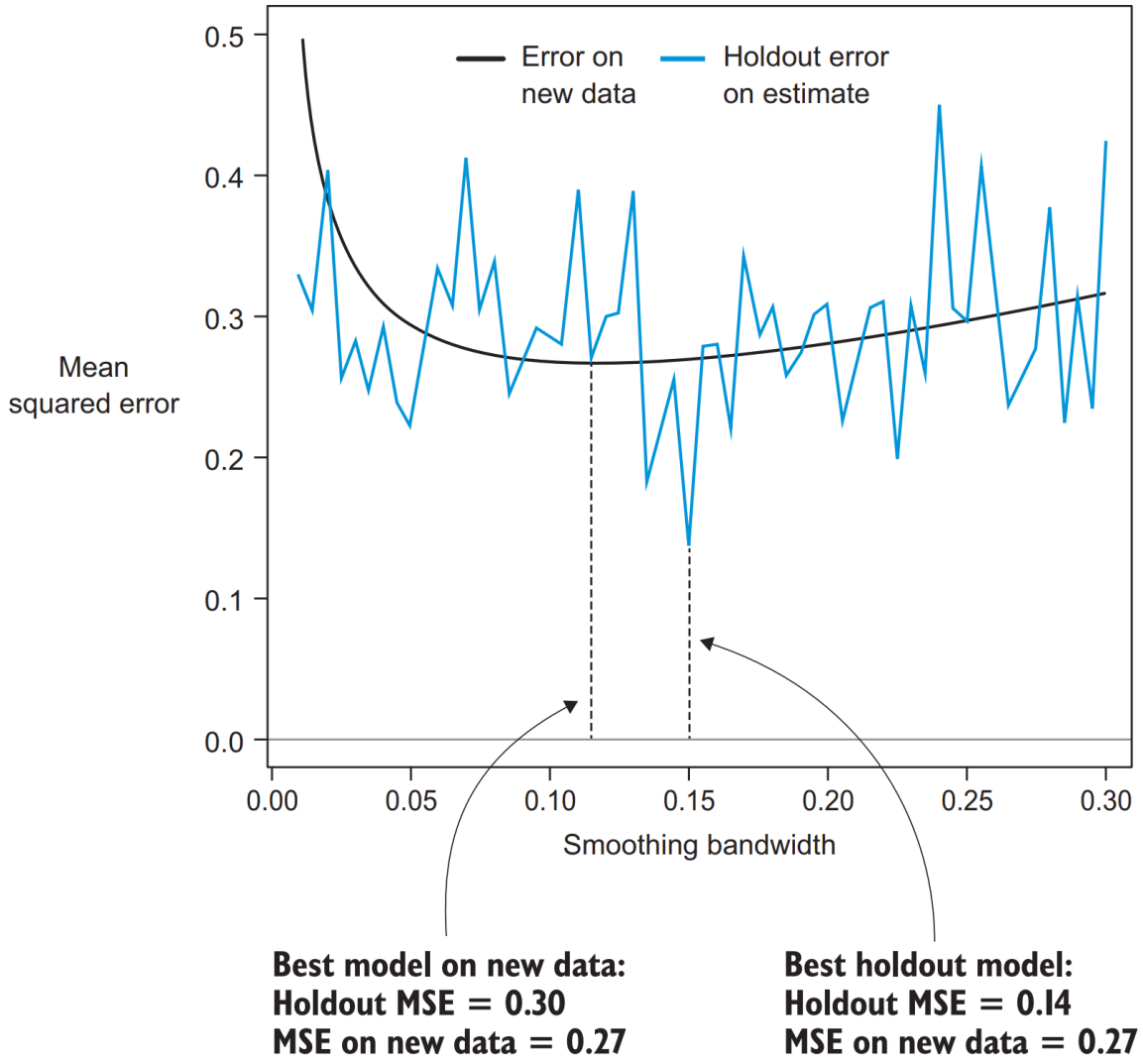
```
# assume that we begin with two inputs:
# features – a matrix of input features
# target –
 an array of target variables corresponding to those featur
es

features = rand(100,5)
target = rand(100) > 0.5
N = features.shape[0] # The total number of instances
N_train = floor(0.7 * N) # The total number of training insta
nces

idx = random.permutation(N)
```

```
idx_train = idx[:N_train]
idx_test = idx[N_train:]

features_train = features[idx_train,:]
target_train = target[idx_train]

features_test = features[idx_test,:]
target_test = target[idx_test]


# Build, predict, evaluate (to be filled out)
# model = train(features_train, target_train)
# preds_test = predict(model, features_test)
# accuracy = evaluate_acc(preds_test, target_test)
```

Hold out cross validation

- If we apply the holdout method to the corn production data using 70/30 split we may get the graph of the next slide
- Two main things stand out
  - The error estimates computed by the holdout method are close to the new-data error of the model
  - They're certainly much closer than the training set error estimates, particularly for small-bandwidth values
  - The holdout error estimates are noisy
  - They bounce around wildly compared to the smooth curve that represents the error on new data

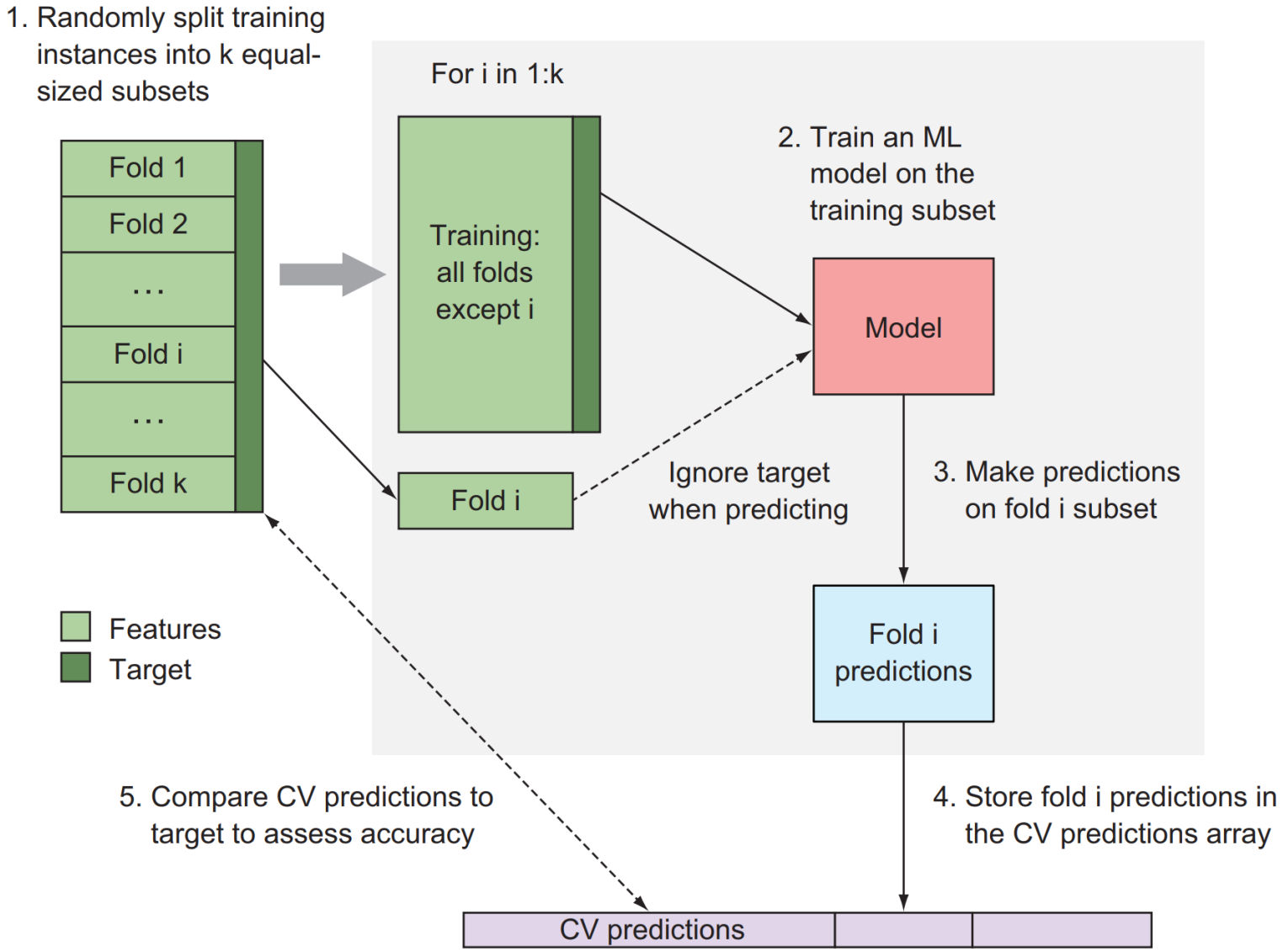
Hold out cross validation



K-fold cross validation

- A better but more computationally intensive approach to cross-validation is *k-fold cross validation*
- Like the holdout method, k-fold cross-validation relies on quarantining subsets of the training data during the learning process
- The primary difference is that k-fold CV begins by randomly splitting the data into *k* disjoint subsets, called *folds* (typical choices for k are 1, 5, 10, or 20)
- When k is 1, it is called leave one out cross validation
- For each fold, a model is trained on all the data *except* the data from that fold and is subsequently used to generate predictions for the data from that fold
- After all k-folds are cycled through, the predictions for each fold are aggregated and compared to the true target variable to assess accuracy

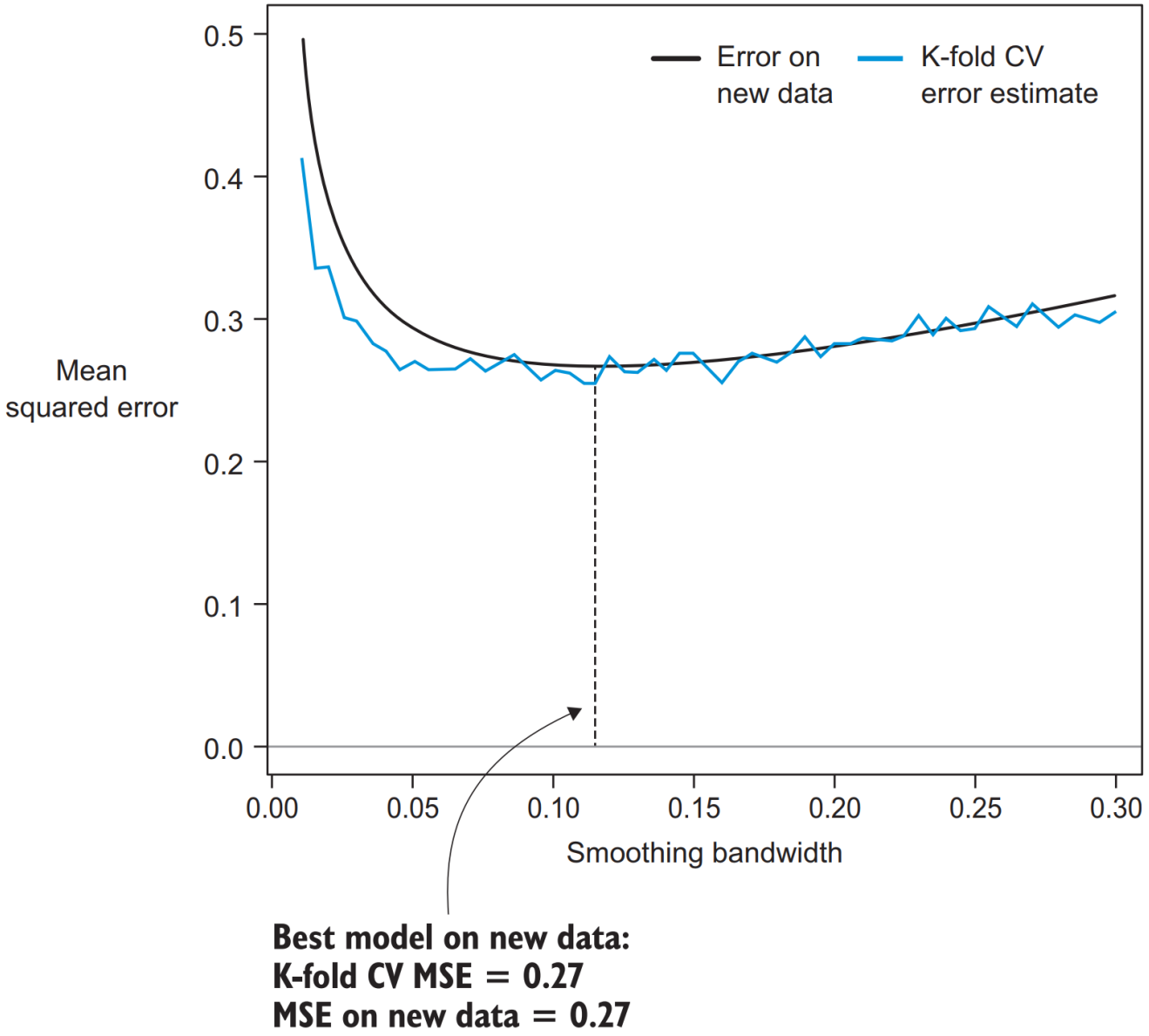
K-fold cross validation



K-fold cross validation: pseudo code

```
N = features.shape[0]
K = 10 # number of folds
preds_kfold = np.empty(N)
folds = np.random.randint(0, K, size=N)

for idx in np.arange(K):
    features_train = features[folds != idx,:]
    target_train = target[folds != idx]
    features_test = features[folds == idx,:]

# Build and predict for CV fold (to be filled out)
# model = train(features_train, target_train)
# preds_kfold[folds == idx] = predict(model, features_test)
# accuracy = evaluate_acc(preds_kfold, target)
```

K-fold cross validation



Choose classification models

- We want to predict whether a Titanic passenger would survive, based on personal, social, and economic factors
- We want to train a classifier that could relate all this information we have to the passengers' survival probability

Target column

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	Male	22	1	0	A/5 21171	7.25	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th…	Female	38	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Helkkinen, Miss Laina	Female	26	0	0	STON/02. 3101282	7.925	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	Female	35	1	0	113803	53.1	C123	S
4	5	0	3	Allen, Mr. William Henry	Male	35	0	0	373450	8.05	NaN	S

Cross validation intuition – again

- Frist step – cross-validation
- We divide the full dataset into training and testing sets and use the holdout method/k-fold of cross-validation
- The model will be built on a training set and evaluated on a held-out testing set
- It’s important to reiterate that our goal isn’t necessarily to obtain the maximum model accuracy on the training data
- But to obtain the highest predictive accuracy on unseen data
- In the model-building phase, we are not yet in possession of this data, by definition, so we pretend that some of the training data is hidden for the learning algorithm

Cross validation intuition – again

Full dataset

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	Male	22	1	0	A/5 21171	7.25	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th…	Female	38	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Helkkinen, Miss Laina	Female	26	0	0	STON/02. 3101282	7.925	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	Female	35	1	0	113803	53.1	C123	S
4	5	0	3	Bannister, Mr. Victor Brian	Male	31	0	0	362400	8.63	C20	C
5	6	1	1	Allen, Mr. William Henry	Male	35	0	0	373450	8.05	NaN	S
6	7	0	3	Mcneeley, Mr. Mike Paul	Male	43	1	0	281654	9.25	C65	C
7	8	1	1	Boden, Mrs Elaina Rose	Female	55	1	0	985111	10.56	NaN	S

Training set: used only for building the model

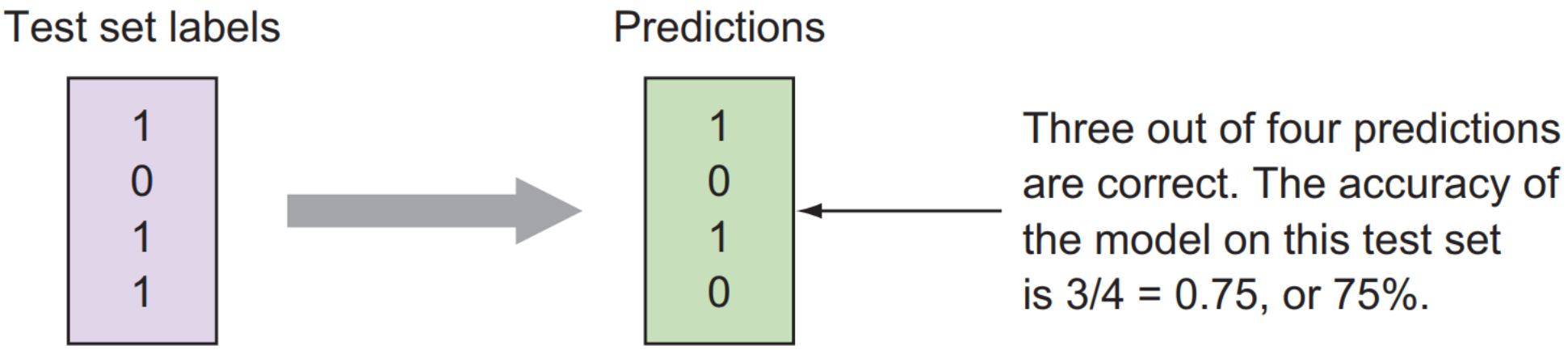
	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	Male	22	1	0	A/5 21171	7.25	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th…	Female	38	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Helkkinen, Miss Laina	Female	26	0	0	STON/02. 3101282	7.925	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	Female	35	1	0	113803	53.1	C123	S
4	5	0	3	Bannister, Mr. Victor Brian	Male	31	0	0	362400	8.63	C20	C

Testing set: used only for evaluating model

5	6	1	1	Allen, Mr. William Henry	Male	35	0	0	373450	8.05	NaN	S
6	7	0	3	Mcneeley, Mr. Mike Paul	Male	43	1	0	281654	9.25	C65	C
7	8	1	1	Boden, Mrs Elaina Rose	Female	55	1	0	985111	10.56	NaN	S

Accuracy metric – MSE for classification?

- The simplest performance measure of a classification model is to calculate the fraction of *correct* answers
- If three out of four rows were correctly predicted, we can say that the *accuracy* of the model on this particular validation set is 3/4 = 0.75, or 75%

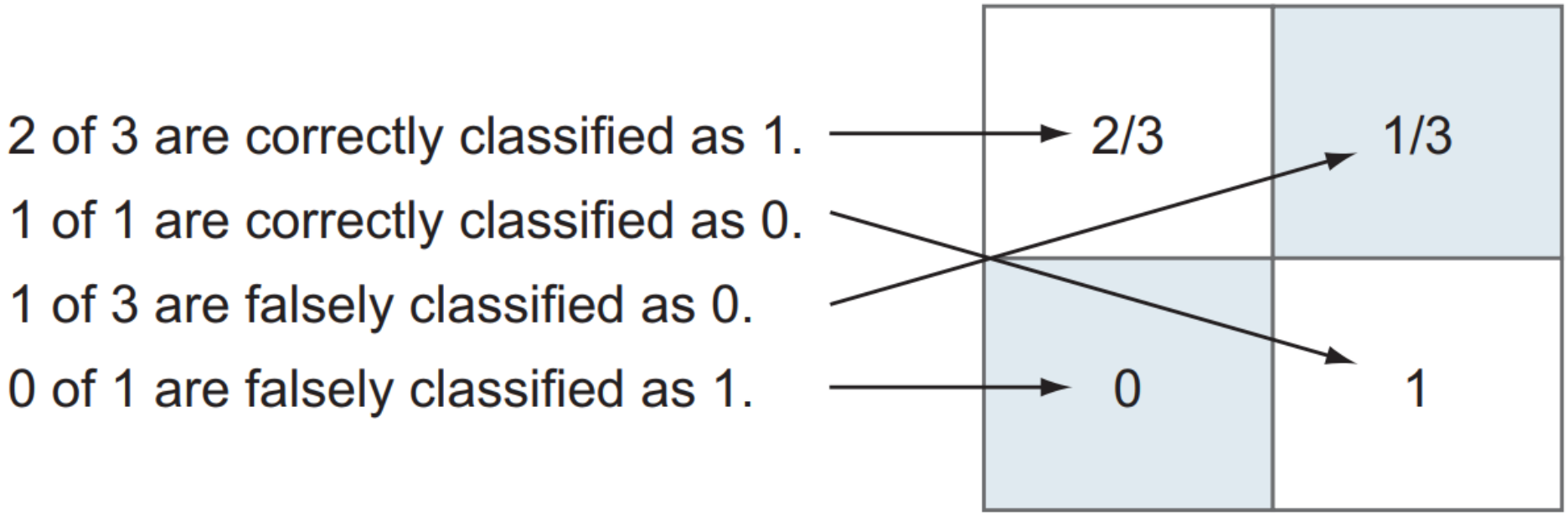


Class accuracy

- The predictions should provide more information than simply being correct or not
- We should be able to analyze the accuracy per class (how many were predicted to survive but actually died or survived)
- For binary classification, you can be wrong in two ways – predicting 0 when the correct value is 1, or predicting 1 when the correct value is 0
- In many classification problems, it's useful to go beyond the simple counting accuracy and look at this class-wise accuracy, or class confusion

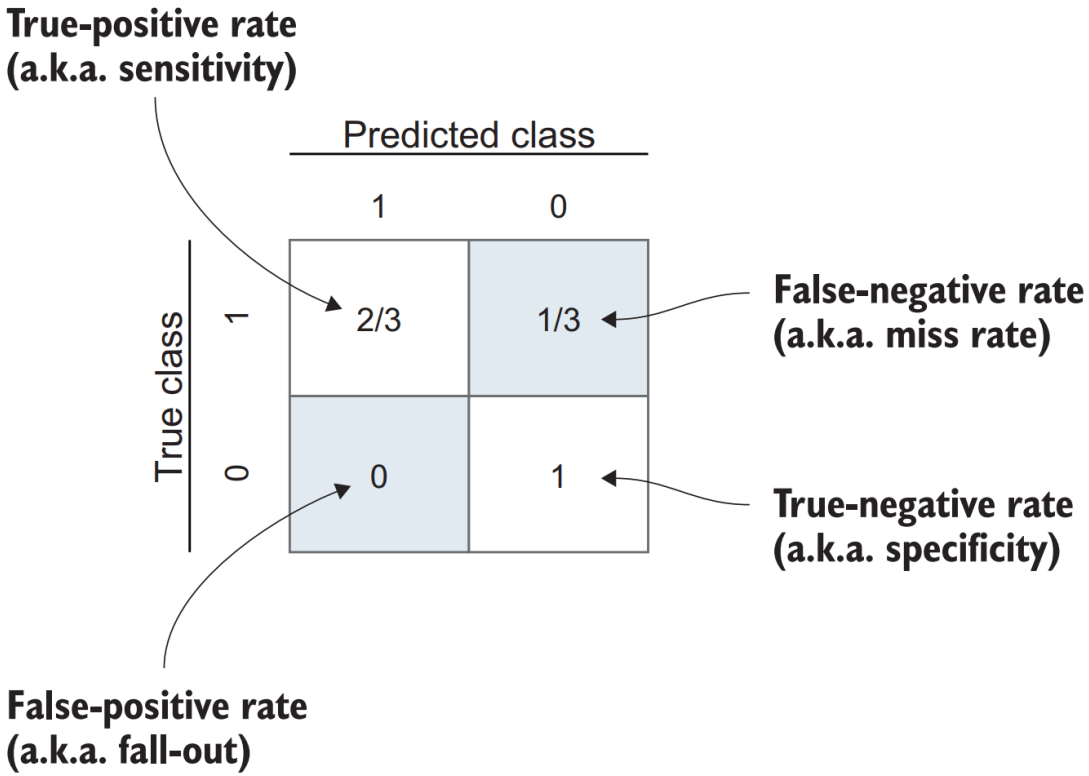
Accuracy – confusion matrix

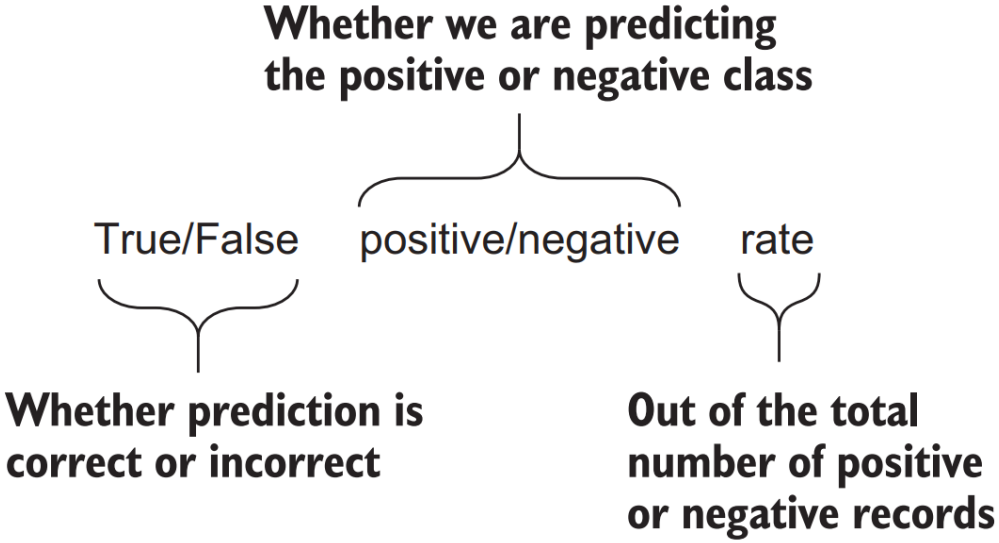
- It turns out to be useful to display these four numbers in a two-by-two diagram called a *confusion matrix*



Confusion matrix

- Each element in the matrix shows the class-wise accuracy or confusion between the positive and the negative class
- They are related to the general concept of receiver operating characteristics (ROCs)





Accuracy tradeoffs

- So far we have looked only at predictions for which the output is the predicted class; in our Titanic example, 1 for survival and 0 otherwise
- Machine-learning predictions usually hold a degree of uncertainty, and many classification algorithms output not only the zero-one predictions, but the full prediction *probabilities*
- For example, what was simply predicted as *survived* in our Titanic model may have had a probability of survival of 0.8, 0.99, or 0.5
- It's clear that there's a big difference in the confidence of these answers

ROC Curve

- The output of a *probabilistic classifier* is the *probability vectors* or *class probabilities*
- For every row in the test set, we get a real-valued number from 0 to 1 for every class in the classifier (summing to 1)
- Until now, you've made predictions by considering probabilities above 0.5 to determine the class predictions, from which we calculated all the performance metrics
- We say that the *threshold* that determines the class is 0.5
- It's clear that we could choose any other threshold and would get different values for all of our metrics

Threshold for survivality

Output from classifier: class probabilities

	Survived	Died
15	0.092	0.908
16	0.904	0.096
17	0.646	0.354
18	0.740	0.260
19	0.460	0.540

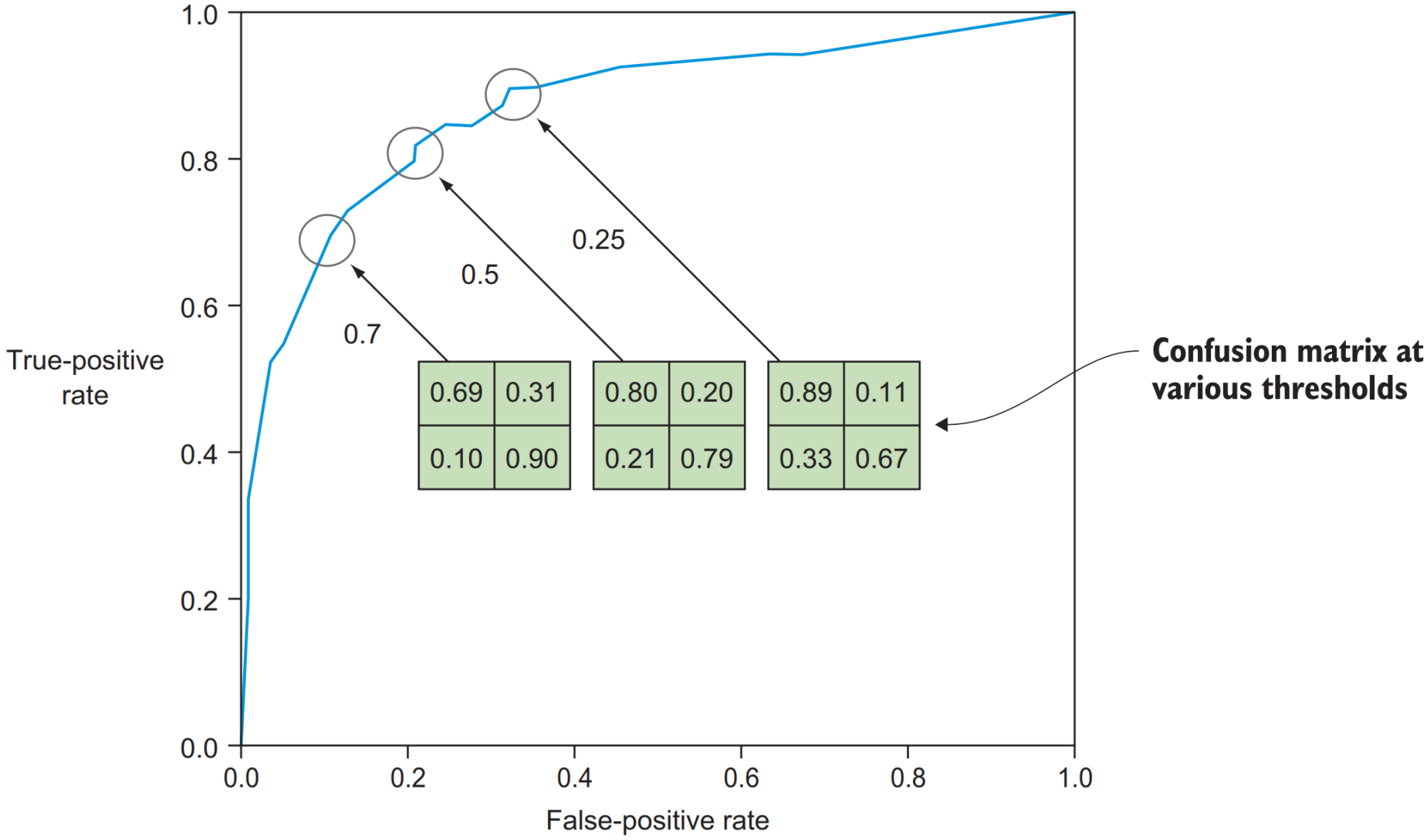
Sorted probabilities

	Survived	Died
308	0.705	0.295
215	0.703	0.297
217	0.700	0.300
54	0.698	0.302
169	0.698	0.302

Threshold: "survived" probabilities > 0.7

ROC Curve

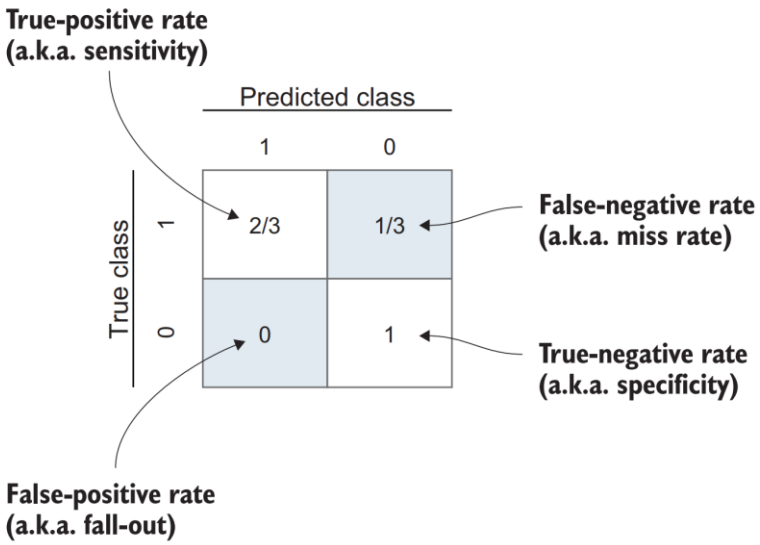
- Figure in next slide shows the process of sorting the probability vectors and setting a threshold of 0.7.
- All rows above the line are now predicted to survive, and you can compare these to the actual labels to get the confusion matrix and the ROC metrics at this particular threshold
- If we follow this process for all thresholds from 0 to 1, we get the *ROC curve*, shown in next slide
- Here we can read out the confusion matrix at all thresholds, making the ROC curve a powerful visualization tool when we're evaluating classifier performance


1.0
0.8
0.6
0.4
0.2
0.0
True-positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False-positive rate
0.7
0.5
0.25
0.69
0.31
0.10
0.90
0.80
0.20
0.21
0.79
0.89
0.11
0.33
0.67
Confusion matrix at various thresholds

Definitions

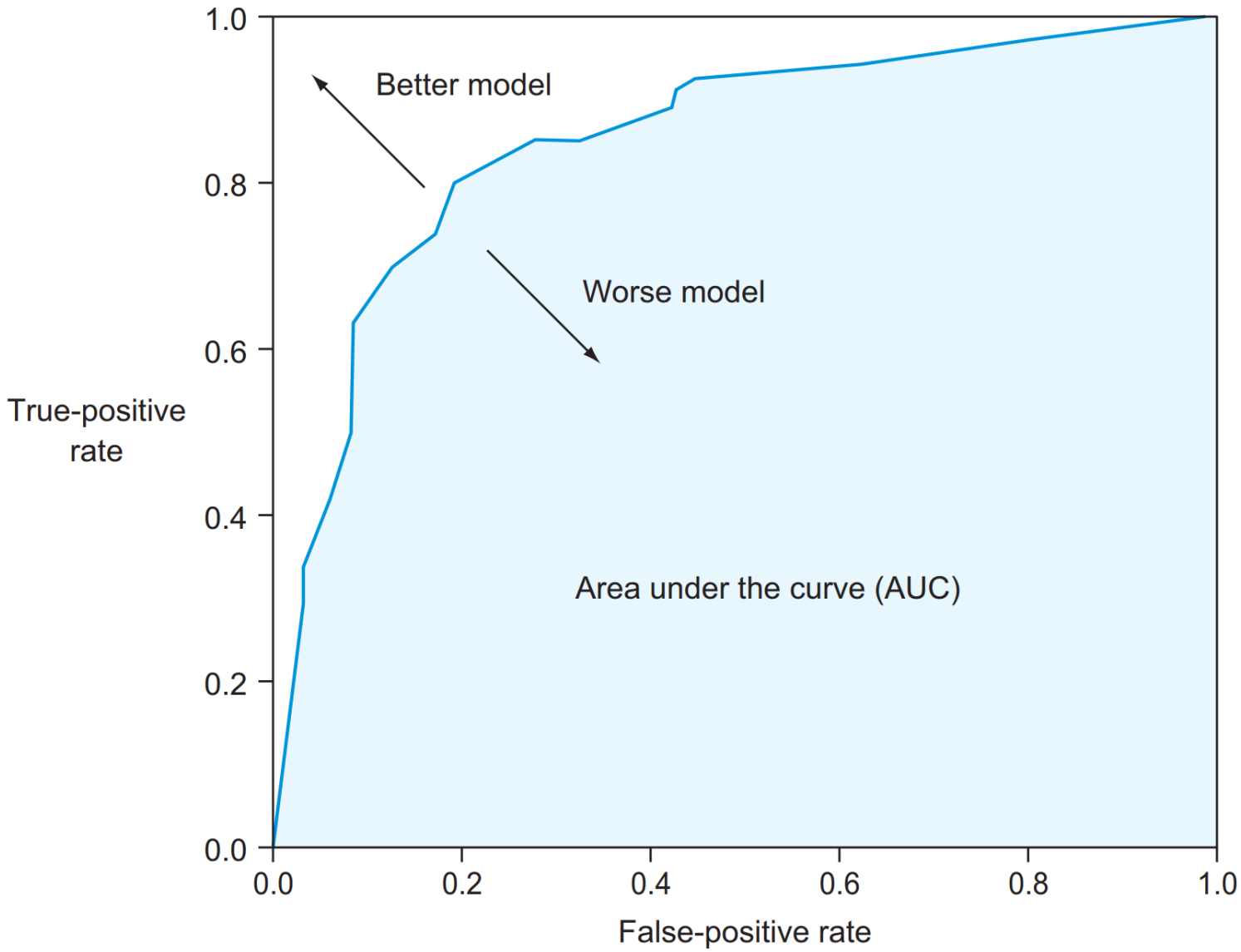
$$\text{True Positive Rate} = \text{Sensitivity} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

$$\text{False Positive Rate} = (1 - \text{Specificity}) = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}}$$



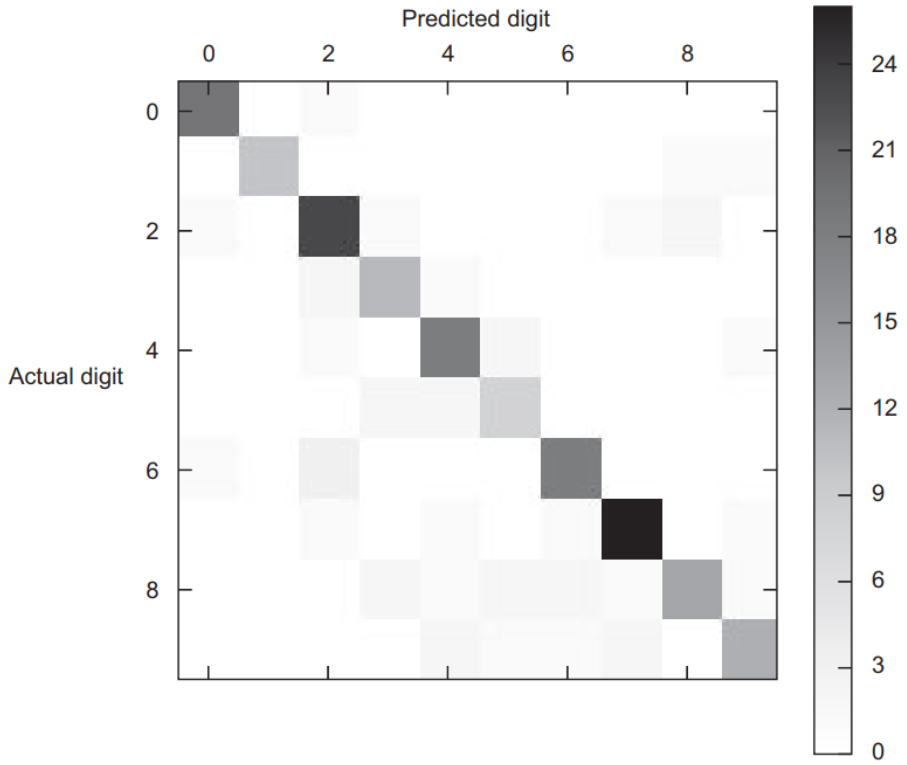
AUC

- The ROC curve itself also provides a view of the overall performance of the classifier
- A perfect classifier would have no false positives and no missed detections, so the curve would be pushed to the top-left corner, as illustrated in the next (slide's) figure
- This leads us naturally to another evaluation metric: the area under the ROC curve (AUC)
- The larger this area, the better the classification performance
- The AUC is a widely used choice for evaluating and comparing models, although in most cases it's important to inspect the full ROC curve

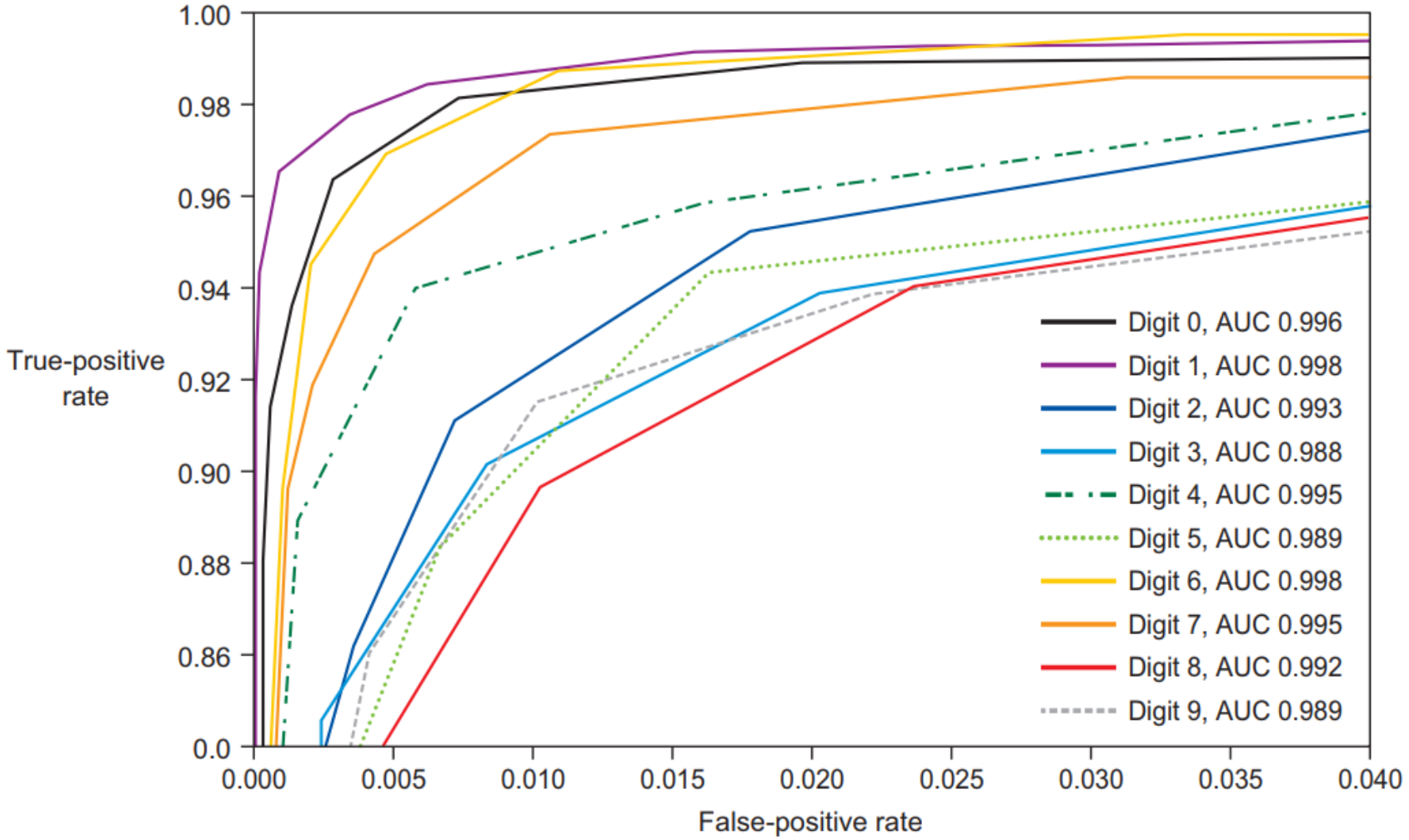

1.0
0.8
0.6
0.4
0.2
0.0
True-positive rate
Better model
Worse model
Area under the curve (AUC)
0.0
0.2
0.4
0.6
0.8
1.0
False-positive rate

Multiclass confusion matrix

- So far you've looked only at binary, or two-class, classification problems, but we can use the same tools for multiclass classifiers
- A well-known multiclass classification problem is handwritten digit recognition



Multiclass classification AUC



Information Criteria

- **Information criterion** is a measure of the goodness of fit of an estimated statistical model.
- It is grounded in the concept of entropy,
  - Offers a relative measure of the information lost
  - Describes the tradeoff precision and complexity of the model
- An IC is not a test on the model in the sense of hypothesis testing
- It is a tool for model selection
- Given a data set, several competing models may be ranked according to their IC
- The model with the lowest IC is chosen as the “best”

Information Criteria

- IC rewards goodness of fit, but also includes a penalty that is an increasing function of the number of estimated parameters.
- This penalty discourages overfitting.
- The IC methodology attempts to find the model that **best explains the data with a minimum of free parameters**.
- IC judges a model by how close its fitted values tend to be to the true values.
- the AIC value assigned to a model is only meant to *rank* competing models and tell you which is the best among the given alternatives.

Akaike Information Criteria (AIC)

$$AIC = -2\log Lik + 2p$$

Akaike, Hirotugu (1974). "A new look at the statistical model identification". *IEEE Transactions on Automatic Control* **19** (6): 716–723..

Bayesian Information Criteria

$$BIC = -2\log Lik + p\ln(N)$$

Schwarz, Gideon E. (1978). "Estimating the dimension of a model". *Annals of Statistics* **6** (2): 461–464.

AIC versus BIC

$$2p \;\; vs. \;\; p\ln(N)$$

- BIC and AIC are similar
- Different penalty for number of parameters
- The BIC penalizes free parameters more strongly than does the AIC.
- Implications: BIC tends to choose smaller models
- **The larger the N, the more likely that AIC and BIC will disagree on model selection**