GENERAL ASSEMBLY

INTRO TO DATA SCIENCE

LECTURE 3: SUPERVISED LEARNING

LAST TIME:

- GGPLOT
- INTRO TO MACHINE LEARNING & TYPICAL PROBLEMS
- KNN CLASSIFICATION

QUESTIONS?

I. CLASSIFICATION PROBLEMS
II. INTRODUCTION TO REGRESSION

I. CLASSIFICATION PROBLEMS

	continuous	*categorical*
supervised	???	???
unsupervised	???	???

	continuous	*categorical*
supervised	regression	classification
unsupervised	dimension reduction	clustering

Here's (part of) an example dataset:

Fisher's *Iris* Data

Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	*I. setosa*
4.9	3.0	1.4	0.2	*I. setosa*
4.7	3.2	1.3	0.2	*I. setosa*
4.6	3.1	1.5	0.2	*I. setosa*
5.0	3.6	1.4	0.2	*I. setosa*
5.4	3.9	1.7	0.4	*I. setosa*
4.6	3.4	1.4	0.3	*I. setosa*
5.0	3.4	1.5	0.2	*I. setosa*

Here's (part of) an example dataset:

Fisher's *Iris* Data

Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	*I. setosa*
4.9	3.0	1.4	0.2	*I. setosa*
4.7	3.2	1.3	0.2	*I. setosa*
4.6	3.1	1.5	0.2	*I. setosa*
5.0	3.6	1.4	0.2	*I. setosa*
5.4	3.9	1.7	0.4	*I. setosa*
4.6	3.4	1.4	0.3	*I. setosa*
5.0	3.4	1.5	0.2	*I. setosa*

independent variables

Here's (part of) an example dataset:

Fisher's *Iris* Data

Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	*I. setosa*
4.9	3.0	1.4	0.2	*I. setosa*
4.7	3.2	1.3	0.2	*I. setosa*
4.6	3.1	1.5	0.2	*I. setosa*
5.0	3.6	1.4	0.2	*I. setosa*
5.4	3.9	1.7	0.4	*I. setosa*
4.6	3.4	1.4	0.3	*I. setosa*
5.0	3.4	1.5	0.2	*I. setosa*

independent variables

class labels *(qualitative)*

Q: What does “supervised” mean?

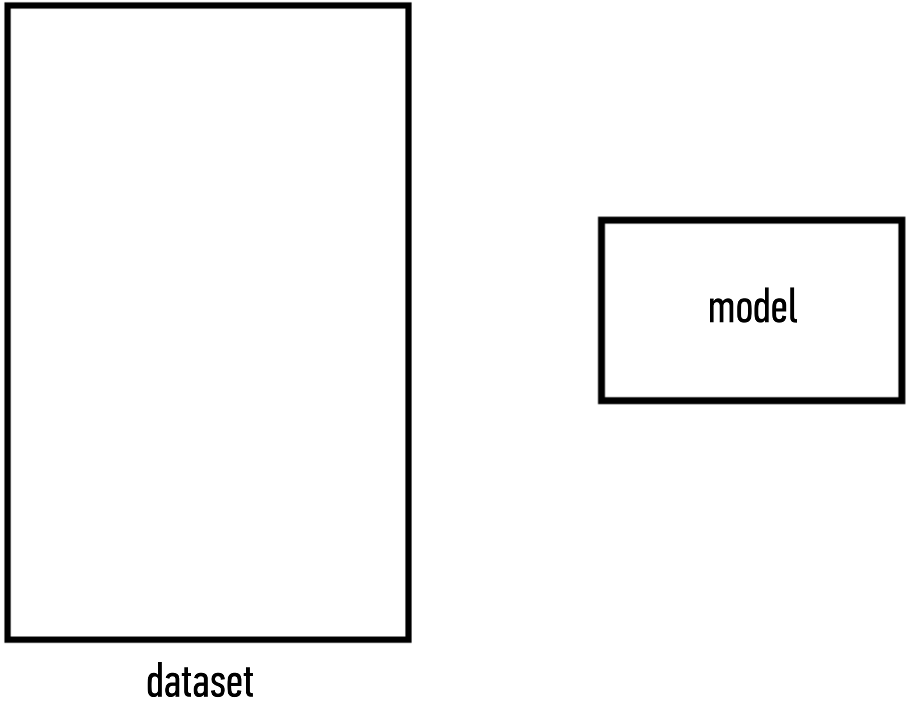
Q: What does "supervised" mean?

A: We know the labels.

```
Welcome to R! Thu Feb 28 13:07:25 2013
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width   
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100  
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300  
 Median :5.800   Median :3.000   Median :4.350   Median :1.300  
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199  
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800  
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500  
       Species  
 setosa    :50  
 versicolor:50  
 virginica :50  
```

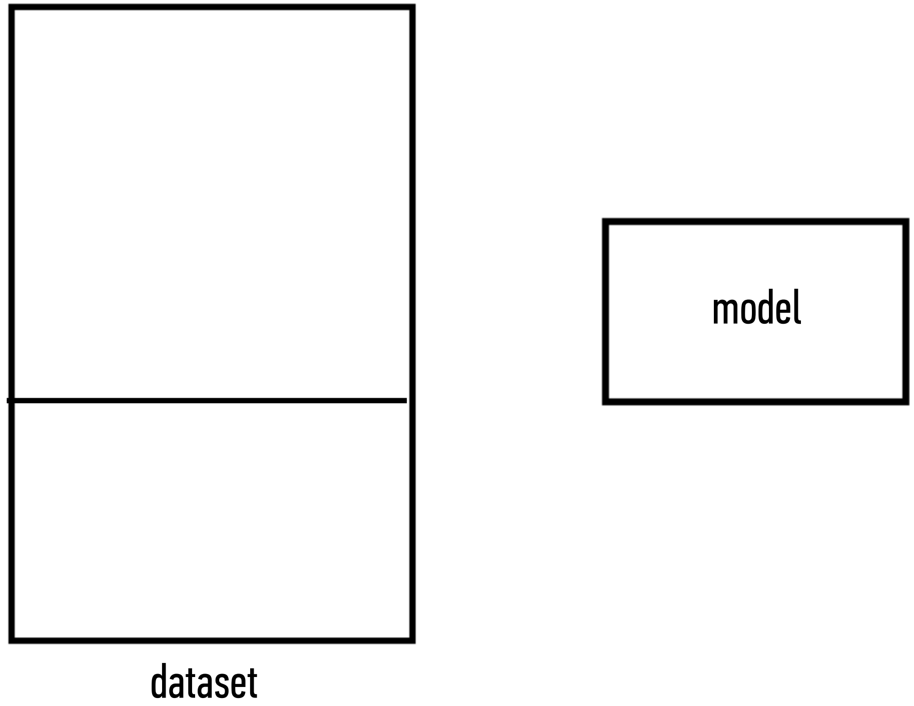
Q: How does a classification problem work?

Q: What steps does a classification problem require?



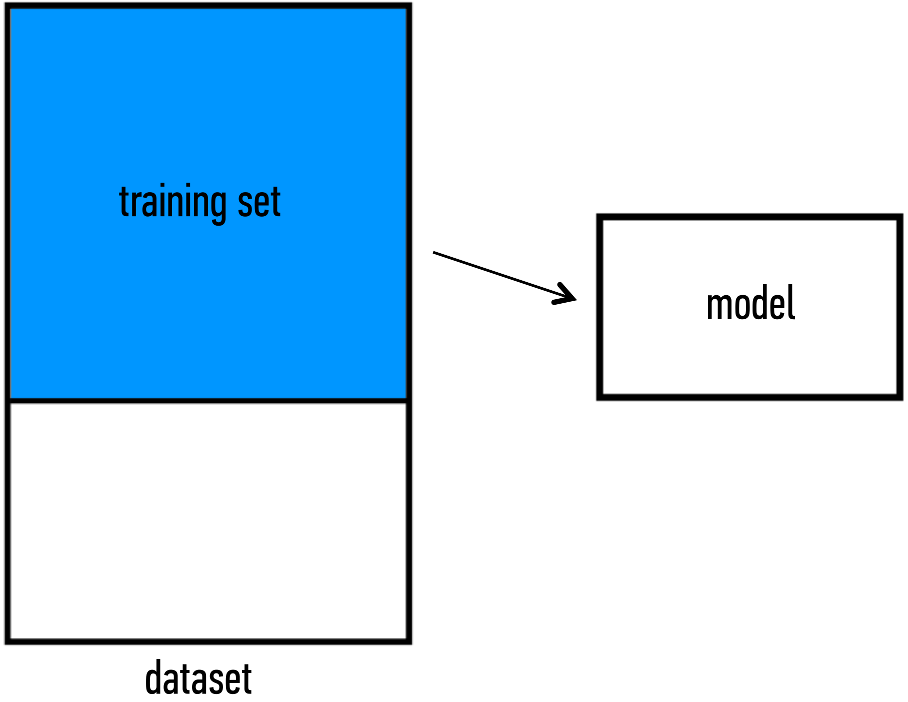
Q: What steps does a classification problem require?

1) split dataset



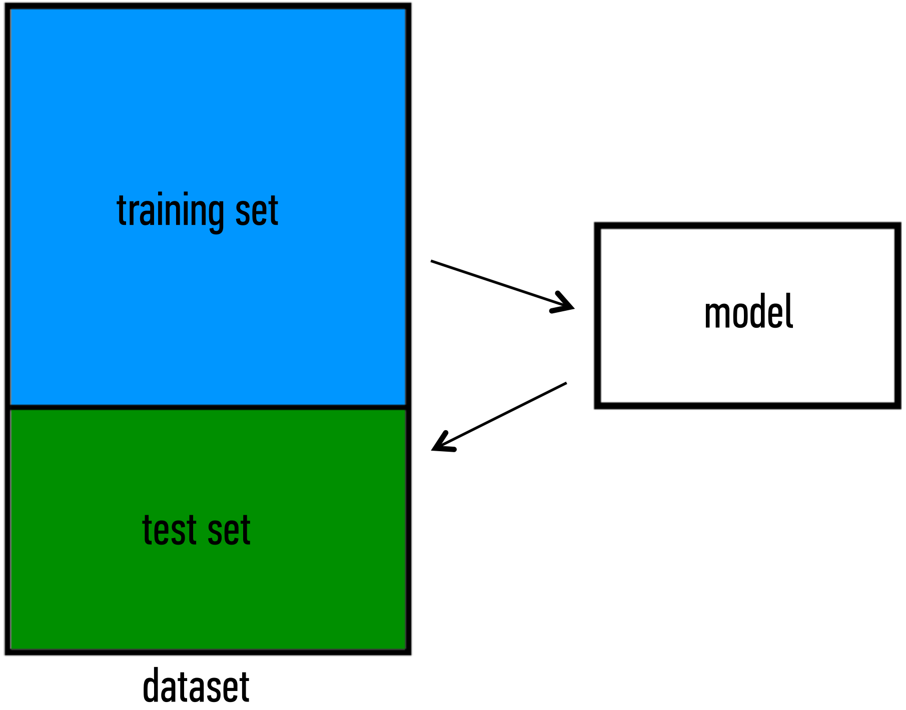
Q: What steps does a classification problem require?

1) split dataset
2) train model



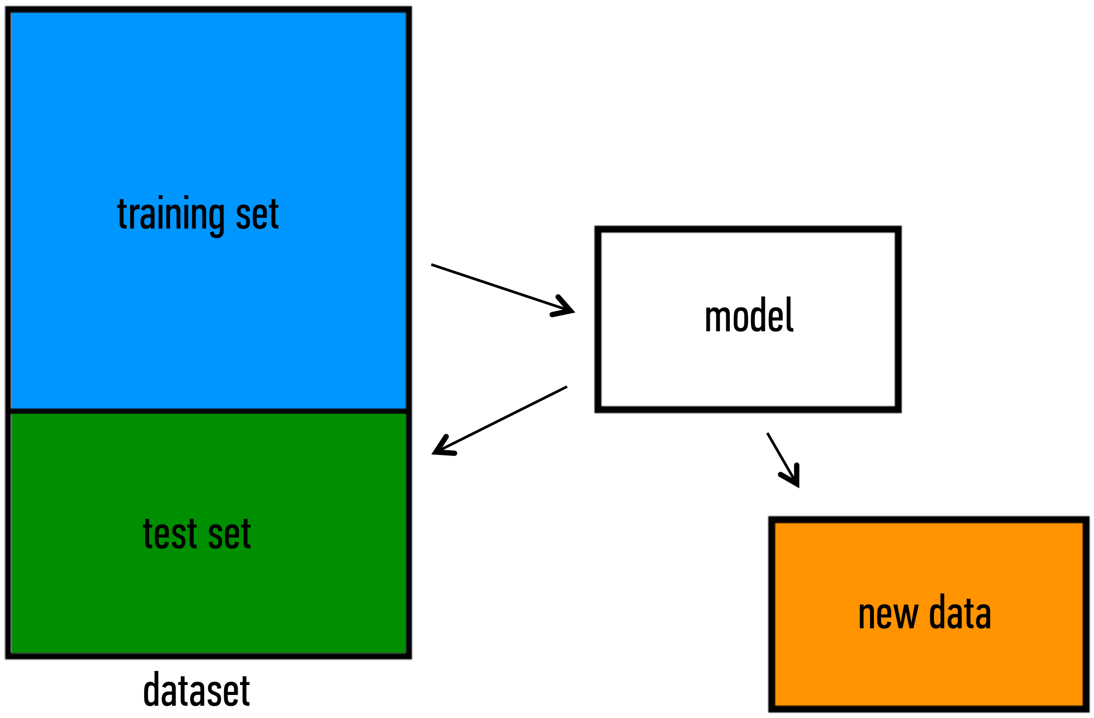
Q: What steps does a classification problem require?

1) split dataset
2) train model
3) test model



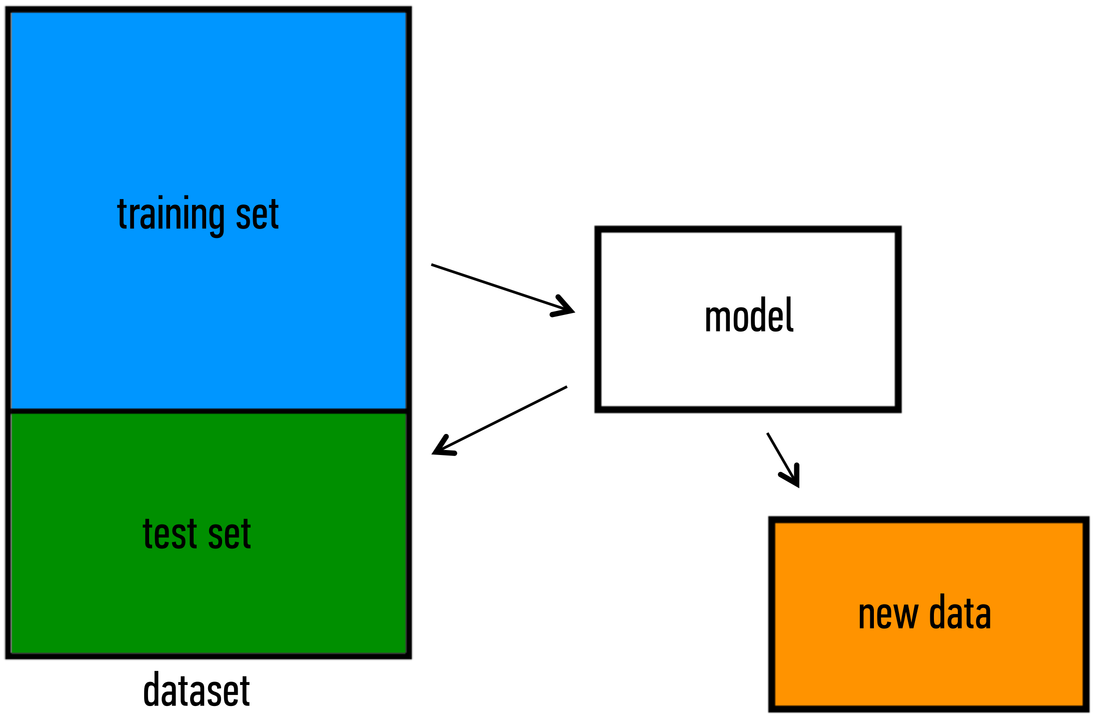
Q: What steps does a classification problem require?

1) split dataset
2) train model
3) test model
4) make predictions



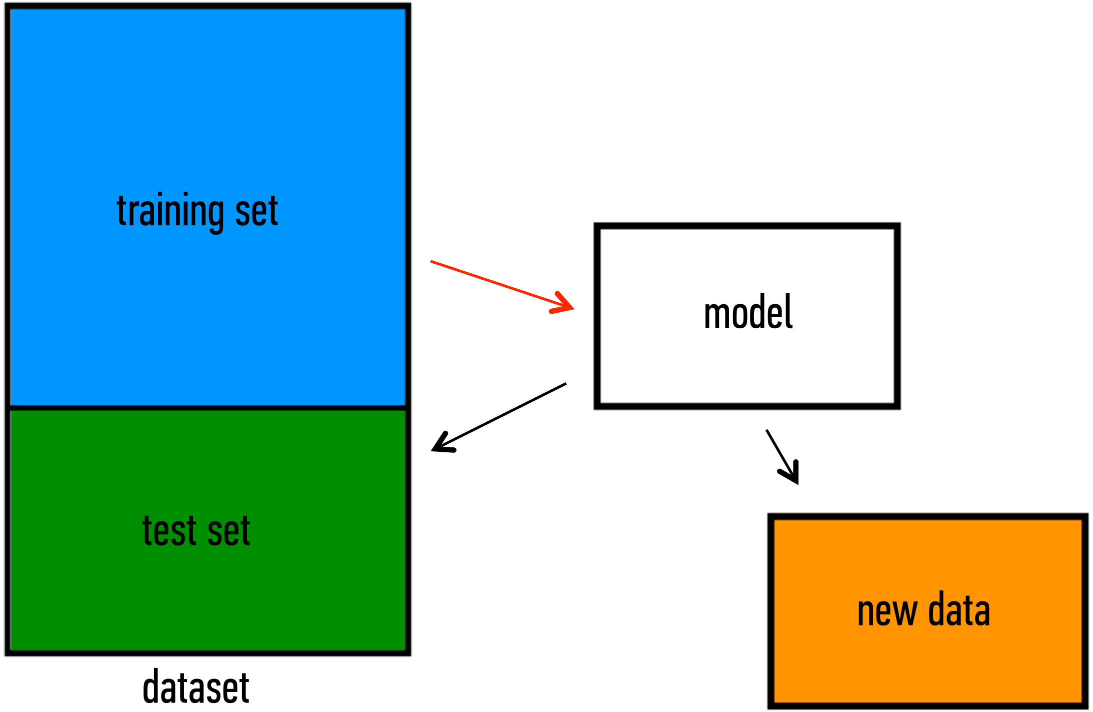
II. BUILDING EFFECTIVE CLASSIFIERS

Q: What types of prediction error will we run into?



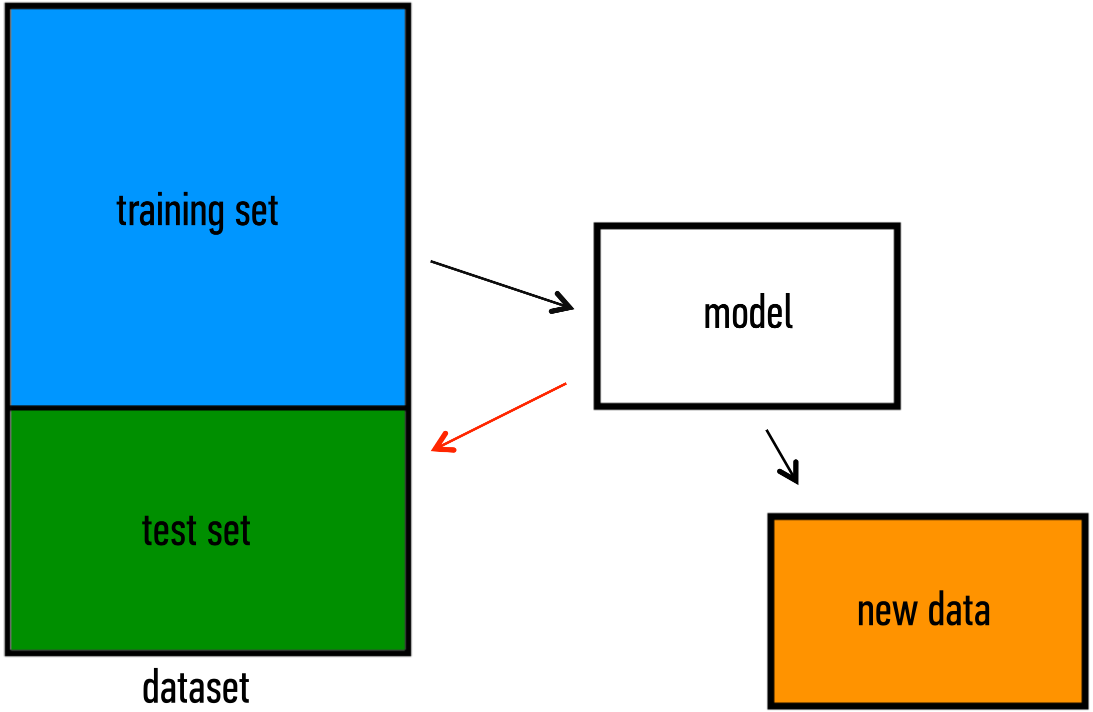
Q: What types of prediction error will we run into?

1) training error



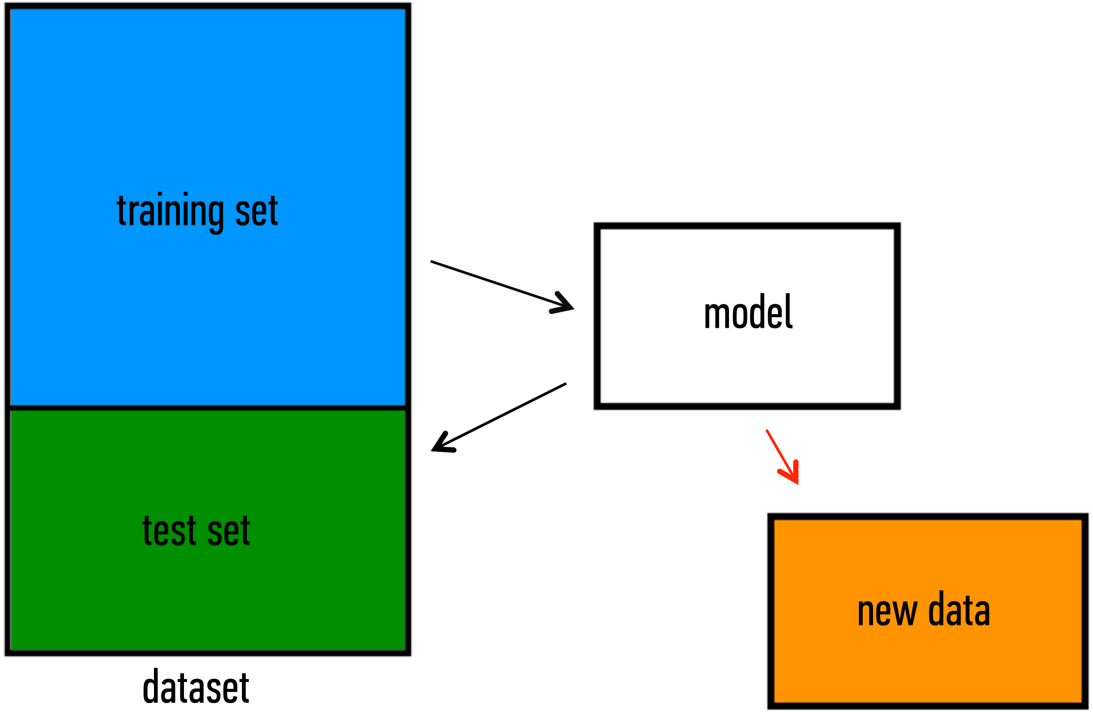
Q: What types of prediction error will we run into?

1) training error
2) generalization error



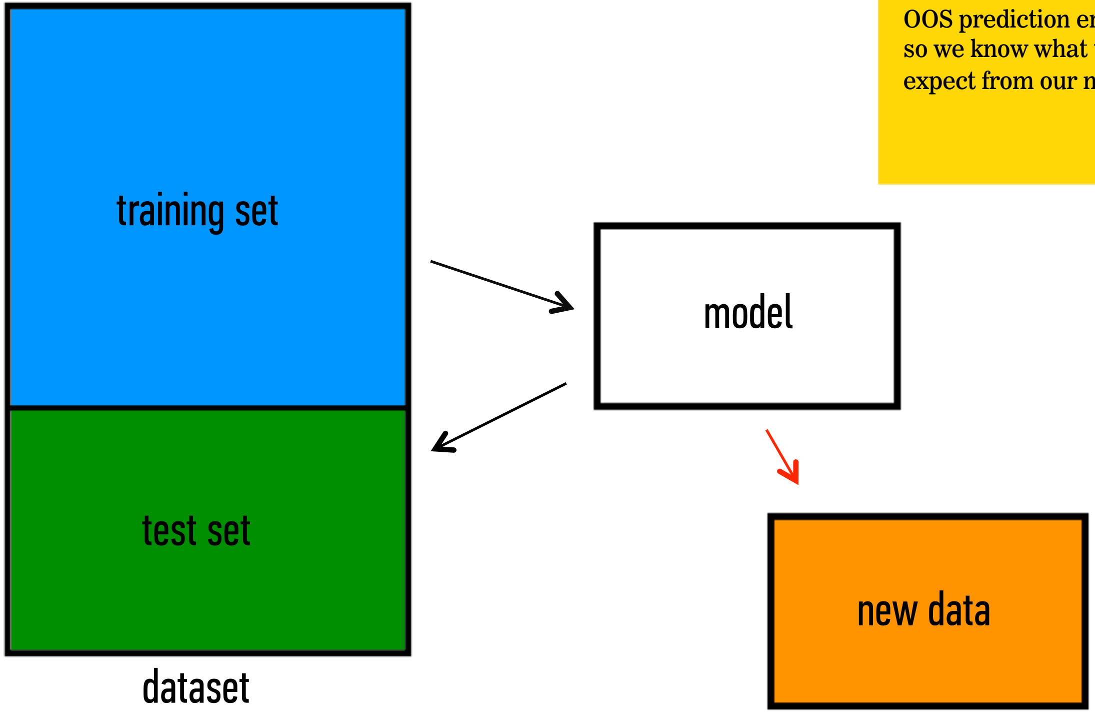
Q: What types of prediction error will we run into?

1) training error
2) generalization error
3) OOS error



Q: What types of prediction error will we run into?

1) training error
2) generalization error
3) OOS error



NOTE

We want to estimate OOS prediction error so we know what to expect from our model.

Q: Why should we use training & test sets?

Q: Why should we use training & test sets?

Thought experiment:
Suppose instead, we train our model using the entire dataset.

Q: Why should we use training & test sets?

Thought experiment:

Suppose instead, we train our model using the entire dataset.

Q: How low can we push the training error?

Q: Why should we use training & test sets?

Thought experiment:
Suppose instead, we train our model using the entire dataset.
Q: How low can we push the training error?

- *We can make the model arbitrarily complex (effectively "memorizing" the entire training set).*

Q: Why should we use training & test sets?

Thought experiment:

Suppose instead, we train our model using the entire dataset.

Q: How low can we push the training error?

- *We can make the model arbitrarily complex (effectively "memorizing" the entire training set).*

A: Down to zero!

Q: Why should we use training & test sets?

Thought experiment:

Suppose instead, we train our model using the entire dataset.

Q: How low can we push the training error?

- *We can make the model arbitrarily complex (effectively "memorizing" the entire training set).*

A: Down to zero!

NOTE

This phenomenon is called *overfitting*.

OVERFITTING

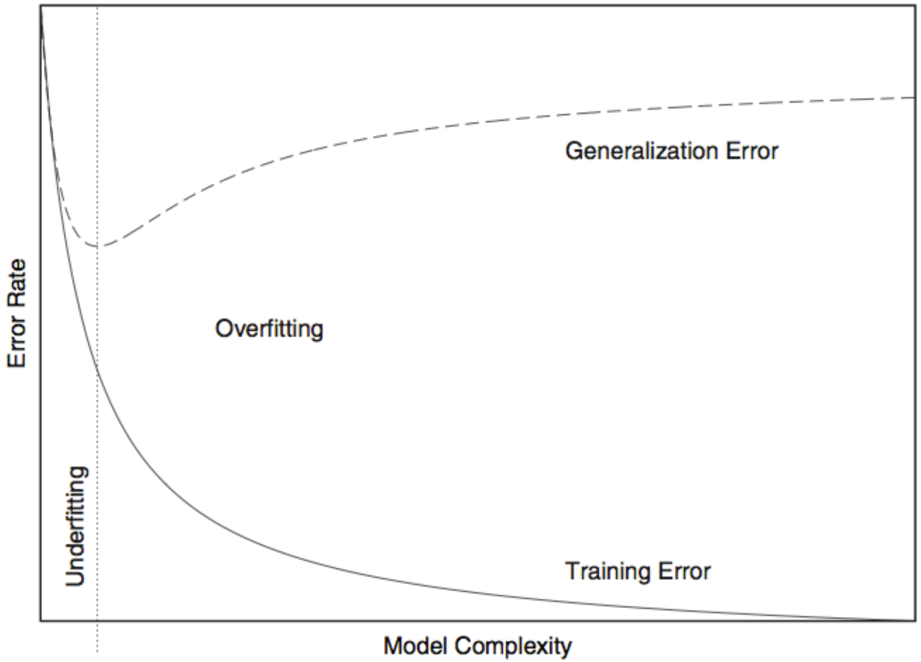
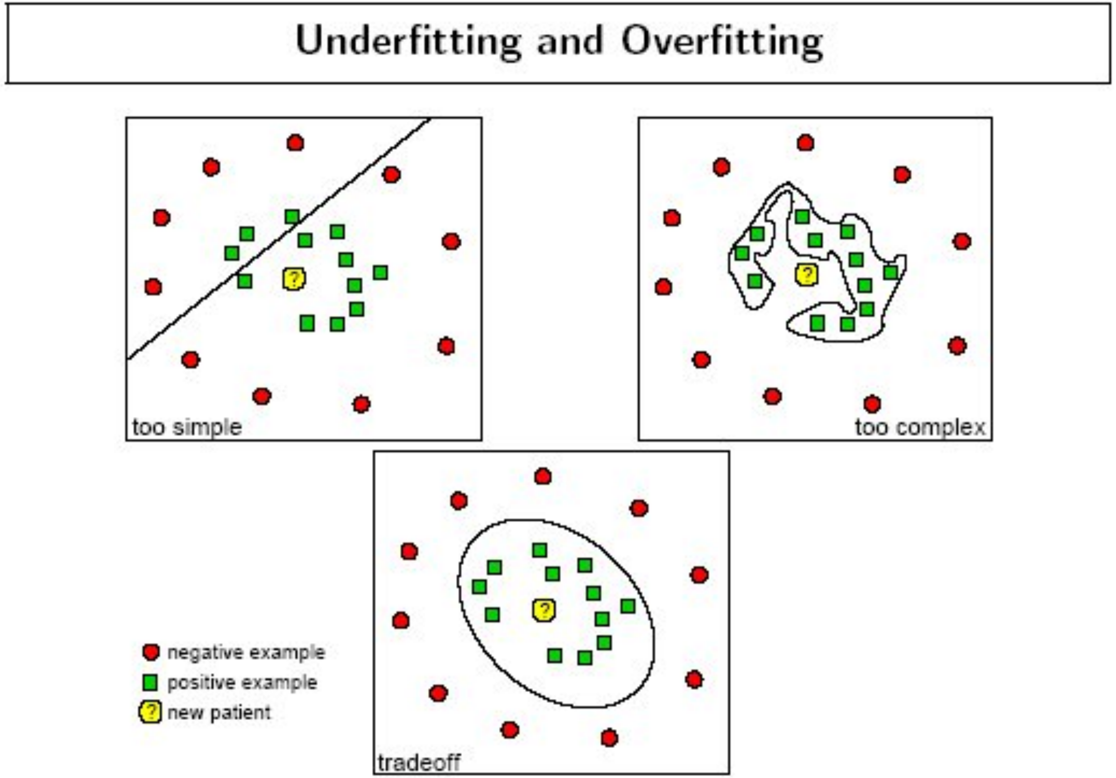


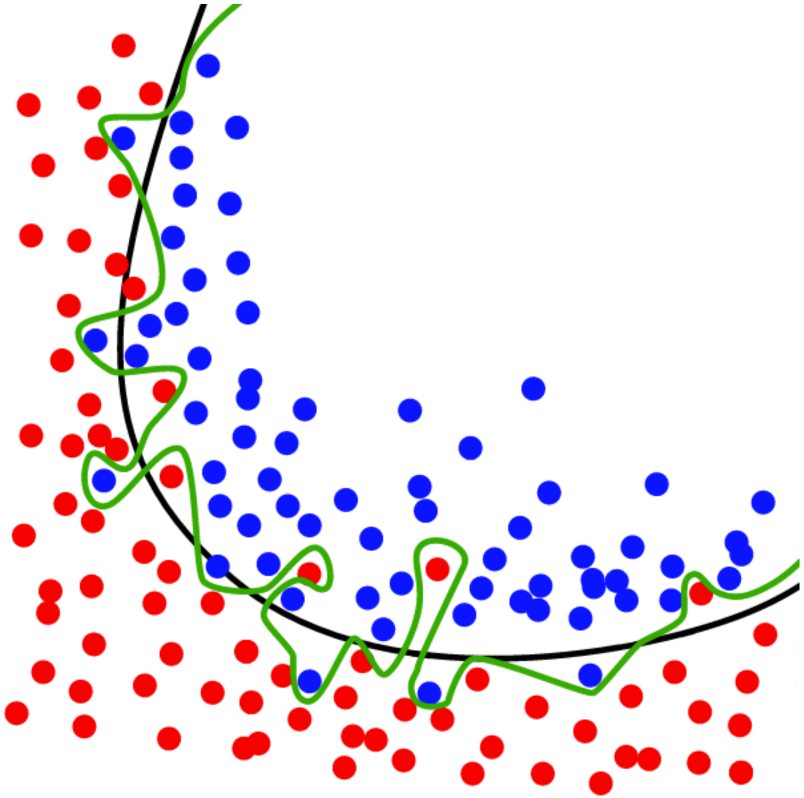
FIGURE 18-1. Overfitting: as a model becomes more complex, it becomes increasingly able to represent the training data. However, such a model is overfitted and will not generalize well to data that was not used during training.

source: Data Analysis with Open Source Tools, by Philipp K. Janert. O'Reilly Media, 2011.

OVERFITTING - EXAMPLE



source: http://www.dtreg.com



source: http://www.dtreg.com

Q: Why should we use training & test sets?

Thought experiment:

Suppose instead, we train our model using the entire dataset.

Q: How low can we push the training error?

- *We can make the model arbitrarily complex (effectively "memorizing" the entire training set).*

A: Down to zero!

NOTE

This phenomenon is called *overfitting*.

A: Training error is not a good estimate of OOS accuracy.

Suppose we do the train/test split.

Suppose we do the train/test split.

Q: How well does generalization error predict OOS accuracy?

Suppose we do the train/test split.

Q: How well does generalization error predict OOS accuracy?

Thought experiment:

Suppose we had done a different train/test split.

Suppose we do the train/test split.

Q: How well does generalization error predict OOS accuracy?

Thought experiment:

Suppose we had done a different train/test split.

Q: Would the generalization error remain the same?

Suppose we do the train/test split.

Q: How well does generalization error predict OOS accuracy?

Thought experiment:

Suppose we had done a different train/test split.

Q: Would the generalization error remain the same?

A: Of course not!

Suppose we do the train/test split.

Q: How well does generalization error predict OOS accuracy?

Thought experiment:

Suppose we had done a different train/test split.

Q: Would the generalization error remain the same?

A: Of course not!

A: On its own, not very well.

Something is still missing!

Something is still missing!

Q: How can we do better?

Something is still missing!

Q: How can we do better?

Thought experiment:

Different train/test splits will give us different generalization errors.

Something is still missing!

Q: How can we do better?

Thought experiment:

Different train/test splits will give us different generalization errors.

Q: What if we did a bunch of these and took the average?

Something is still missing!

Q: How can we do better?

Thought experiment:

Different train/test splits will give us different generalization errors.

Q: What if we did a bunch of these and took the average?

A: Now you're talking!

Something is still missing!

Q: How can we do better?

Thought experiment:

Different train/test splits will give us different generalization errors.

Q: What if we did a bunch of these and took the average?

A: Now you're talking!

A: Cross-validation.

Steps for n-fold cross-validation:

Steps for n-fold cross-validation:

1) Randomly split the dataset into n equal partitions.

Steps for n-fold cross-validation:

1) Randomly split the dataset into n equal partitions.
2) Use partition 1 as test set & union of other partitions as training set.

Steps for n-fold cross-validation:

1) Randomly split the dataset into n equal partitions.
2) Use partition 1 as test set & union of other partitions as training set.
3) Find generalization error.

Steps for n-fold cross-validation:

1) Randomly split the dataset into n equal partitions.
2) Use partition 1 as test set & union of other partitions as training set.
3) Find generalization error.
4) Repeat steps 2-3 using a different partition as the test set at each iteration.

Steps for n-fold cross-validation:

1) Randomly split the dataset into n equal partitions.
2) Use partition 1 as test set & union of other partitions as training set.
3) Find generalization error.
4) Repeat steps 2-3 using a different partition as the test set at each iteration.
5) Take the average generalization error as the estimate of OOS accuracy.

Features of n-fold cross-validation:

Features of n-fold cross-validation:

1) More accurate estimate of OOS prediction error.

Features of n-fold cross-validation:

1) More accurate estimate of OOS prediction error.
2) More efficient use of data than single train/test split.
   - Each record in our dataset is used for both training and testing.

Features of n-fold cross-validation:

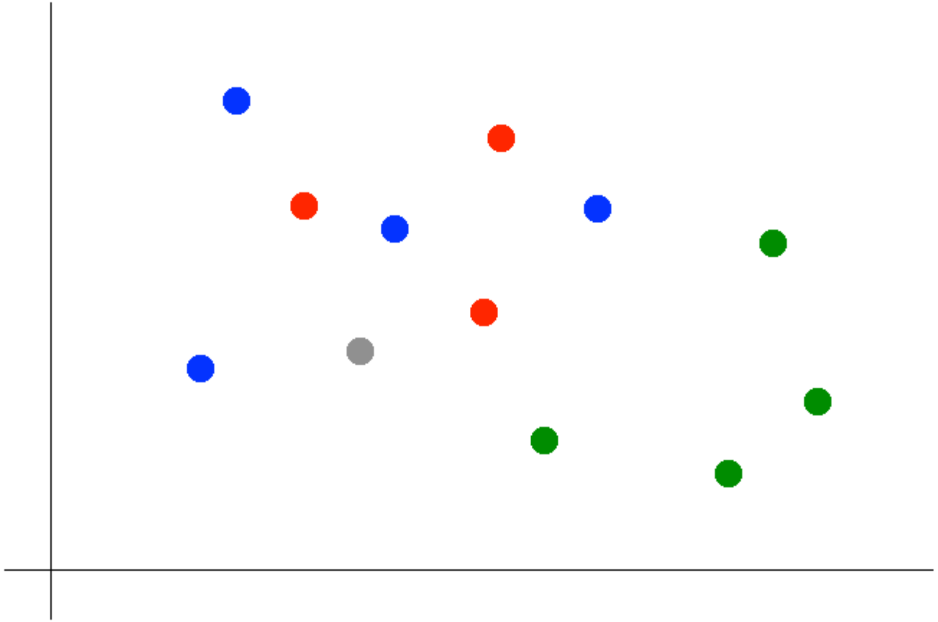
1) More accurate estimate of OOS prediction error.
2) More efficient use of data than single train/test split.
   - Each record in our dataset is used for both training and testing.
3) Presents tradeoff between efficiency and computational expense.
   - 10-fold CV is 10x more expensive than a single train/test split

Features of n-fold cross-validation:

1) More accurate estimate of OOS prediction error.
2) More efficient use of data than single train/test split.
    - Each record in our dataset is used for both training and testing.
3) Presents tradeoff between efficiency and computational expense.
    - 10-fold CV is 10x more expensive than a single train/test split
4) Can be used for model selection.

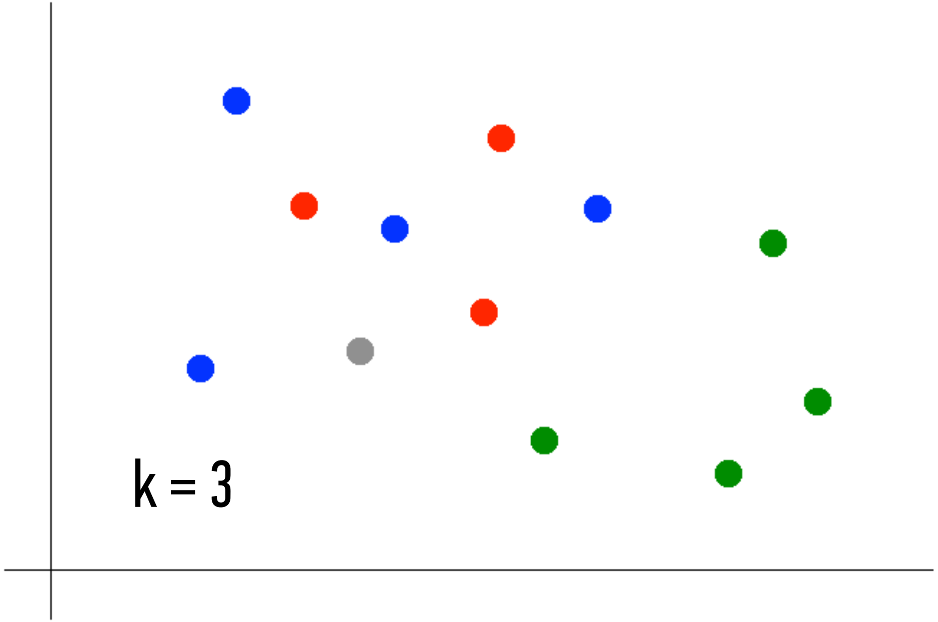
III. KNN CLASSIFICATION

Suppose we want to predict the color of the grey dot.



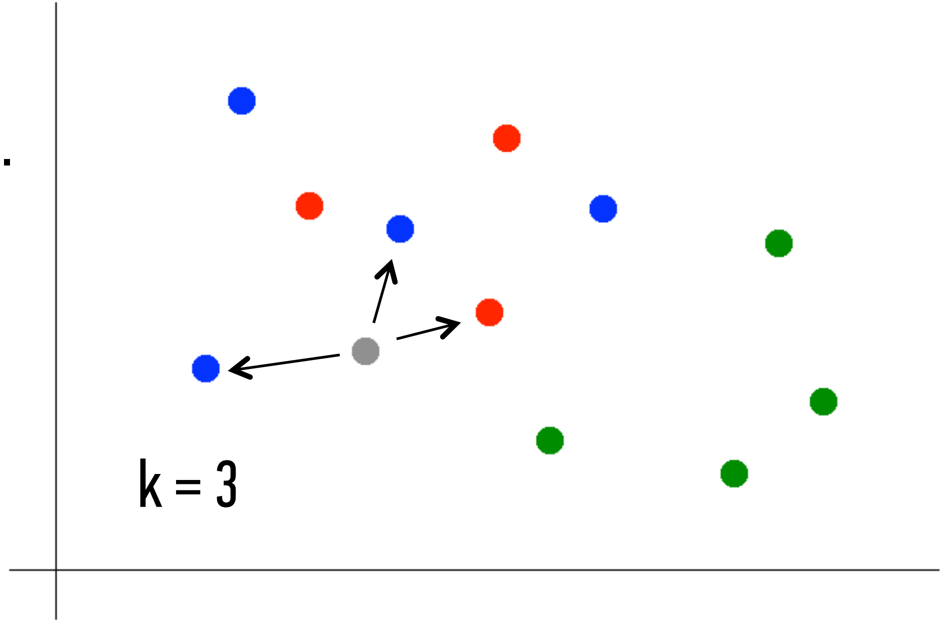
Suppose we want to predict the color of the grey dot.

1) Pick a value for k.



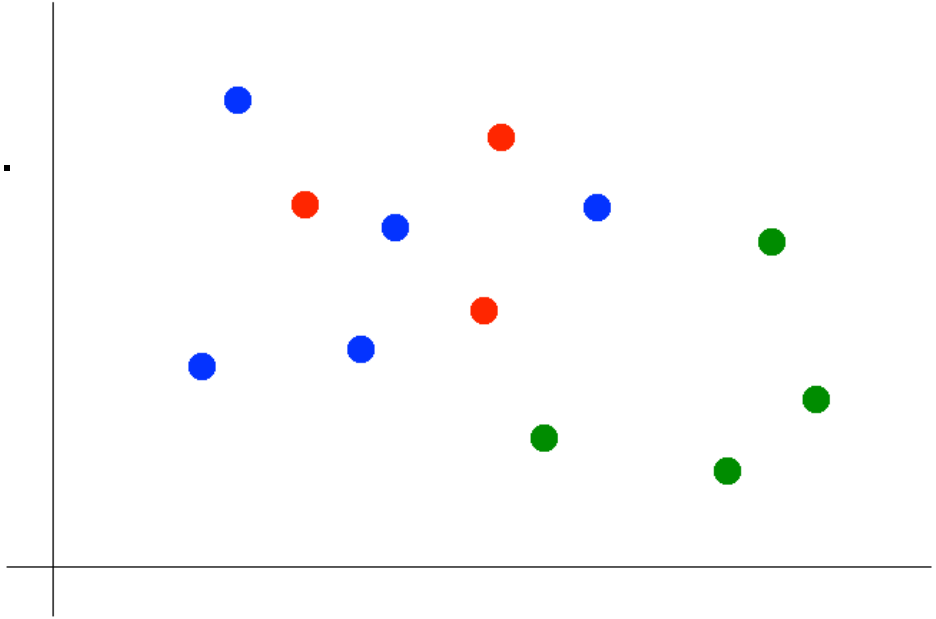
Suppose we want to predict the color of the grey dot.

1) Pick a value for k.
2) Find colors of k nearest neighbors.



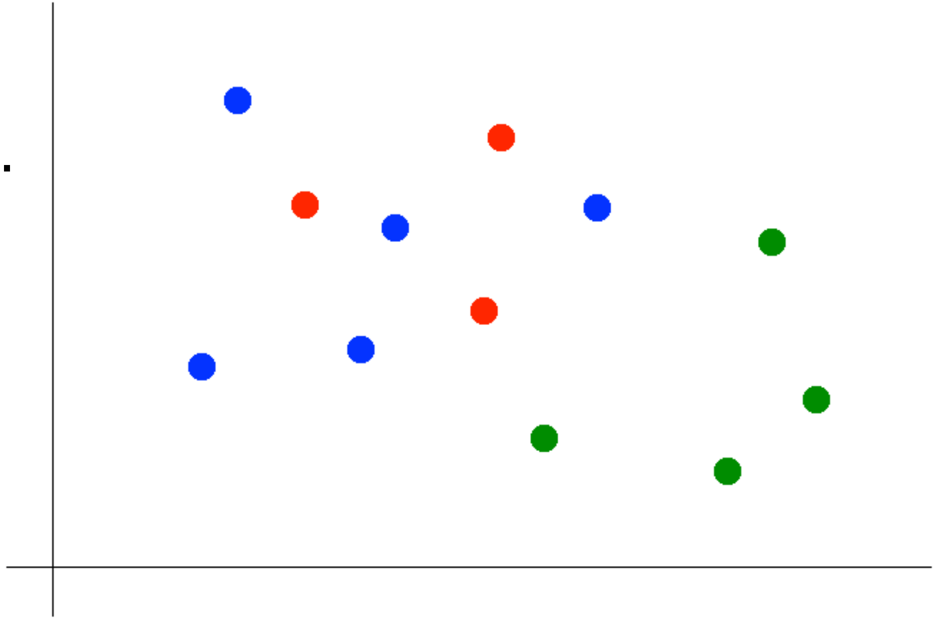
Suppose we want to predict the color of the grey dot.

1) Pick a value for k.
2) Find colors of k nearest neighbors.
3) Assign the most common color to the grey dot.



Suppose we want to predict the color of the grey dot.

1) Pick a value for k.
2) Find colors of k nearest neighbors.
3) Assign the most common color to the grey dot.



OPTIONAL NOTE

Our definition of "nearest" implicitly uses the *Euclidean distance function.*

IV. LINEAR REGRESSION

	continuous	*categorical*
supervised	???	???
unsupervised	???	???

	continuous	*categorical*
supervised	regression	classification
unsupervised	dimension reduction	clustering

Q: What is a **regression** model?

Q: What is a **regression** model?

A: A functional relationship between input & response variables.

Q: What is a **regression** model?

A: A functional relationship between input & response variables

The **simple linear regression** model captures a linear relationship between a single input variable x and a response variable y:

Q: What is a **regression** model?
A: A functional relationship between input & response variables

The **simple linear regression** model captures a linear relationship between a single input variable x and a response variable y:

$$y = \alpha + \beta x + \varepsilon$$

Q: What do the terms in this model mean?

$$y = \alpha + \beta x + \varepsilon$$

Q: What do the terms in this model mean?

$$y = \alpha + \beta x + \varepsilon$$

A: y = **response variable** (the one we want to predict)

Q: What do the terms in this model mean?

$$y = \alpha + \beta x + \varepsilon$$

A: y = **response variable** (the one we want to predict)

x = **input variable** (the one we use to train the model)

Q: What do the terms in this model mean?

$$y = \alpha + \beta x + \varepsilon$$

A: y = **response variable** (the one we want to predict)

x = **input variable** (the one we use to train the model)

α = **intercept** (where the line crosses the y-axis)

Q: What do the terms in this model mean?

$$y = \alpha + \beta x + \varepsilon$$

A: y = **response variable** (the one we want to predict)

x = **input variable** (the one we use to train the model)

α = **intercept** (where the line crosses the y-axis)

β = **regression coefficient** (the model "parameter")

Q: What do the terms in this model mean?

$$y = \alpha + \beta x + \varepsilon$$

A: y = **response variable** (the one we want to predict)

x = **input variable** (the one we use to train the model)

α = **intercept** (where the line crosses the y-axis)

β = **regression coefficient** (the model "parameter")

ε = **residual** (the prediction error)

We can extend this model to several input variables, giving us the **multiple linear regression** model:

We can extend this model to several input variables, giving us the **multiple linear regression** model:

$$y = \alpha + \beta_1 x_1 \; + \; \dots \; + \; \beta_n x_n + \varepsilon$$

Linear regression involves several technical assumptions and is often presented with lots of mathematical formality.

The math is not very important for our purposes, but you should check it out if you get serious about solving regression problems.

Q: How do we fit a regression model to a dataset?

Q: How do we fit a regression model to a dataset?

A: In theory, minimize the sum of the squared residuals (OLS).

Q: How do we fit a regression model to a dataset?

A: In theory, minimize the sum of the squared residuals (OLS).

In practice, any respectable piece of software will do this for you.

Q: How do we fit a regression model to a dataset?
A: In theory, minimize the sum of the squared residuals (OLS).

In practice, any respectable piece of software will do this for you.

But again, if you get serious about regression, you should learn how this works!

V. POLYNOMIAL REGRESSION

Consider the following **polynomial regression** model:

$$y = \alpha + \beta_1 x + \beta_2 x^2 + \varepsilon$$

Consider the following **polynomial regression** model:

$$y = \alpha + \beta_1 x + \beta_2 x^2 + \varepsilon$$

Q: This represents a nonlinear relationship. Is it still a linear model?

Consider the following **polynomial regression** model:

$$y = \alpha + \beta_1 x + \beta_2 x^2 + \varepsilon$$

Q: This represents a nonlinear relationship. Is it still a linear model?

A: Yes, because it's linear in the β's!

Polynomial regression allows us to fit very complex curves to data.

$$y = \alpha + \beta_1 x + \beta_2 x^2 + \ldots + \beta_n x^n + \varepsilon$$

Polynomial regression allows us to fit very complex curves to data.

$$y = \alpha + \beta_1 x + \beta_2 x^2 + \ldots + \beta_n x^n + \varepsilon$$

But there is one problem with the model we've written down so far.

Polynomial regression allows us to fit very complex curves to data.

$$y = \alpha + \beta_1 x + \beta_2 x^2 + \ldots + \beta_n x^n + \varepsilon$$

But there is one problem with the model we've written down so far.

Q: Does anyone know what it is?

Polynomial regression allows us to fit very complex curves to data.

$$y = \alpha + \beta_1 x + \beta_2 x^2 + \dots + \beta_n x^n + \varepsilon$$

But there is one problem with the model we've written down so far.

Q: Does anyone know what it is?

A: This model violates one of the assumptions of linear regression!

This model displays **multicollinearity**, which means the predictor variables are highly correlated with each other.

$$y = \alpha + \beta_1 x + \beta_2 x^2 + \dots + \beta_n x^n + \varepsilon$$

```
> x <- seq(1, 10, 0.1)
> cor(x^9, x^10)
[1] 0.9987608
```

This model displays **multicollinearity**, which means the predictor variables are highly correlated with each other.

$$y = \alpha + \beta_1 x + \beta_2 x^2 + \dots + \beta_n x^n + \varepsilon$$

Multicollinearity causes the linear regression model to break down, because it can’t tell the predictor variables apart.

Q: What can we do about this?

Q: What can we do about this?

A: Replace the correlated predictors with uncorrelated predictors.

Q: What can we do about this?

A: Replace the correlated predictors with uncorrelated predictors.

$$y = \alpha + \beta_1 f_1(x) + \beta_2 f_2(x^2) + \ldots + \beta_n f_n(x^n) + \varepsilon$$

So far, we've seen how polynomial regression allows us to fit complex nonlinear relationships, and even to avoid multicollinearity (by using basis functions).

So far, we've seen how polynomial regression allows us to fit complex nonlinear relationships, and even to avoid multicollinearity (by using basis functions).

Q: Can a regression model be too complex?

V. REGULARIZATION

Recall our earlier discussion of **overfitting**.

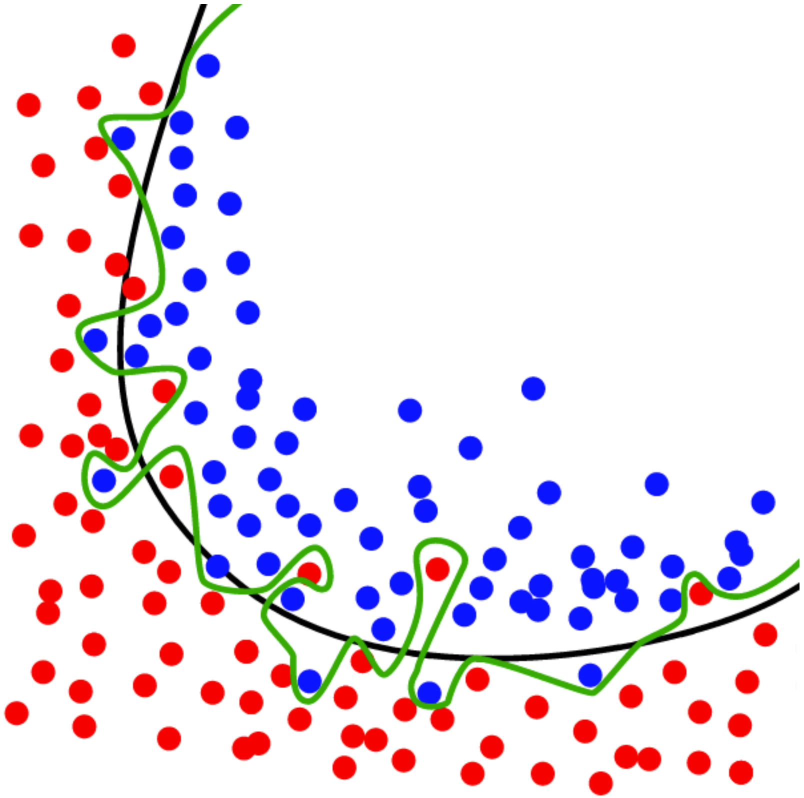
Recall our earlier discussion of **overfitting**.

When we talked about this in the context of classification, we said that it was a result of matching the training set too closely.

Recall our earlier discussion of **overfitting**.

When we talked about this in the context of classification, we said that it was a result of matching the training set too closely.

In other words, an overfit model matches the **noise** in the dataset instead of the **signal**.



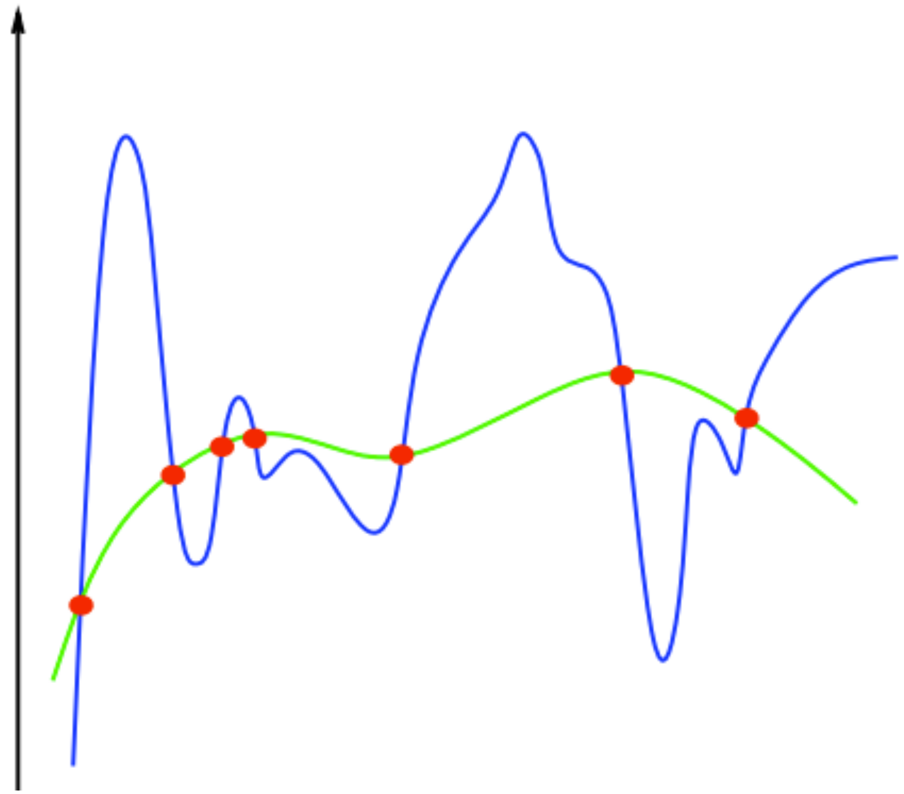
source: *http://upload.wikimedia.org/wikipedia/commons/1/19/Overfitting.svg*

The same thing can happen in regression.

It's possible to design a regression model that matches the noise in the data instead of the signal.

This happens when our model becomes *too complex* for the data to support.

OVERFITTING EXAMPLE (REGRESSION)



source: http://www.mit.edu/~9.520/spring12/slides/class02/class02.pdf

Q: How do we define the **complexity** of a regression model?

Q: How do we define the **complexity** of a regression model?

A: One method is to define complexity as a function of the size of the coefficients.

Q: How do we define the **complexity** of a regression model?

A: One method is to define complexity as a function of the size of the coefficients.

Ex 1: $\Sigma\,|\beta_i|$

Ex 2: $\Sigma\,\beta_i{}^2$

Q: How do we define the **complexity** of a regression model?

A: One method is to define complexity as a function of the size of the coefficients.

Ex 1: $\Sigma\,|\beta_i|$ this is called the **L1-norm**

Ex 2: $\Sigma\,{\beta_i}^2$ this is called the **L2-norm**

These measures of complexity lead to the following **regularization** techniques:

These measures of complexity lead to the following **regularization** techniques:

L1 regularization: $y = \Sigma\, \beta_i x_i + \varepsilon \quad st. \quad \Sigma\, |\beta_i| < s$

These measures of complexity lead to the following **regularization** techniques:

L1 regularization: $y = \Sigma\, \beta_i x_i + \varepsilon \quad st. \quad \Sigma\, |\beta_i| < s$

L2 regularization: $y = \Sigma\, \beta_i x_i + \varepsilon \quad st. \quad \Sigma\, \beta_i^2 < s$

These measures of complexity lead to the following **regularization** techniques:

L1 regularization: $y = \Sigma\, \beta_i x_i + \varepsilon \quad st. \quad \Sigma\, |\beta_i| < s$

L2 regularization: $y = \Sigma\, \beta_i x_i + \varepsilon \quad st. \quad \Sigma\, \beta_i^2 < s$

Regularization refers to the method of preventing **overfitting** by explicitly controlling model **complexity**.

These measures of complexity lead to the following **regularization** techniques:

***Lasso* regularization**: $y = \Sigma\, \beta_i x_i + \varepsilon \quad st. \quad \Sigma\, |\beta_i| < s$

***Ridge* regularization**: $y = \Sigma\, \beta_i x_i + \varepsilon \quad st. \quad \Sigma\, \beta_i^2 < s$

Regularization refers to the method of preventing **overfitting** by explicitly controlling model **complexity**.

These regularization problems can also be expressed as:

L1 regularization: $min(\|y - x\beta\|^2 + \lambda\|x\|)$

L2 regularization: $min(\|y - x\beta\|^2 + \lambda\|x\|^2)$

These regularization problems can also be expressed as:

L1 regularization: $min(\|y - x\beta\|^2 + \lambda\|x\|)$

L2 regularization: $min(\|y - x\beta\|^2 + \lambda\|x\|^2)$

We are no longer just minimizing error but also an additional term.

DISCUSSION