Data Science for Business

MGT 432

Prof. Kenneth Younge, Ph.D.

Associate Professor

Chair of Technology and Innovation Strategy

Session 2: Data & Text

Review: Setting up for the Course

- Did you…
  - send us your GitHub username? Do you have access?
  - clone the class repo?
  - install Anaconda?
  - run a Jupyter notebook?
  - complete the python data camp/tutorials?
- Other questions / problems before we get started?

Review: Big Picture

- This is a course about ***predictive*** analytics

- We cover methods not applications

    - By <u>methods</u>, we do not mean optimization

        We will learn how to build and then evaluate
        the performance of different kinds of predictive **models**

    - By <u>applications</u>, we do **not** mean we avoid "applied work"

        We will work on many applied examples and a project.

        But we will not learn specific solutions for specific problems
        (e.g., customer retention, credit analysis, demand forecasting, etc.)

Review: How to learn the material

- Instructor-directed learning
  - Lectures
  - Examples
  - Demos
- Self-directed learning
  - Readings
  - Assignments
  - Project
  - Presentation

Today’s Agenda

- Assigned Reading
  - DMCT chapter 3
  - DSFB chapter 10
- Lecture
  - Core concepts
  - Data preprocessing
  - Text as data
- Examples
- Assignment 1

```
DSFB:  Data Science for Business
ISL:   Introduction to Statistical Learning: with Applications in R
DMCT:  Data Mining: Concepts and Techniques
DPDM:  Data Preprocessing in Data Mining
ELA:   Evaluation Learning Algorithms: A Classification Perspective
DL:    Deep Learning
```

Lecture: Core Concepts

Today we learn the building blocks of a **new language**

Core Concepts

- Measurement
- Sampling
- Data Preprocessing
- Description
- Modeling
- Validation
- Evaluation

Error

The objective is to **minimize error**

Measurement

- Many types of measurement
  - quantitative vs. qualitative
    - we'll deal with quantitative data, but qualitative data can also be encoded
  - structured vs. unstructured
    - structured data fits into known dimensions or a known taxonomy, and often is already tabularized (in tables)
  - observed (or manifest) vs. latent
    - manifest data can be directly observed (e.g., height) latent constructs cannot (e.g., social status)
  - direct proxy vs. composite proxy
    - we may not have direct data for either manifest or latent factors — often we use ***proxies***
  - metric "cardinal" vs. non-metric "categorical"
    - metric variables exhibit distance along a scale non-metric exhibit categorical states
  - non-metric: binary vs. nominal vs. ordinal
    - binary states (0, 1); nominal categories (zipcodes); ordinal rankings (1st, 2nd, 3rd, …)
  - metric values: discrete vs. continuous
    - discrete is restricted to defined values (e.g., countable) continuous is not limited to certain defined values
  - interval-scaled vs. ratio-scaled
    - interval-scaled variables have units of equal size ratio-scaled variables have an inherent zero-point (and thus one value is a ***multiple*** of another value)
  - raw vs. transformed
    - data can be ***standardized*** or ***normalized*** for more precise or more interpretable computation

Measurement

- Data Science crosses many domains,
so how we reference the data can get confusing

<u>Left Hand Side</u>

- Y
- LHS
- Variable
- Target variable
- Outcome
- Response
- Dependent variable (DV)

<u>Right Hand Side</u>

- X
- RHS
- Variable
- Predictor
- Independent variable
- Explanatory variable
- Covariate (certain types)
- Control (certain types)
- Attribute
- Feature

Machine Learning

Sampling

- Population vs. Sample
  - **Sampling variation** (i.e., sample uncertainty) implies that a sample only provides an estimate of a true population
  - Statisticians worry about the “statistical power” of their analysis (i.e., likelihood that the sample provides a good estimate of the population)
  - But in data science, we often have an abundance of data and/or we may even have the entire population of interest!
- Sample selection
  - The way a sample comes about may lead to ***selection bias***
  - For example, there may be ***self-selection*** into or out-of the sample
  - Or there may be ***framing variation*** in how different sub-samples are defined, drawn from the population, or captured as values

Data Preprocessing

- Why preprocess your data?

  “Garbage in, garbage out”

- Data ***pre***-processing is actually an iterative process

  Measurement

  Sampling

  Descriptive Statistics

Data Preprocessing: Integration

- Merge, join, append data into a ***schema***
  - ***Schema*** = requirements for a consistent hierarchy & organization of data
  - You need to preprocess data to make sure it conforms to the schema
  - You may need to remove redundant or duplicate data
  - Often there are problematic encodings for certain values or conditions e.g., ***top-coding*** of errors, missing values, etc. (99 years of education)
- Use SQL with relational databases or pandas with tables (CSV files, spreadsheets, etc.) to import and inspect data

Data Preprocessing: Cleaning

- Missing values - What to do?
  - It depends… there is no single correct answer…
  - Options are:
    - drop observation (row-wise deletion)
    - replace with sample mean
    - replace with a group mean (if there is one and it is reliable)

      a group may be known based on another variable or clustering
    - replace with a predicted value from another model
  - You can take a "try and see" approach for predictive modeling, but you need a statistically rigorous approach for inferential testing

Most stats packages default to this

Data Preprocessing: Cleaning

- Other problems
  - **Outliers**
    - scatter plot; pair plot; check min/max; check leverage (if you known how)
  - **Duplicate & redundant data**
    - duplicate data overweight model to predict that outcome
    - redundant data — can be derived from other data — check ***correlations***
  - **Special encodings**
    - some systems may “top-code” certain conditions (e.g., 99 or 9999)
    - variable may be a *continuous,* but encoding indicates a *nominal category*
- Use summary statistics and ***stratified*** random checks to make sure your data makes sense! (Question: Why a “stratified” check ???)

Data Preprocessing: Transformation

- Standardization
  - rescale a variable based on where it fits in some distribution
  - "*z-scoring*" is a frequently used standardization

$$z = \frac{x - \mu}{\sigma}$$

μ = mean
σ = standard deviation

  - converts a data point into the *number of standard deviations* the observation is above the mean value of the sample
  - results in new distribution with mean of zero and variance of 1

- Why standardize?
  - easier comparison between variables (& estimated coeffs)
  - help optimizers *converge* better and faster
  - some learning algorithms (e.g. SVM) expect variances of the *same order*

Data Preprocessing: Transformation

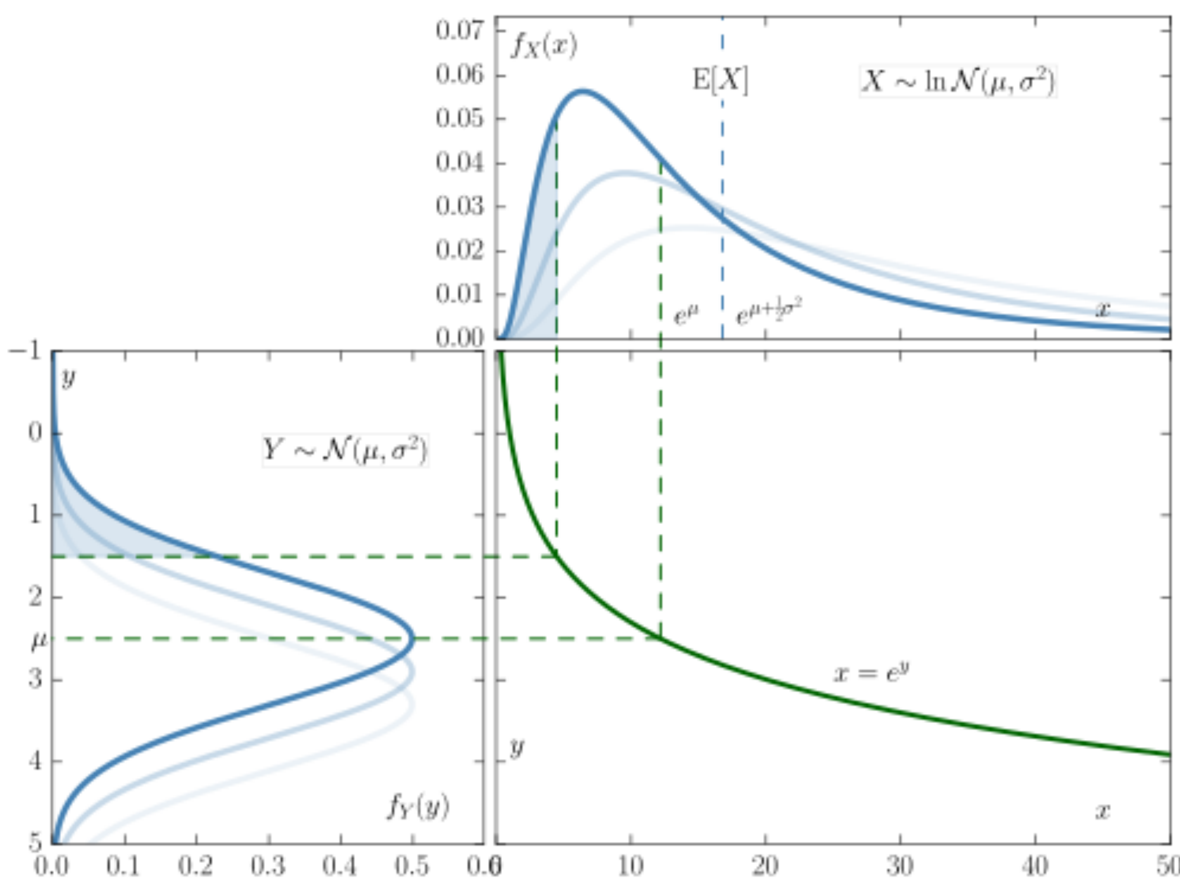
- Normalization
  - Rescale variable to fit an interval
    - Often the interval is [-1, 1] or [0, 1] (careful! sometimes also called “Standardization")
    - but one might also normalize to a unit norm (e.g., an L_1 or L_2 norm)
  - Examples:
    - min-max linear transformation to rescale variable to [0.0, 1.0]
    - Normalize a vector into a unit vector of length 1 (why? because dot product of two unit vectors is scalar equal to cosine of angle)
- Why normalize?
    - help optimizers converge better and faster
    - units of measurement can affect methods such as neural nets and nearest neighbors (smaller units leads to a larger range, which leads to more predictive weight for predictor)

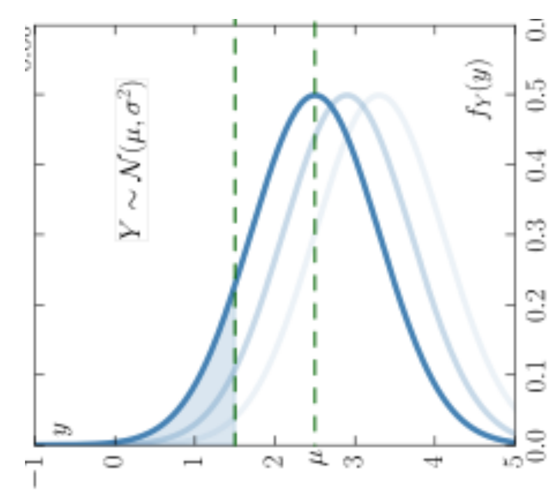
Data Preprocessing: Transformation

- Functional Transformations

  - **arcsin($\sqrt{x}$)** pull ratio variables s away from the ends of the interval

  - **ln(x)** pull in the right tail & make distribution more normal

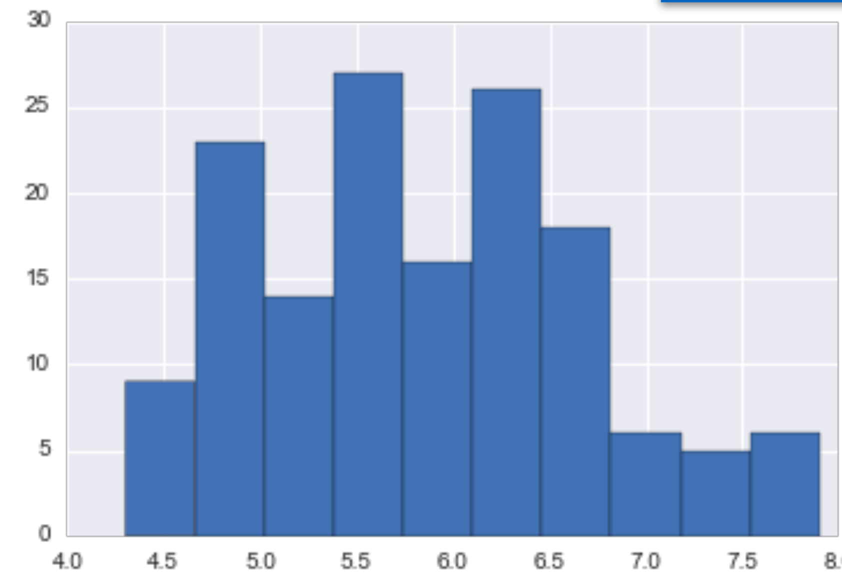




Source: By StijnDeVuyst [CC BY-SA 4.0] via Wikimedia Commons

Data Preprocessing: Transformation

- Binning
  - Split variable into discrete “bins” (e.g., a histogram does this)
  - Or split variable into nominal categories based on meaningful bucket ranges (e.g., years of schooling)

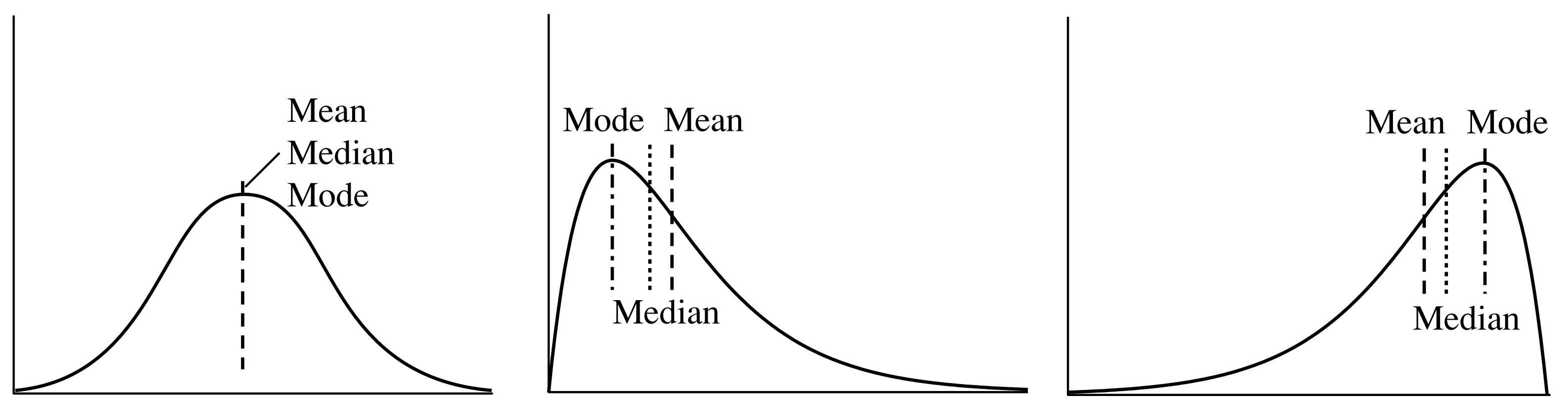


- Encoding categorical features
  - Why should we **not** assign categories a number? (e.g., a variable with 1,2,3,4,…)
    - Because that may introduce a false ordering or false interval spacing
    - E.g., If red=1, blue=2, green=3, it is not the case that green is two units “more” than red
  - “One Hot Encoding” is one way to convert categorical features into separate numerical “predictors”
    - Separate the variable into separate bins or categories, then create new 1/0 predictor variables for each category to predict each condition separately. For example, if there were 10 bins, there would be 10 indicators (“dummy variables equal to 0 or 1) to signify belonging to each bin

Description

- Basic Descriptive Statistics: central tendency

  - Mean
  - Median
  - Mode



(a) Symmetric data **(b)** Positively skewed data **(c)** Negatively skewed data

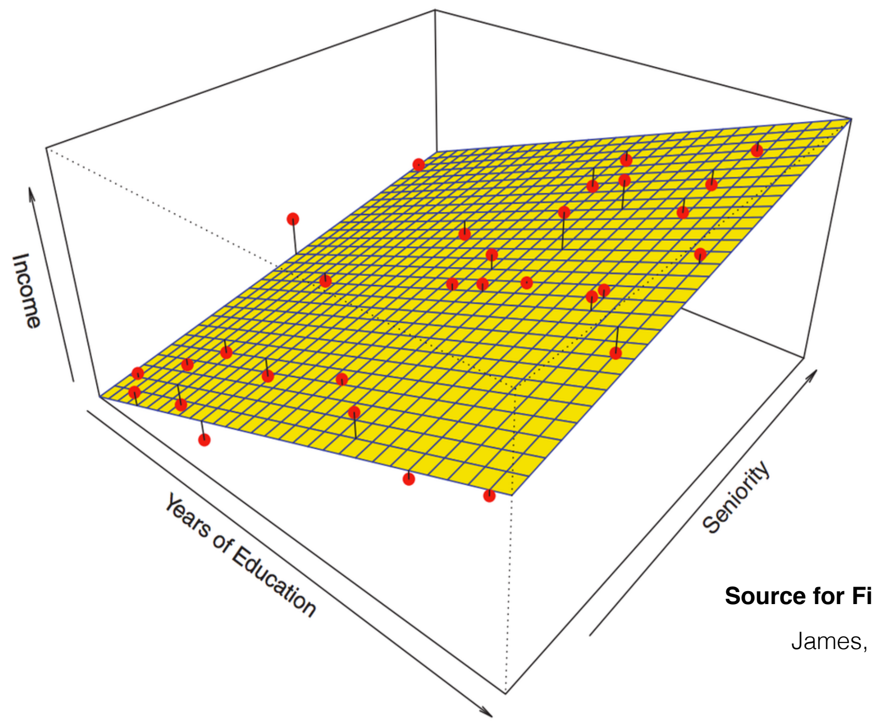
Source: Data Mining Concepts and Techniques, 2012

  - Dispersion
    - the *spread* of data away from the central tendency
    - often this can best be understood through a visual plot
    - numerically summarized by ***percentiles***, ***quartiles***, and/or a measure of ***variance***

  - Skew
    - the degree to which the dispersion away from the central tendency is *non-symmetric*

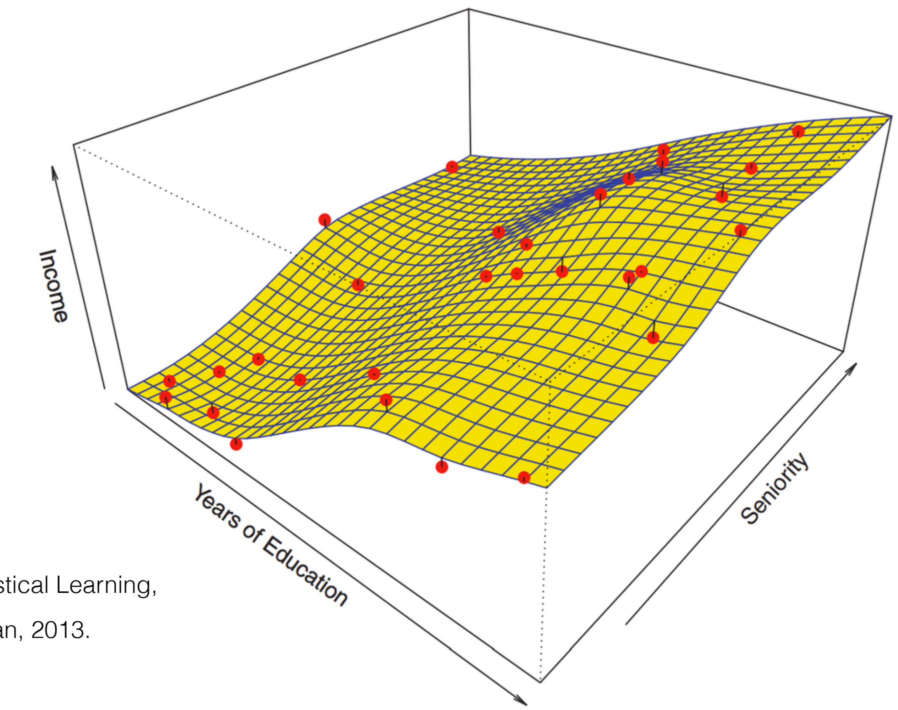
Modeling

- The course focuses on methods for ***predictive*** modeling



Less accurate, more interpretable

Source for Figures: Introduction to Statistical Learning, James, Witten, Hastie, and Tibshiran, 2013.



More accurate, less interpretable

- The quality of the data affects the quality of the prediction…

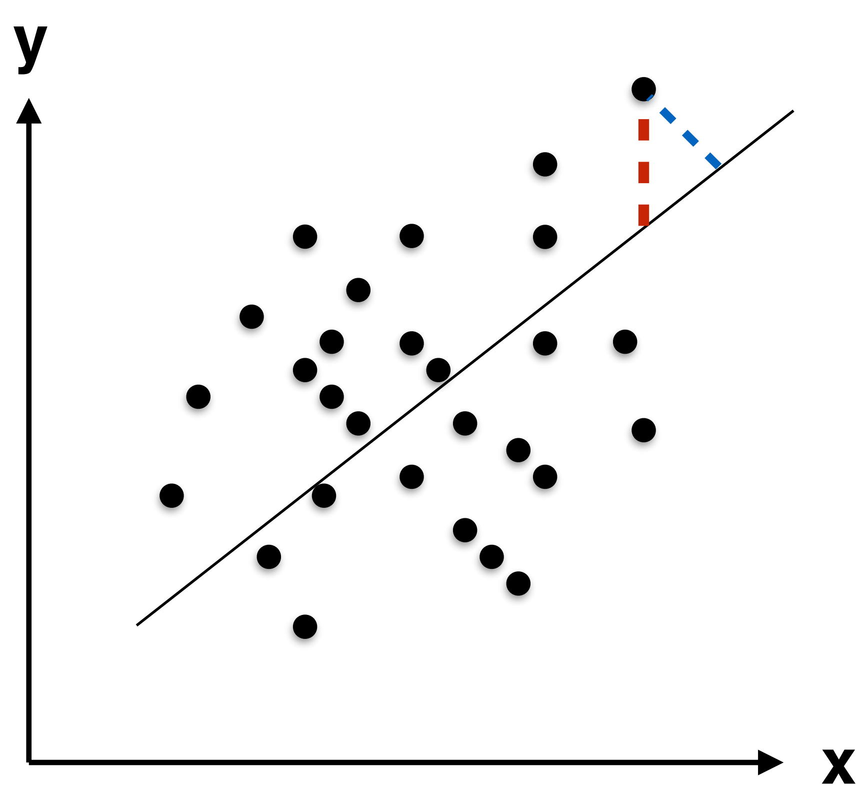
  … so we also spend time on measurement and preprocessing

Modeling: Supervision

- **Supervised** models - You have “labeled outcomes”
  - classification
  - regression
- **Unsupervised** models - you do NOT have labeled outcomes
  - clustering
  - dimensionality reduction
- In this course, we focus more on data-driven predictive modeling so much of the focus is on supervised models

Modeling: Prediction Error

- For continuous outcomes, "***error*** " is the ***distance*** between an observation and prediction by the model
  - Distance is defined as "the length of the space between two points."
  - But error can be measured along different *orientations* and/or scaled by different *transformations* (squared value, absolute value, …)



Euclidean "straight-line" distance along the orientation of the target variable

Euclidean "straight-line" distance in an orthogonal orientation to the prediction function

The most widely used error is *squared-error* in the orientation of the outcome $(\hat{y}_i - y_i)^2$

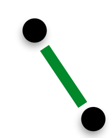
Modeling: Prediction Error

- For continuous outcomes, "***error*** " is the ***distance*** between an observation and prediction by the model

  - Distance is "the length of the space between two points," but error can be measured along different ***orientations*** and/or scaled by different ***transformations*** (by the square of the value, the absolute value, …)

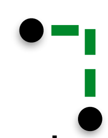
  - There are alternatives to Euclidean Distance:

    L_2 norm = Euclidean Distance

    L_1 norm = Manhattan Distance

    $$d(i, j) = \sqrt[h]{|x_{i1} - x_{j1}|^h + |x_{i2} - x_{j2}|^h + \cdots + |x_{ip} - x_{jp}|^h},$$

  - also non-metric "distances" that violate symmetry and triangle equality…

Modeling: Prediction Error

- For categorical outcomes, there is no "distance" between observation and prediction, so we use a "confusion matrix"

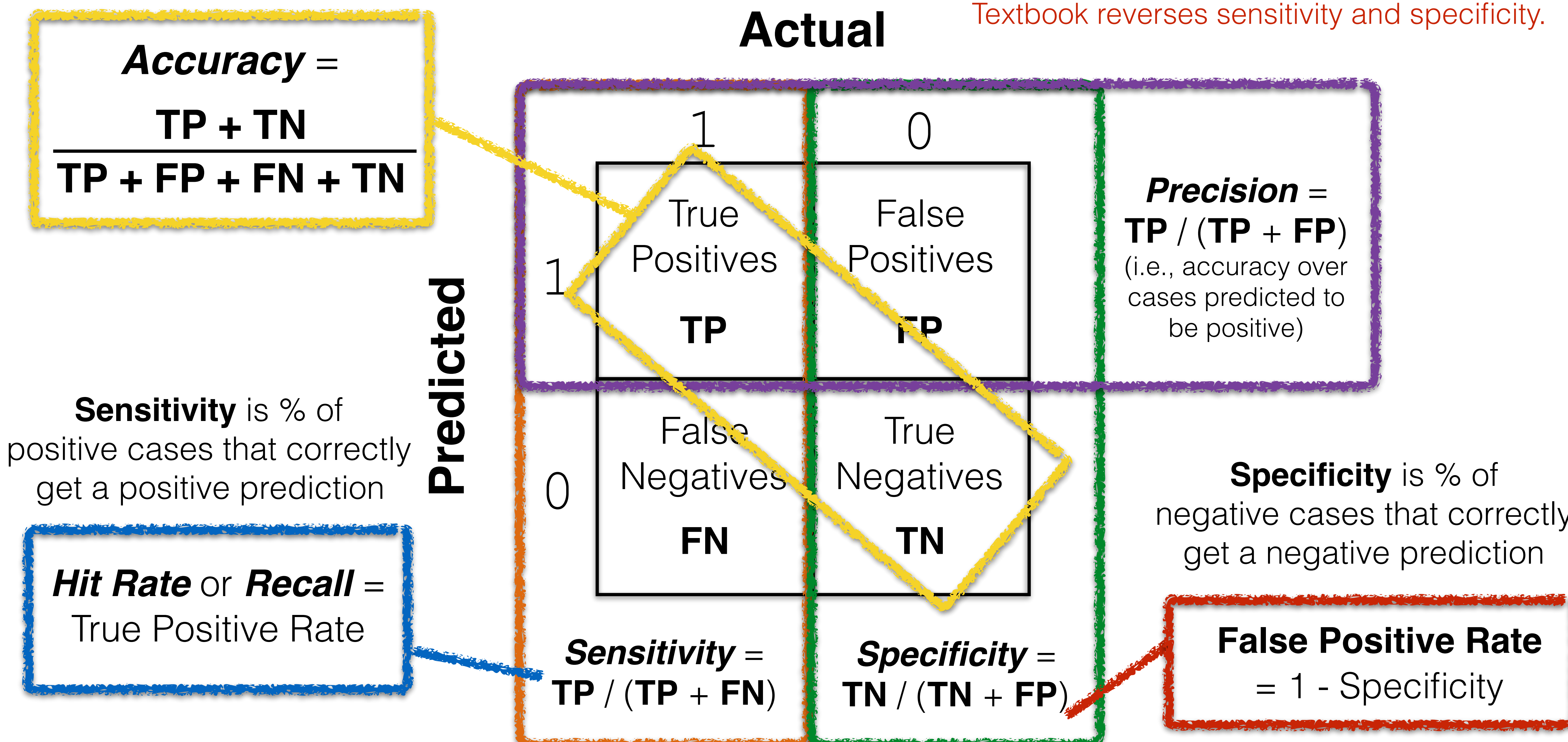
		Actual "Positive" 1	**Actual** "Negative" 0
Predicted	"Yes" 1	**Hits**	**False Alarms**
Predicted	"No" 0	False Negatives **FN**	True Negatives **TN**

For a binary outcome, the "positive" state is the one that happens or is predicted to happen. It has nothing to do with the *valence* of the state.

Modeling: Prediction Error

- For categorical outcomes, there is no “distance” between observation and prediction, so we use a “confusion matrix”

Warning: See errata for DSFB.
Textbook reverses sensitivity and specificity.



Modeling: Prediction Error

- For categorical outcomes, there is no "distance" between observation and prediction, so we use a "confusion matrix"

- Base-rates matter a lot when talking about the accuracy of a prediction…
- And so we will revisit this matrix and adapt it to meet business objectives …
- But for now, you should **know these well**.
- These rates will shape the optimal information acquisition strategies (more on that later in course)

Actual

Predicted	1	0
1	True Positives **TP**	False Positives **FP**
0	False Negatives **FN**	True Negatives **TN**

Precision = **TP** / (**TP** + **FP**)
(i.e., accuracy over cases predicted to be positive)

Sensitivity= **TP** / (**TP** + **FN**)

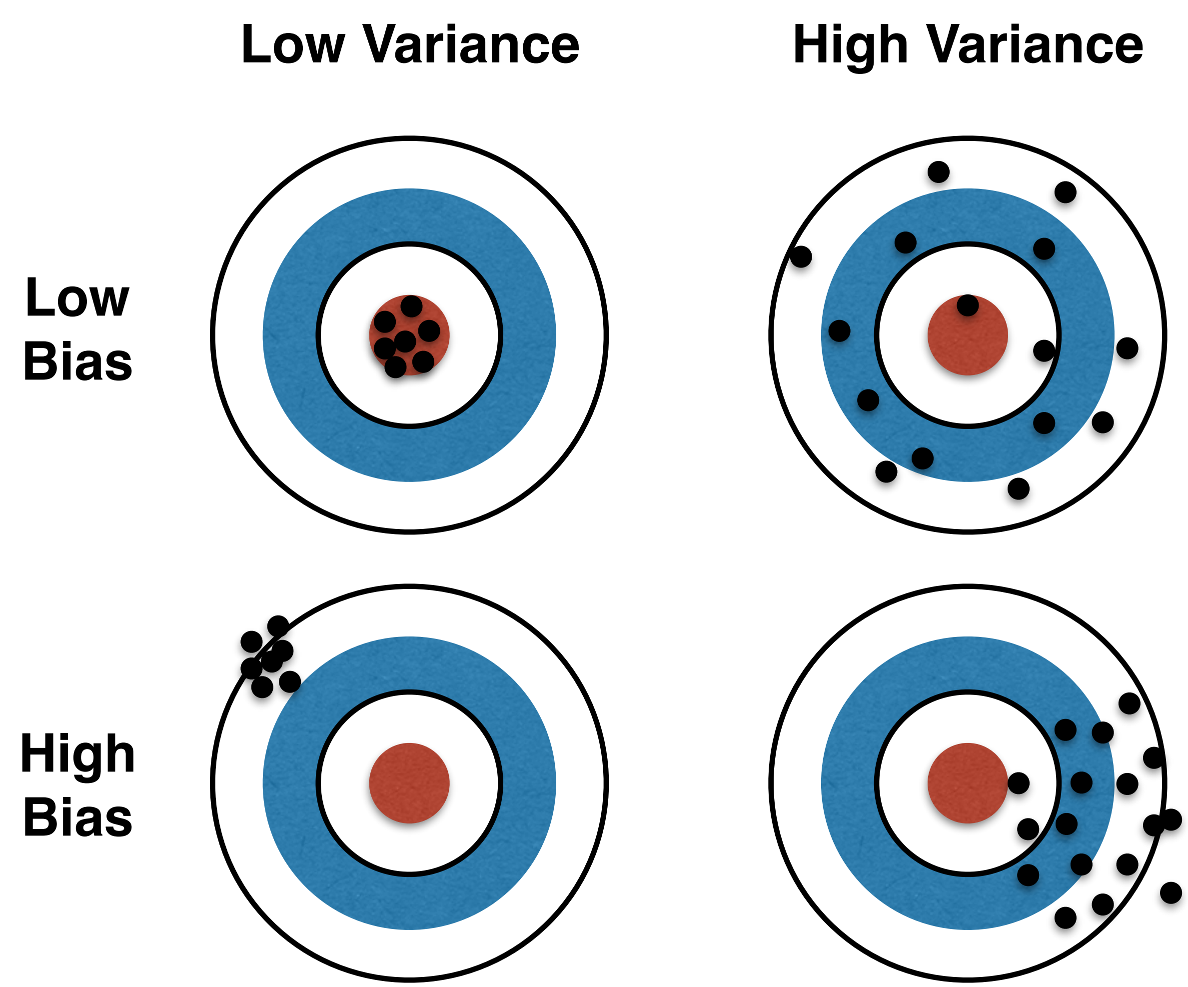
Specificity = **TN** / (**TN** + **FP**)

Modeling: Prediction Error

- Prediction error is composed of both ***bias***, ***variance***, *and* ***noise***
  - Error ***variance*** is the spread (squared deviation) of predictions around the mean prediction.
  - Error ***bias*** is a systematic difference between an average prediction and the *true* generating function.
  - Error ***noise*** is the irreducible (uncontrollable) disturbance that results from random disturbances, etc.

- Achieving low bias AND low variance is ideal, but there is a limit to how much systematic error (bias) you can reduce before you increase variance (and thus overall error) for future predictions.

- This is known as the "**bias-variance tradeoff**."

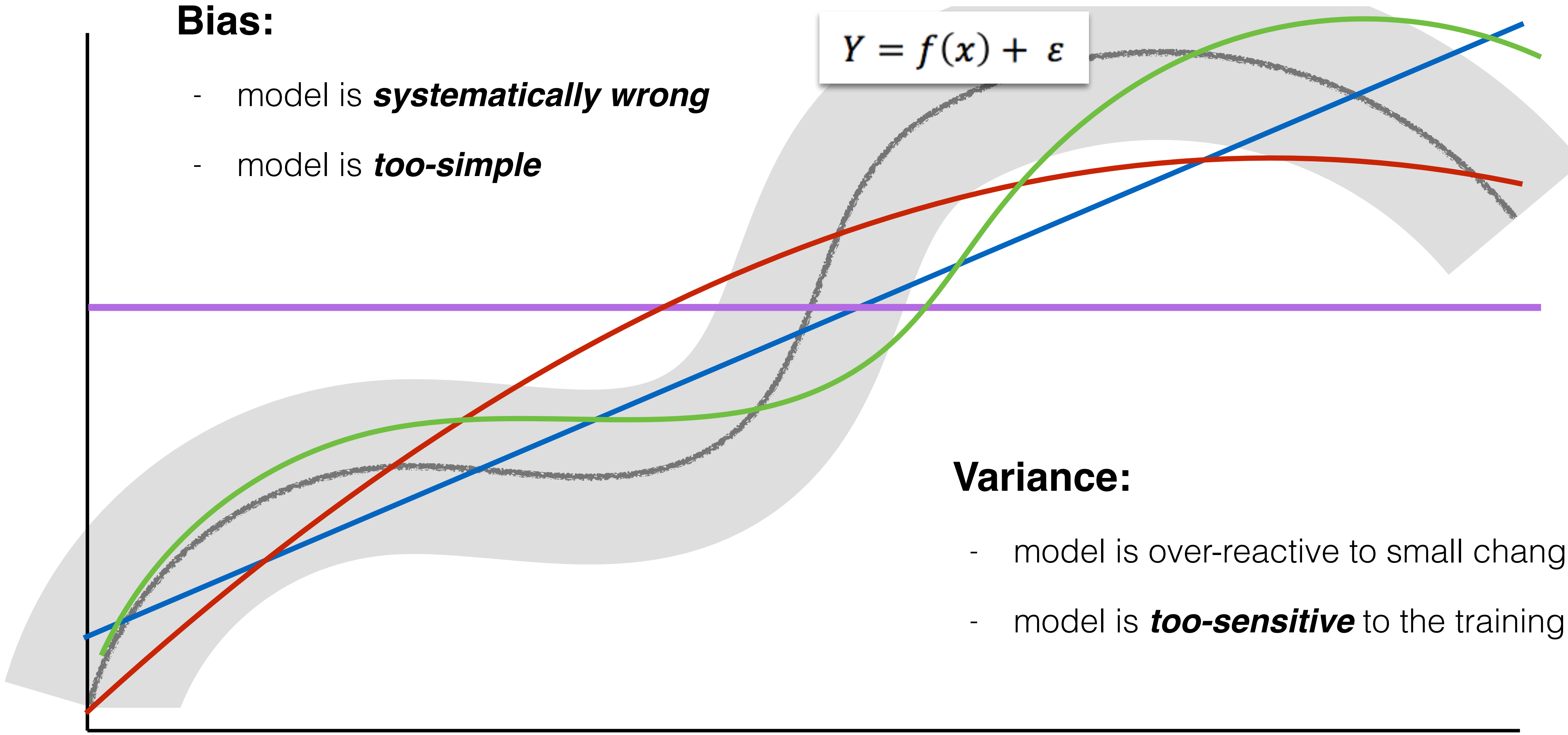


Modeling: Prediction Error

- Learning algorithms face a fundamental tradeoff between ***bias*** and ***variance*** in making predictions:

Bias:

- model is ***systematically wrong***
- model is ***too-simple***

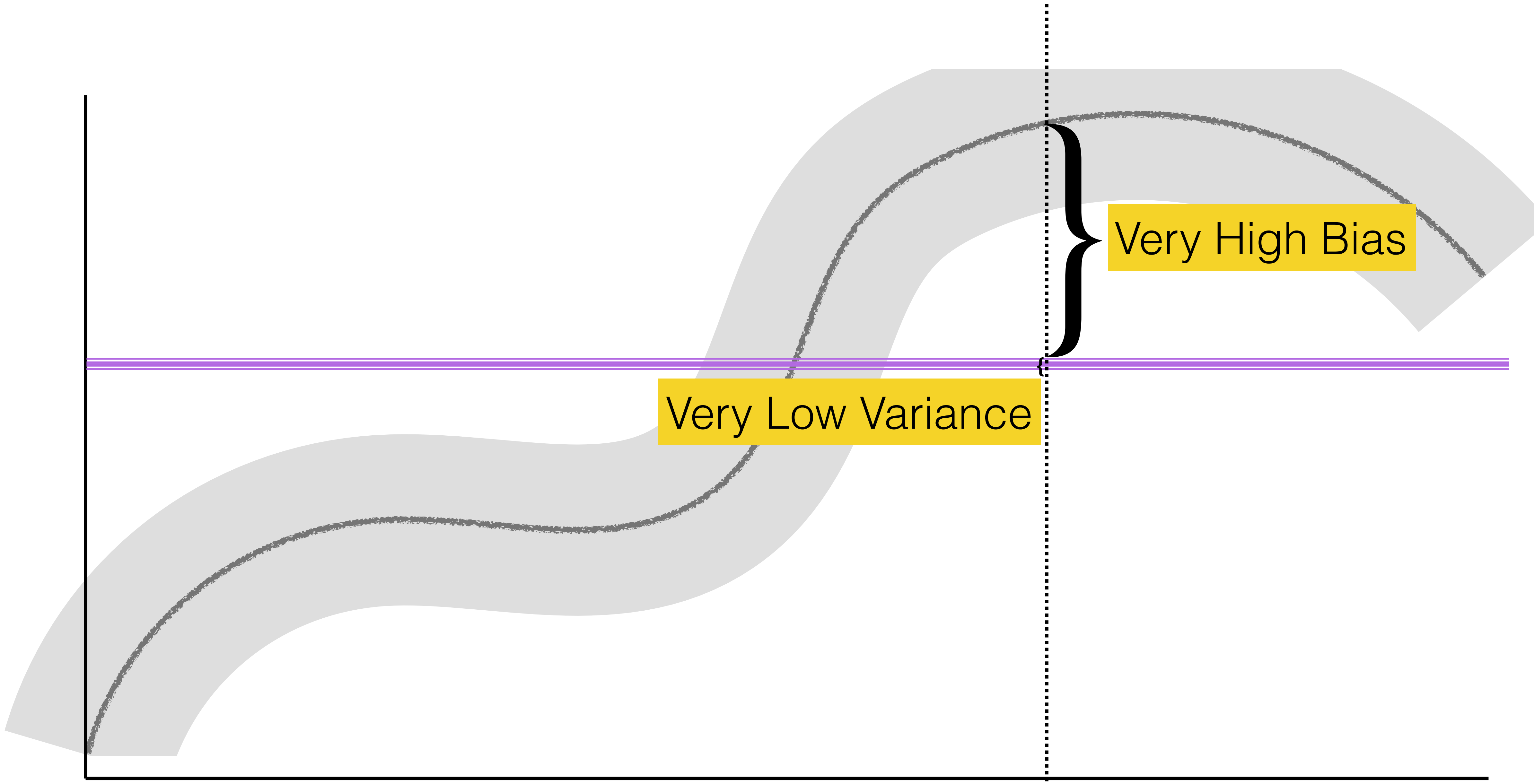


Variance:

- model is over-reactive to small changes
- model is ***too-sensitive*** to the training data

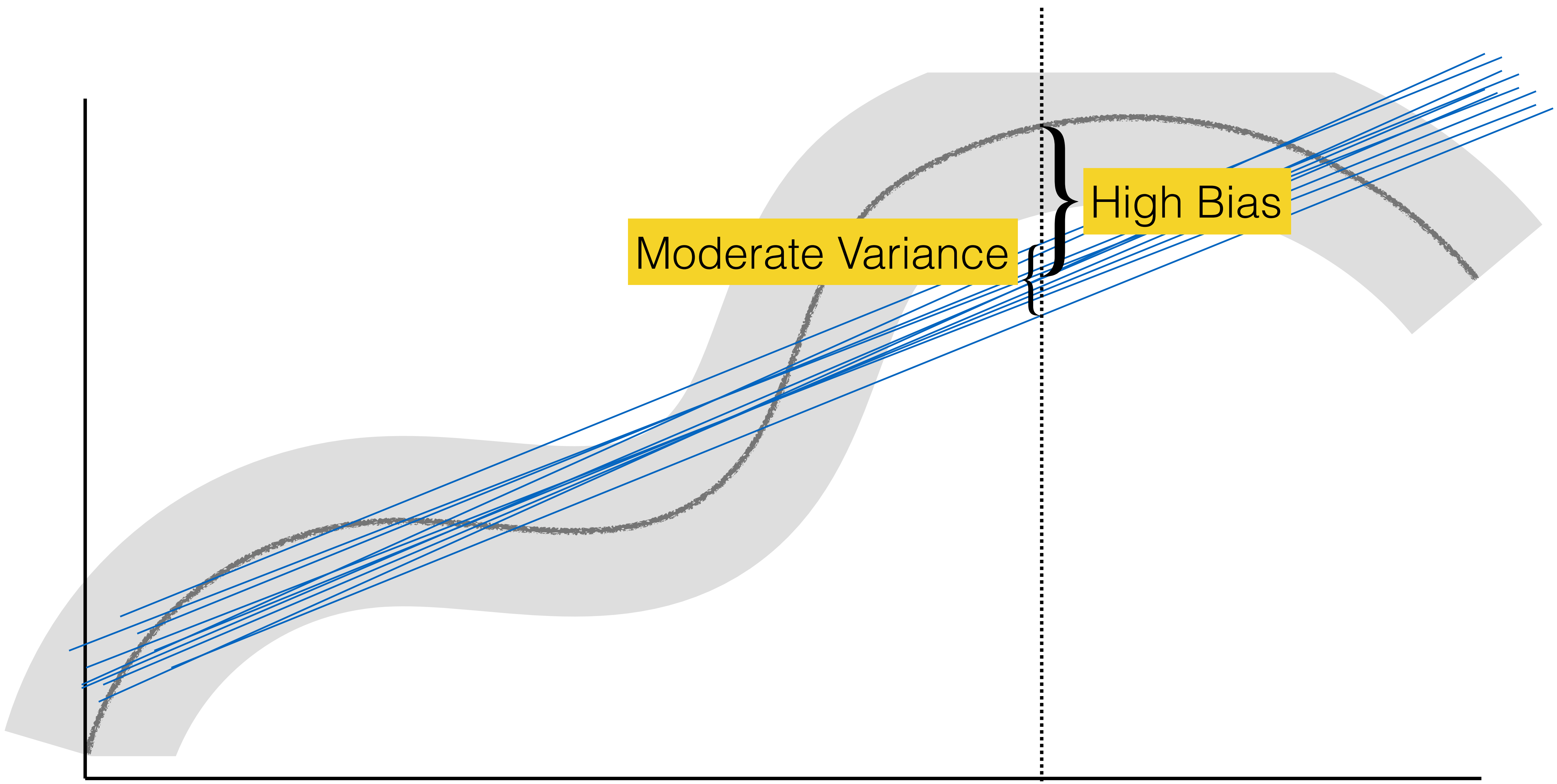
Modeling: Prediction Error

- Learning algorithms face a fundamental tradeoff between ***bias*** and ***variance*** in making predictions:



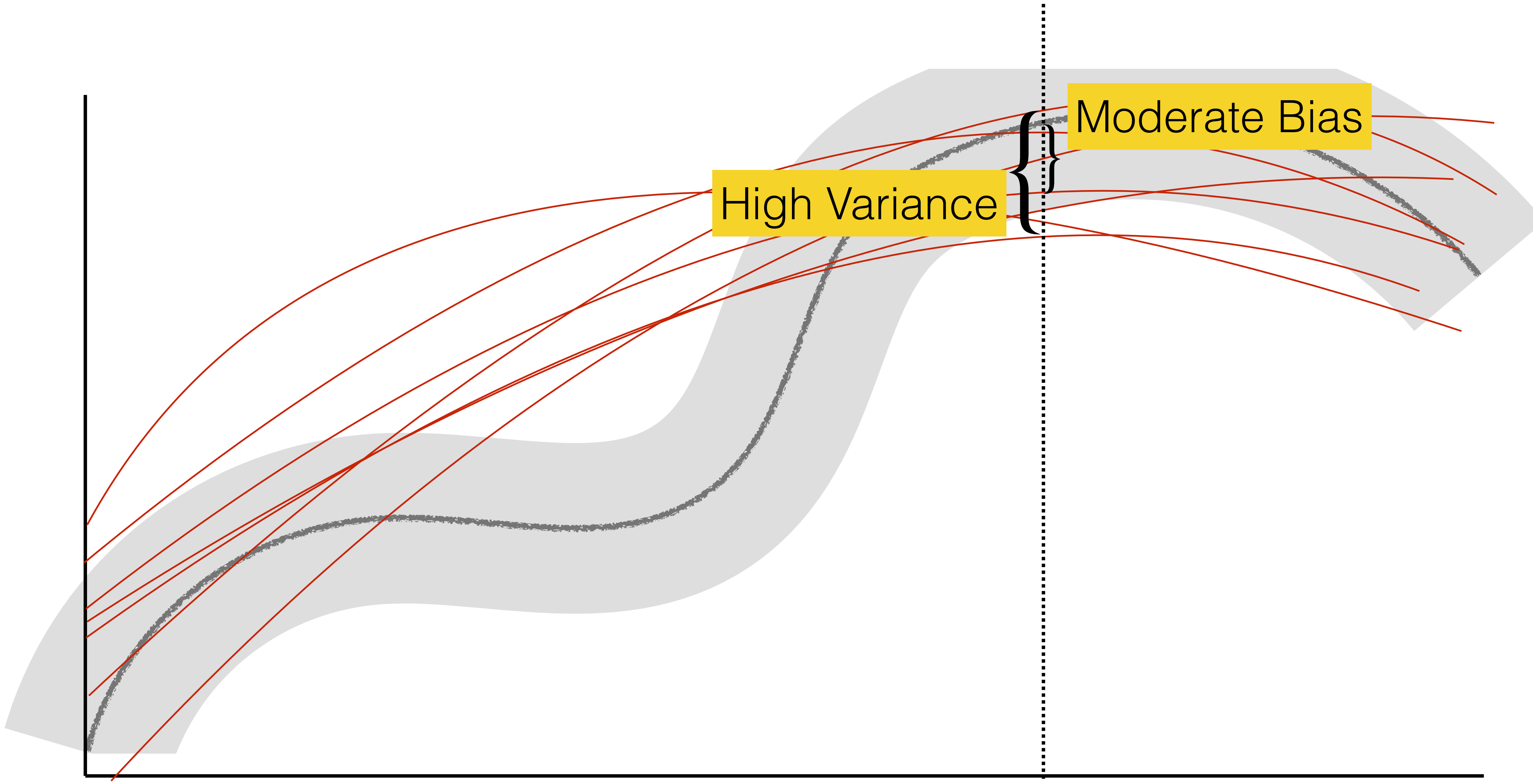
Modeling: Prediction Error

- Learning algorithms face a fundamental tradeoff between ***bias*** and ***variance*** in making predictions:



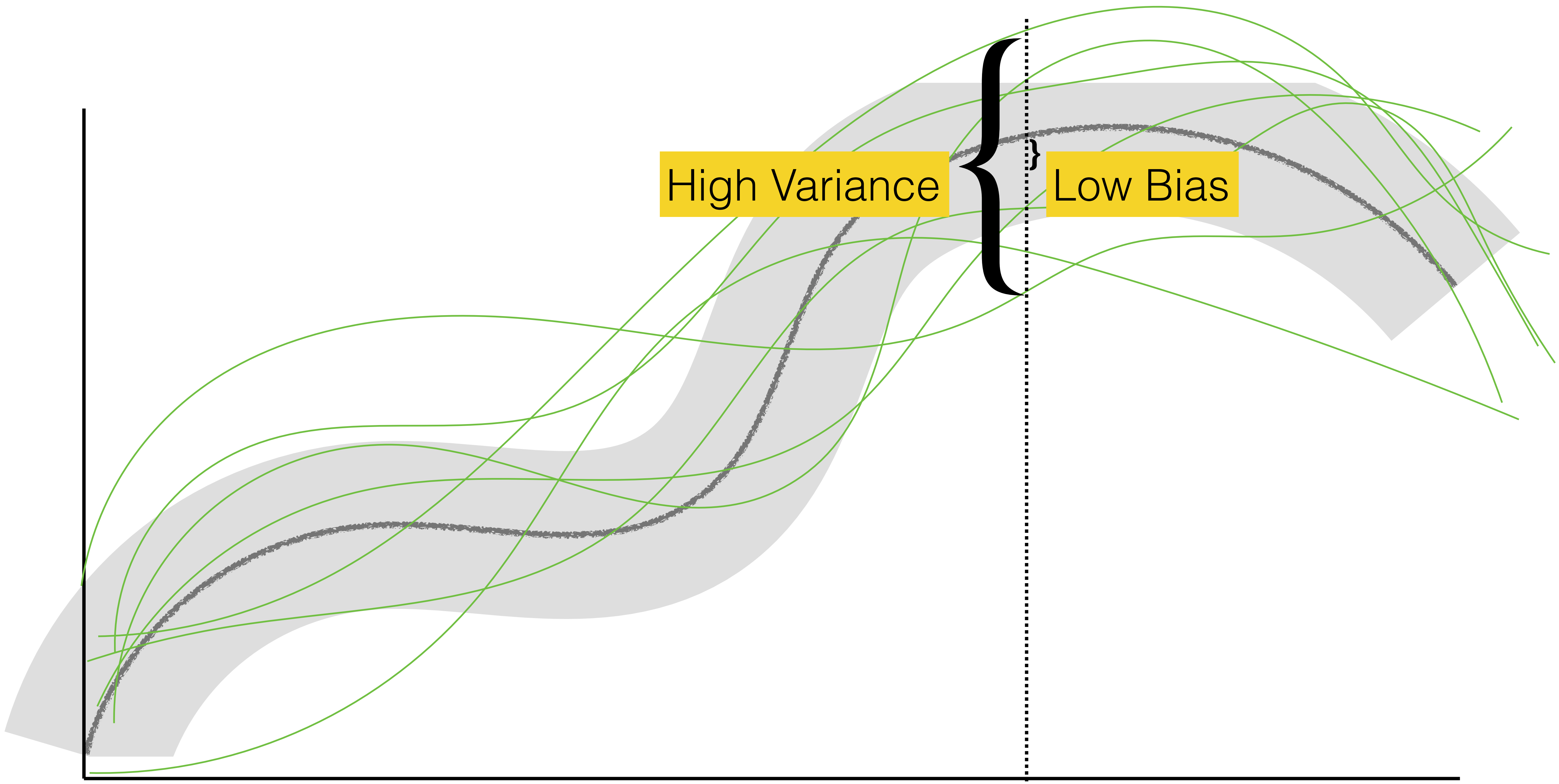
Modeling: Prediction Error

- Learning algorithms face a fundamental tradeoff between ***bias*** and ***variance*** in making predictions:



Modeling: Prediction Error

- Learning algorithms face a fundamental tradeoff between ***bias*** and ***variance*** in making predictions:



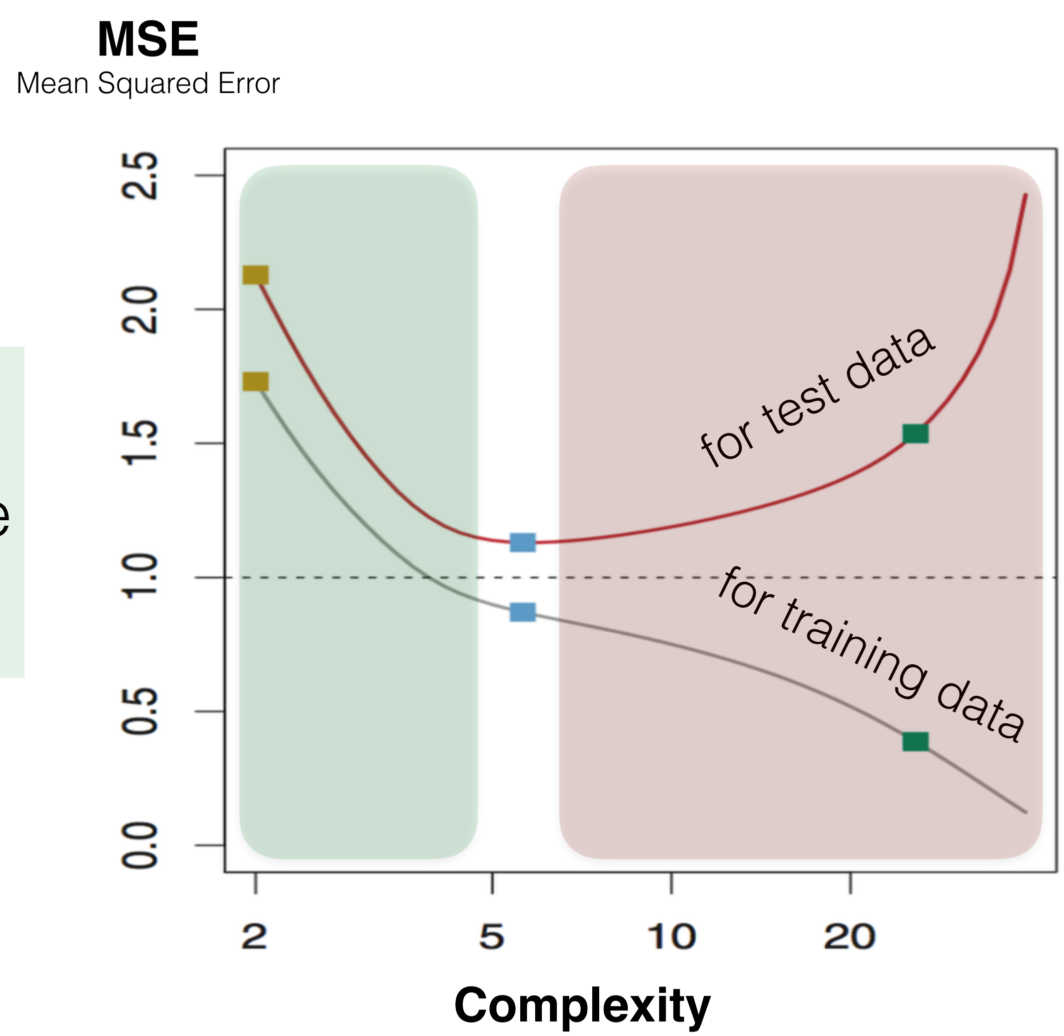
Modeling: Prediction Error

- The bias-variance tradeoff can also be thought of as a tradeoff between ***memorizing*** and ***generalizing*** from a set of data

Under-fitting

model is too-simple

not learning signal



Over-fitting

model is too-complex

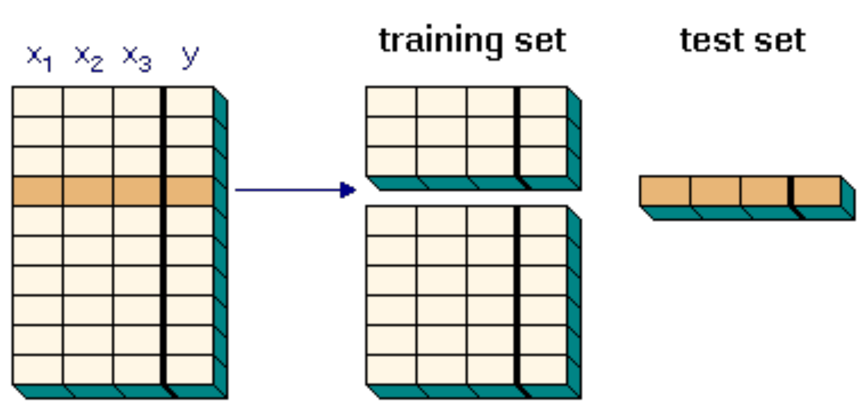
learning from noise

Validation

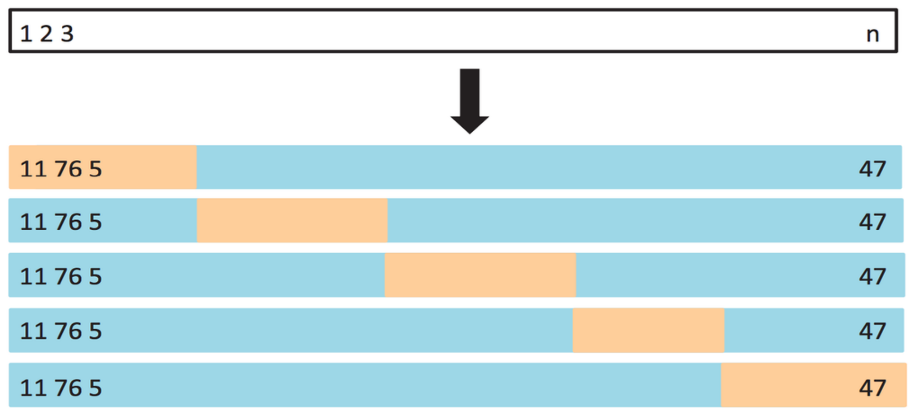
- To "predict" implies that you do not know the outcome

- To "validate" a prediction (i.e., "demonstrate truth or value"), we need to test our model on different data than we used to "learn"
  - Ideally you get test data from a new (but similar) source.
  - But test data can be hard to obtain.

- So typically you use "cross-validation" and do it "k-fold" times



Source: http://www.statistics4u.com/fundstat_eng/cc_cross_validation.html



Source: Introduction to Statistical Learning, James, Witten, Hastie, and Tibshiran, 2013.

Lecture: Text as Data

Text as Data

- Analyzing text can be important in business contexts.
- Text is everywhere, but it is usually unstructured, without a fixed meaning, and dependent on the surrounding context, tone, etc.
    - assessing product reviews
    - mapping out competing products
    - analyzing financial data
- True understanding of text requires computational linguistics and/or modeling of complicated sequences of words
    - That's more than we can cover in this course…

      … but there are basic tools that work pretty well that we can still use

Text as Data

- Core idea is to convert text into ***weighted vectors*** that then point into a high-dimensional **Vector Space Model**

- Frequency-based models
  - **bag of words**

    simplest approach — we will use bag of words
  - n-gram sequences
  - named entity extractions

- Latent-topic-based models

  Why are these "*latent*" ?
  - latent topic models (LSA, LDA, …)

- Prediction-based models
  - word-to-vec & other neural embeddings of text

Text as Data: Bag of Words

- Identify words (“terms”)
  - each term from a “vocabulary” defines a separate dimension
  - for simplicity, we assume all word dimensions are **orthogonal**
    - each and every word is not actually orthogonal, but the overlap seems to average out for large vocabularies
  - to find identify words, you need to:
    - ***parse*** (or ***tokenize***) the words (difficult for Chinese!)
    - remove ***stopwords*** (but this is debatable… more later)
    - ***stem*** words to a common root (and not break this so badly

Text as Data: Bag of Words

- Count terms
  - For each term in the document, you need to calculate:
    - **tf** *term-frequency* (# of times term appears in doc)
    - **df** *document-frequency* (# of documents term appears in)
  - And then calculate the inverse *document-frequency*:
    - **idf** $= \ln\left(\frac{\text{\# docs}}{\textbf{df}}\right)$
    - **tf*idf** = **tf** * **idf**
  - Note how **tf*idf** pushes the “weight” of common terms toward zero

Text as Data: Bag of Words

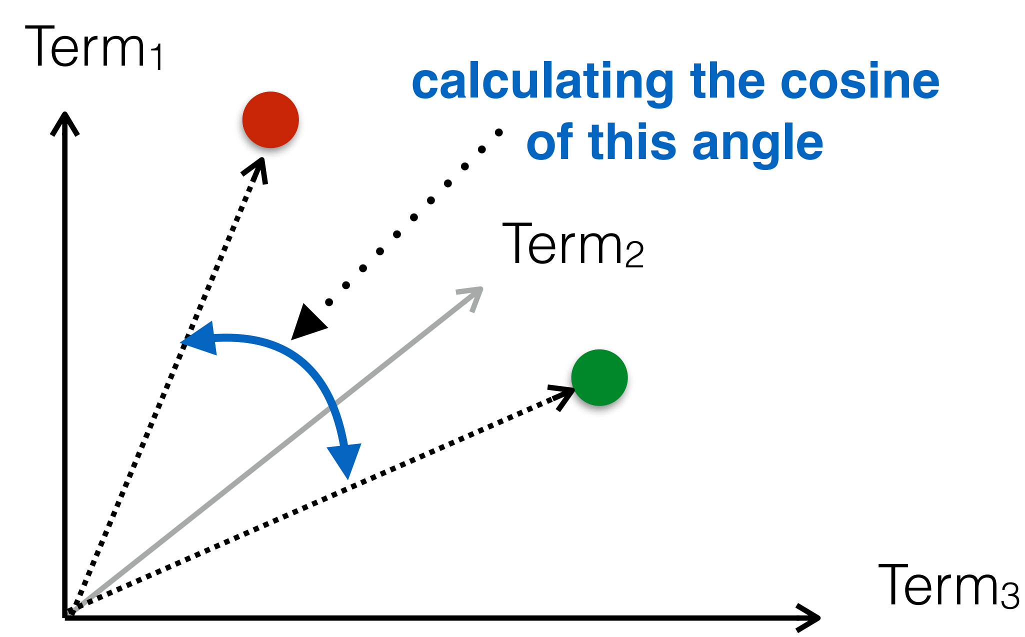
- Vectorize each document
  - Each document encoded into a vector into the high-dimensional Vector Space.
  - Note that sequential meaning of text is lost (it is a "***bag***" of words)
  - Any one vector will be "sparse"
    - most documents do not have most words, so those dimensions are zero
    - there are efficient tools for handling sparse vectors

Text as Data: Bag of Words

- Calculate the similarity between documents

  - ***Cosine similarity*** is a commonly-used metric (with nice properties).

    - Cosine similarity is also called "angular separation," for it only depends on the direction of two vectors pointing into the space (which works well with bag of words)



    - If you pre-***normalize*** each vector to unit vector length, then the cosine is a fast and efficient calculation (sum of inner products between two vectors)

  - But one can use Euclidean Distance (L_2), or Squared Euclidean Distance, or…

Text as Data: Bag of Words — Cosine Similarity

- Calculate the similarity matrix

  - The end objective is often to calculate a ***similarity matrix***.

    - Similarity matrix is a symmetric, square matrix of every pairwise document comparison

      - entries on main diagonal are all 1.0 — i.e., perfect similarity, because cosine(0)=1

      - off-diagonal entries are all positive and range [0.0, 1.0] — i.e., the range of cosine

	1234568	1234569	1234570	1234571	1234572	1234573	1234574
1234568	1.000						
1234569	0.215	1.000					
1234570	0.272	0.395	1.000				
1234571	0.384	0.158	0.715	1.000			
1234572	0.618	0.577	0.000	0.686	1.000		
1234573	0.301	0.715	0.326	0.755	0.682	1.000	
1234574	0.866	0.932	0.747	0.175	0.265	0.170	1.000

    - Many algorithms ask for a ***distance matrix***, which is just *distance* = 1 - *similarity*

Text as Data: Other models

- Latent-topic-based models
    - latent topic models (LSA, LDA, …)
- Prediction-based models
    - word-to-vec & other neural embeddings of text

We will cover these later in the course…

Examples

&

Assignment 1