INFSCI 2915: Machine Learning
Models and Trade-offs

Mai Abdelhakim

School of Computing and Information

610 IS Building

Spring 2018

Last Meeting

- Ability to learn without being explicitly programmed

- Machine learns with respect to a particular task T, performance metric P and experience E, if the performance P on task T improves with experience E.

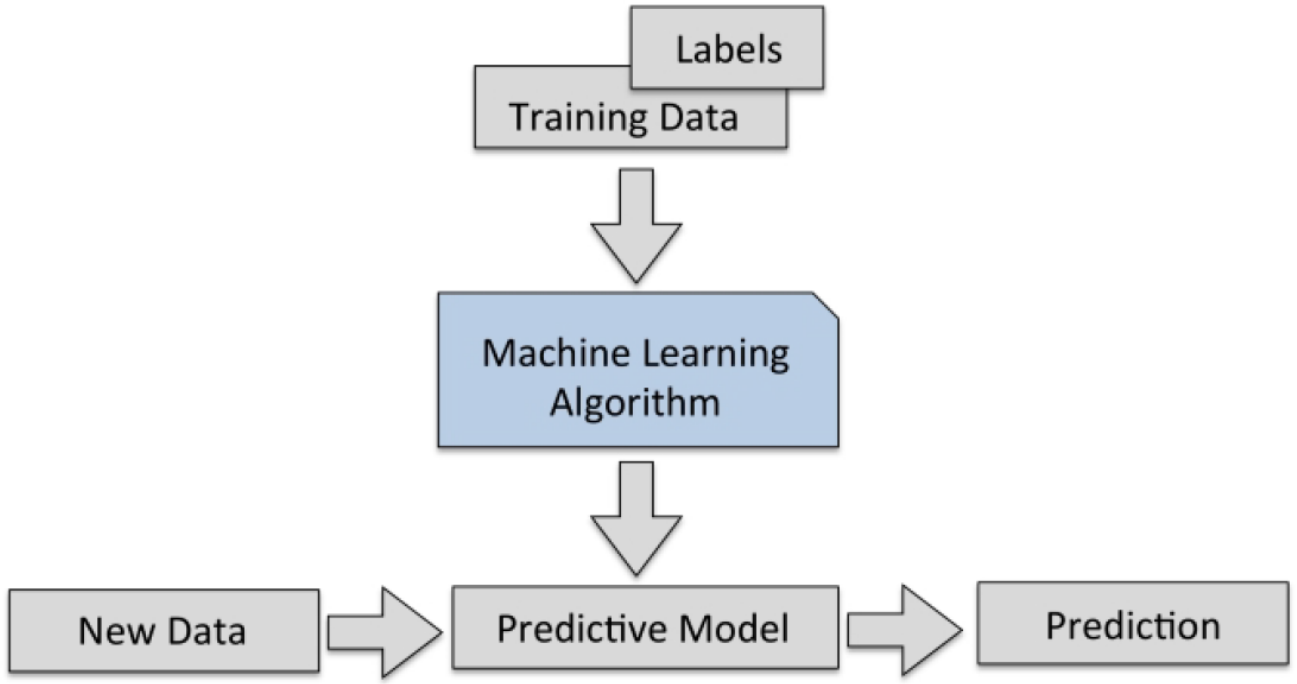
- Algorithms:
  - Supervised Learning
  - Unsupervised learning

Last Meeting

Supervised Learning: predict target values from labeled data

- **Regression**: Target values (Y) are continuous/quantitative
- **Classification**: Target values (Y) are discrete/finite/qualitative

Labels can be obtained manually. Some tools available, e.g. Amazon Mechanical Turk: https://requester.mturk.com/case_studies (workers provide labels)



Sabastian Raschka, Python Machine Learning

Last Meeting

Unsupervised learning: Find structure in unlabeled data

- Examples: market segmentation, social analysis, anomaly detection
- Hard to evaluate, (can be evaluated manually by looking at the data)

This Meeting

- Models
- Evaluation
  - Training and test
  - Accuracy
- Tradeoffs
  - Flexibility/complexity versus accuracy
  - Bias-variance trade-off
- K-Nearest Neighbor

How to make predictions?

- Can we predict the output using the features?

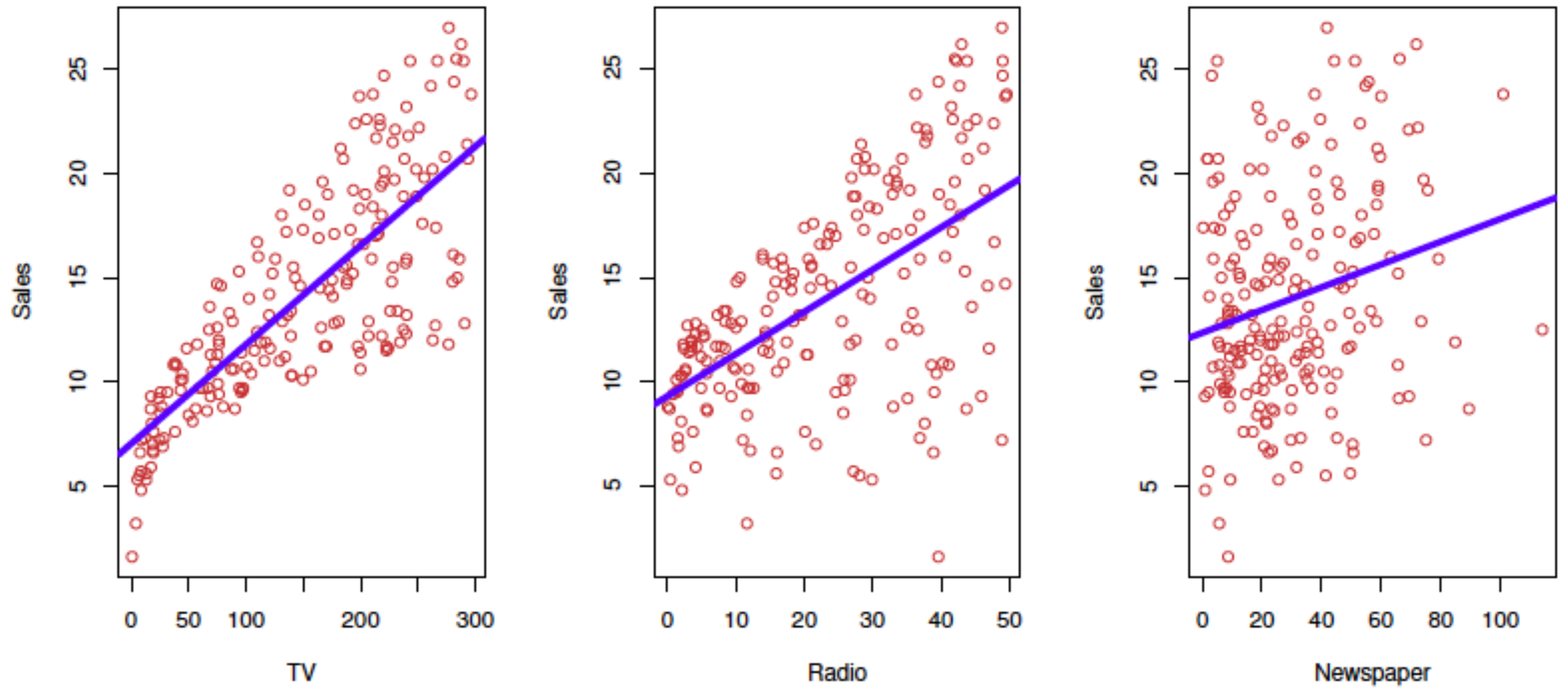
- It is helpful to have a **model**: $Y \approx f(x)$

- Example:
  - Assume you are a marketing consultant, and want to suggest marketing plan for next year that results in **high product sales**
    - Given TV, Radio, newspaper **advertising budget** as input features

Advertising Example – Regression Problem

- Advertising dataset: sales function of TV, radio, newspaper budgets?

$$\text{sales} \approx f(\text{TV, radio, newspaper})$$



Sales in thousands of units, vs TV, Radio, Newspaper in thousands of dollars for 200 markets

Why do we need a model? Why estimate f?

- 1 **Predictions**: Make predictions for new inputs/features

- 2 **Inference**: understand the way Y is affected by each features

  - Which feature has stronger **impact on the response**?
  - Is relation **positive or negative**
  - Is the relationship **linear or more complicated**

  - Example: Not interested in predicting house price, but want to know the impact of river view on a house value

From Advertising Example

- **Is there a relationship** between advertising budget and sales?
  - If there isn't, then maybe we don't need advertising
- **How strong** is the relationship between advertising budget and sales?
- **Which media** contribute to sales?
- **How accurately can we predict** future sales?
- **Is there synergy** among the advertising media?
  - Spend $50, 000 on television and $50, 000 on radio advertising results in more sales than allocating $100, 000 to either television or radio individually?

Model

- Y: output
    - Target/response/label that we wish to predict. In previous example, Y= sales
- Features/predictors (X)
    - E.g. TV, radio, newspaper budgets
    - Let TV be X_1, Radio be X_2, newspaper be X_3 .. Then $X = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix}$
- We can write

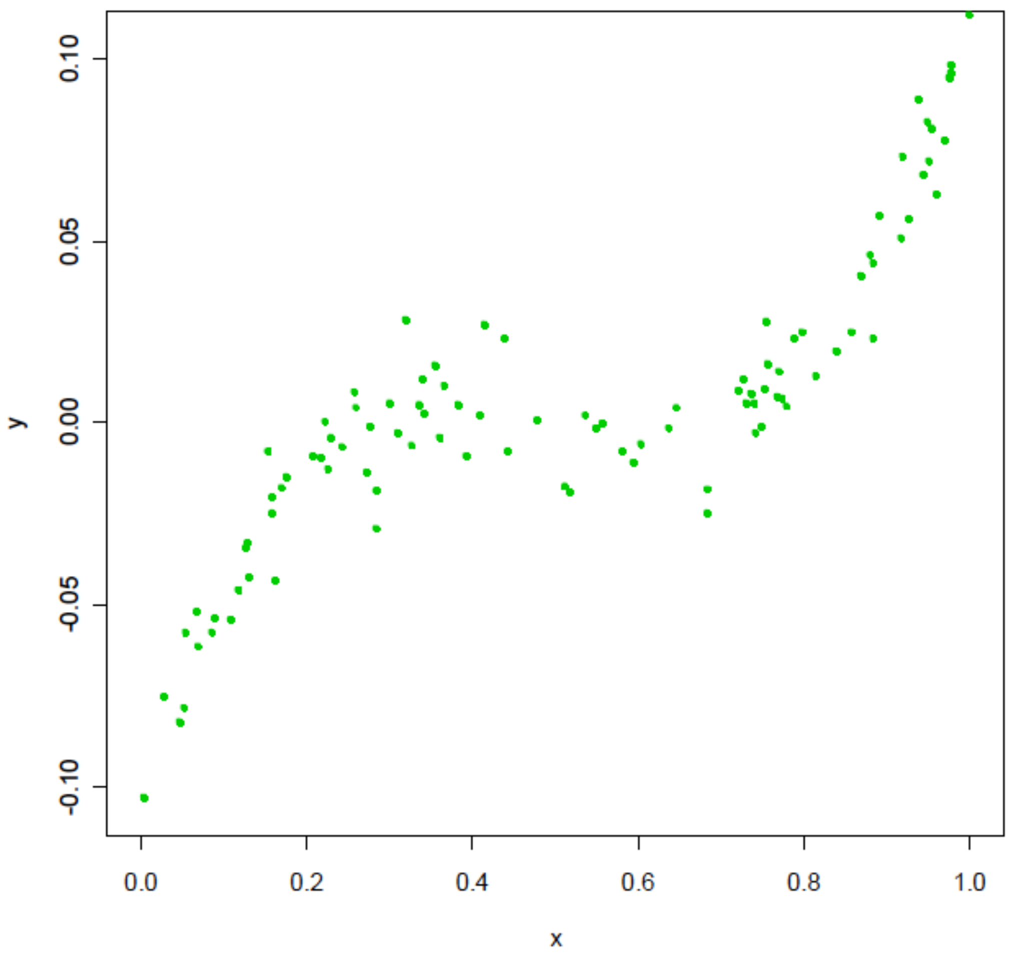
$$Y = f(X) + \epsilon$$

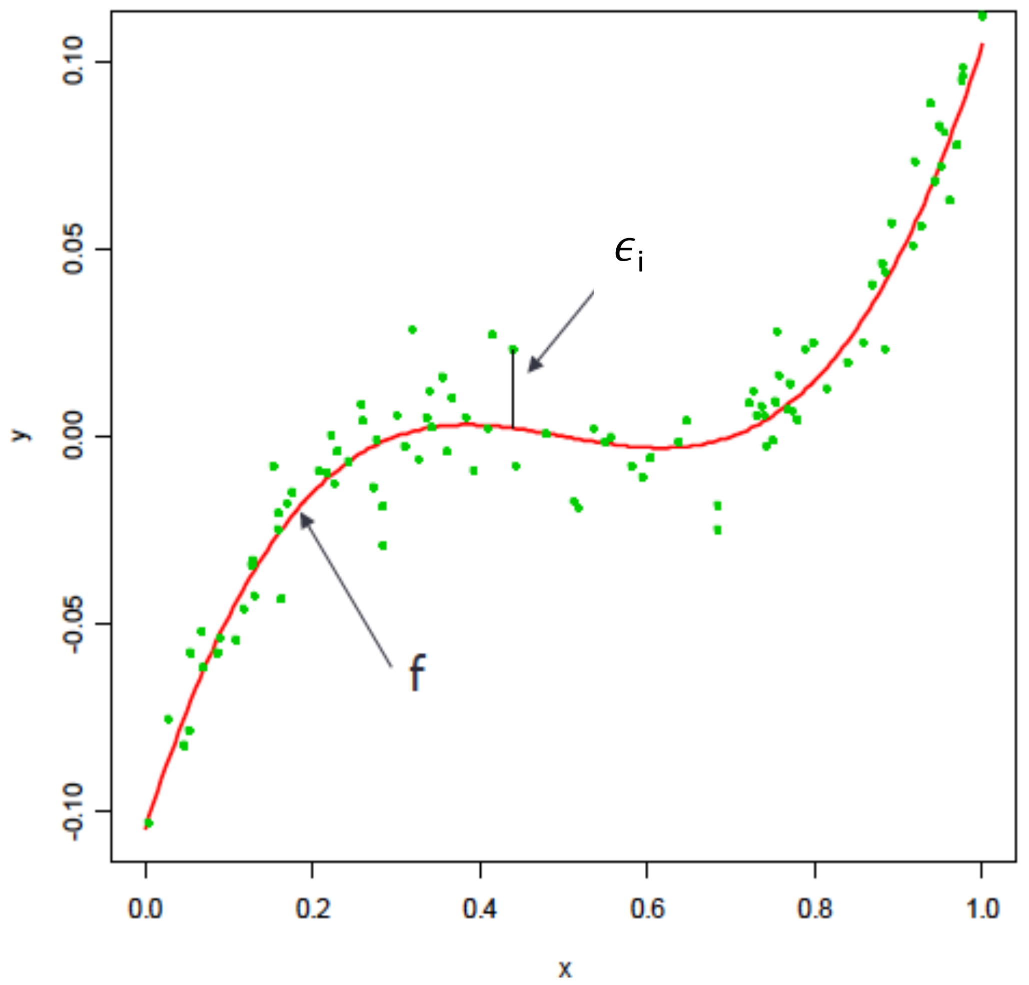
    - ∈ is a random measurements error or due to the distribution of Y at each,
    - f is the unknown function

- **With good estimate of f , we will be able to make predictions.**

Example

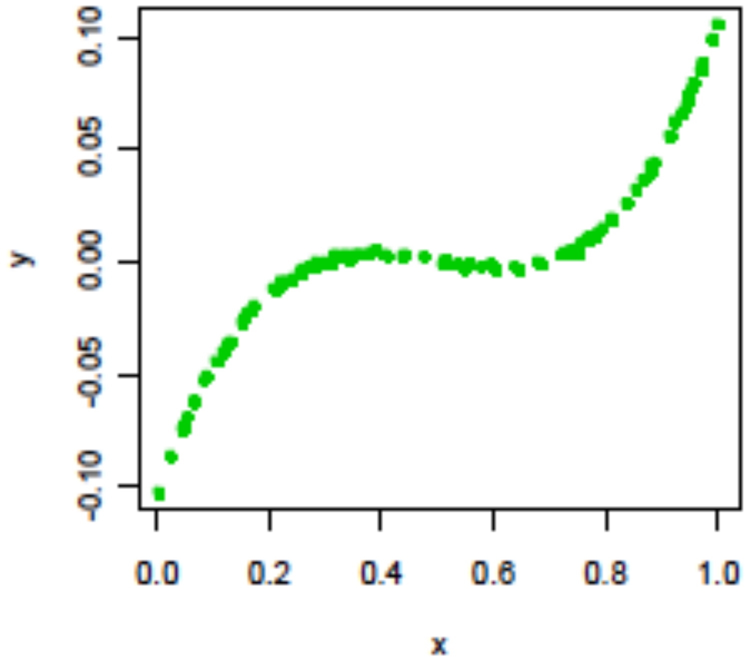
- $Y = f(X) + \epsilon$

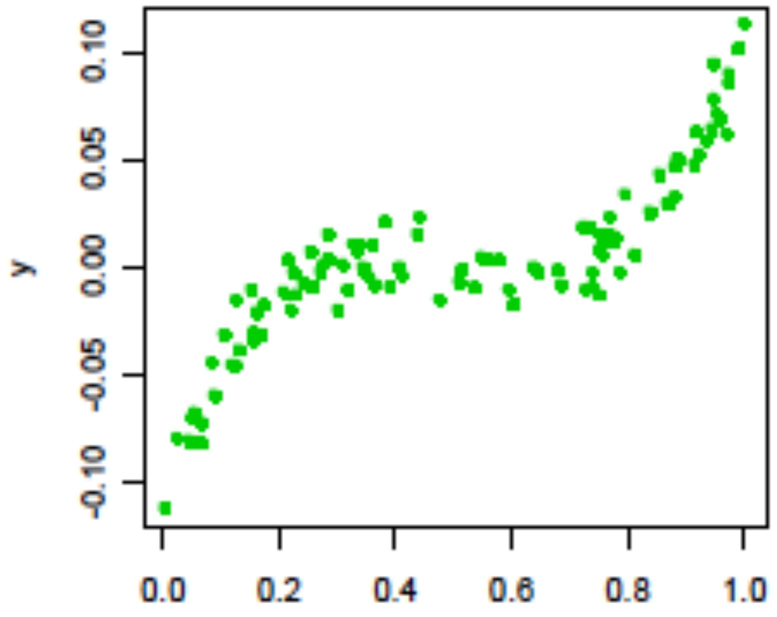




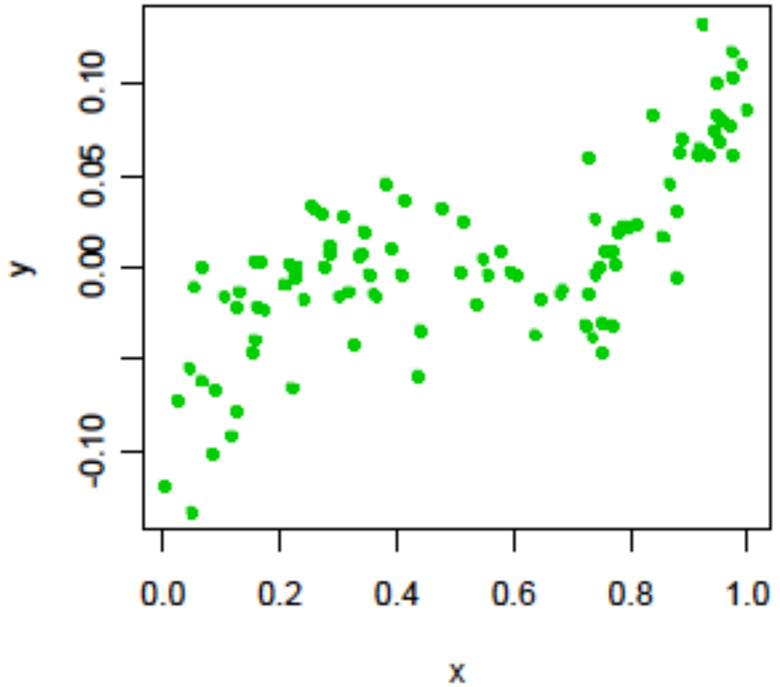
- The variance of the error affects the accuracy of estimating *f*

Variance error = Var(ϵ) = 10^{-6}





Var(ϵ) = 10^{-4}



Var(ϵ) = 9×10^{-4}

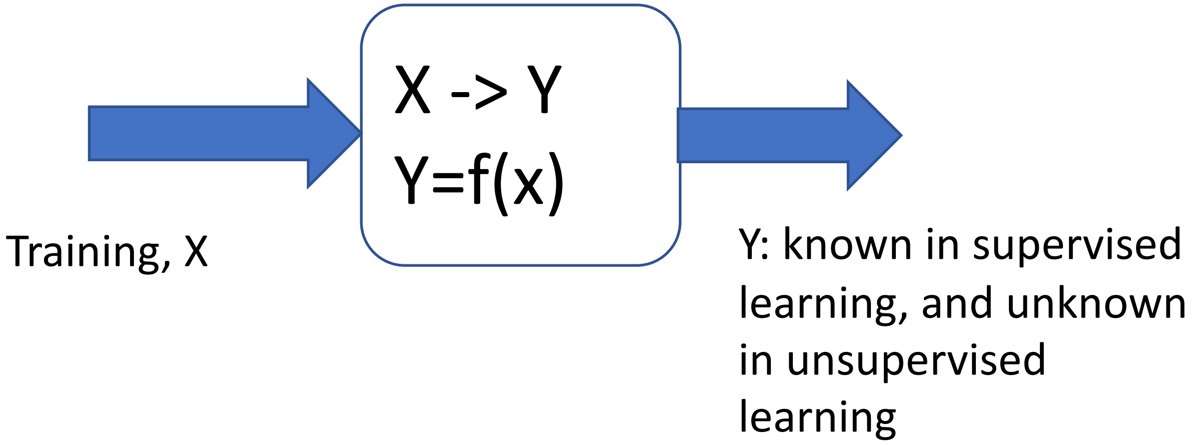
How to estimate f?

- We use data to estimate ("**learn**") *f*.

Training data: $\{(x_1, y_1),\ldots, (x_n, y_n)\}$

Training Phase: the model learns, i.e., estimate function *f*

- Two approaches to estimate *f*
  - **Non-parametric approach**
  - **Parametric approach**


X -> Y
Y=f(x)
Training, X
Y: known in supervised learning, and unknown in unsupervised learning

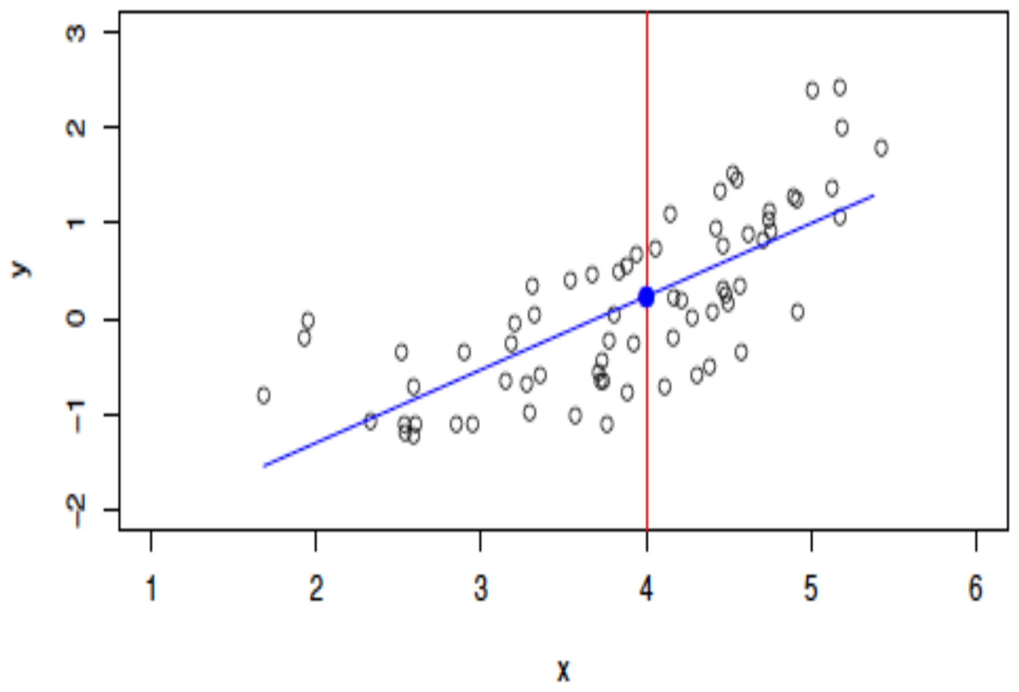
How to estimate f? - Parametric Approach

- First, **assume function form**
  - Example: linear regression, $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots . + \beta_P X_P$

- Second, use training to fit the model (get the parameters of the assumed function)
  - Find β_0, β_1,…., β_P
  - Common approach is ordinary least square (discussed later)

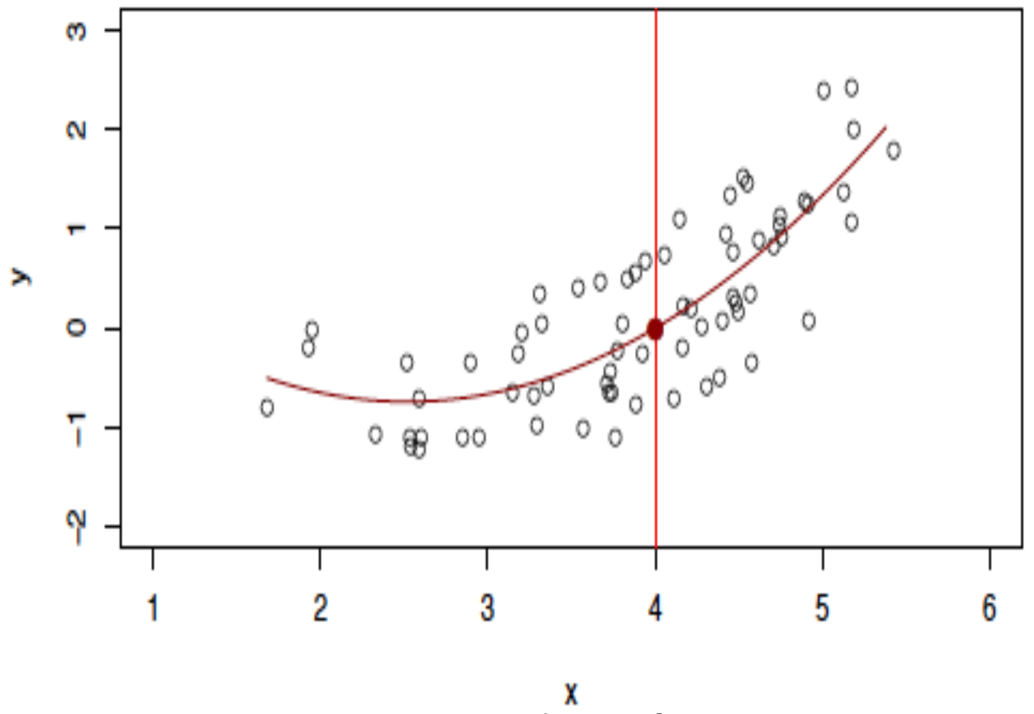
- Examples:
  - income= $\beta_0 + \beta_1 Education + \beta_2 Seniority$
  - $Sales = \beta_0 + \beta_1 TV + \beta_2 Radio + \beta_3 Newspaper$

Parametric Approach – Simple functions



Linear function

$$\hat{f}_L(X) = \hat{\beta}_0 + \hat{\beta}_1 X$$



Quadratic function

$$\hat{f}_Q(X) = \hat{\beta}_0 + \hat{\beta}_1 X + \hat{\beta}_2 X^2$$

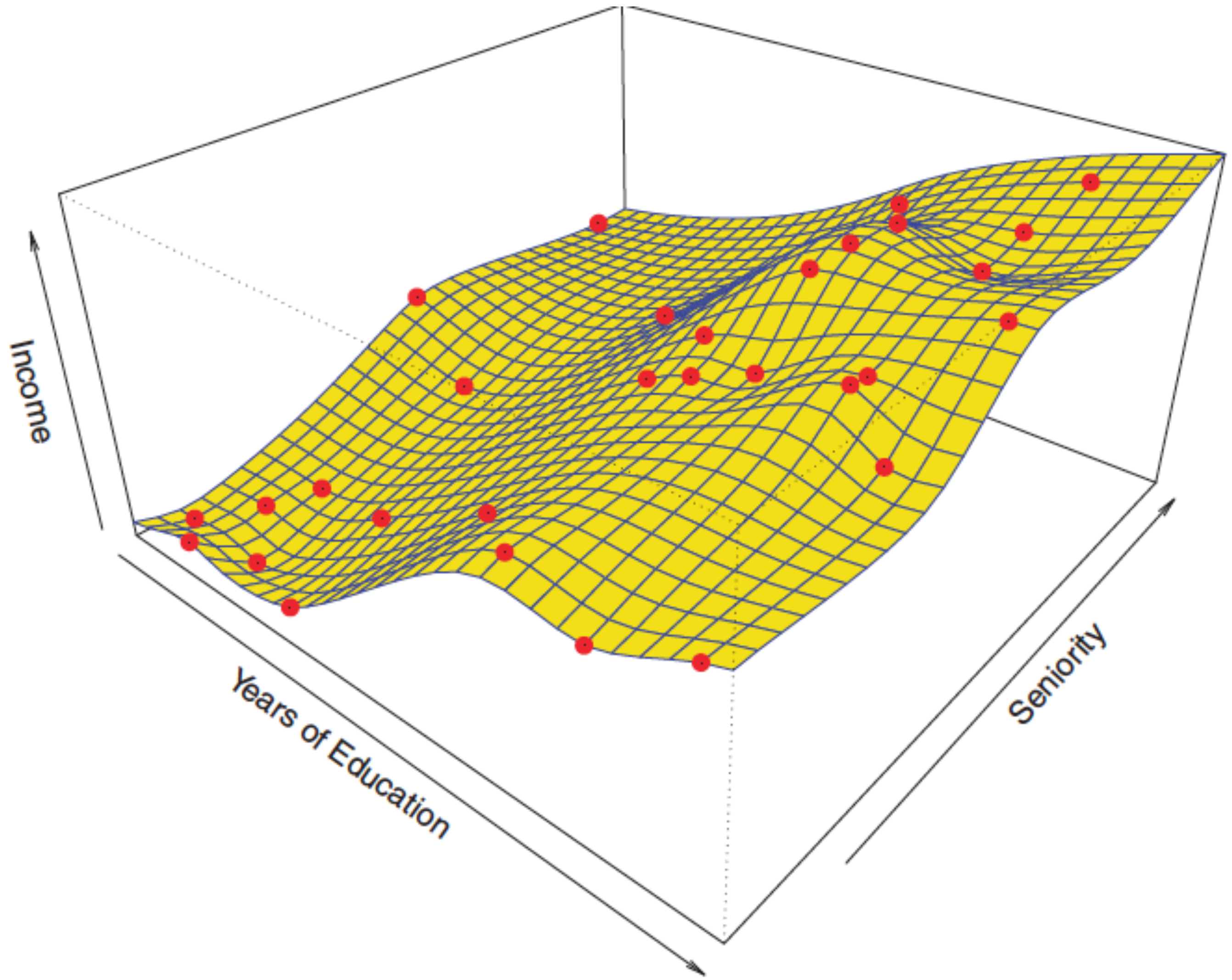
If the assumed model is wrong, then the accuracy of the algorithm will be low

How to estimate f? – Non-Parametric Approach

- No explicit form of function *f* is assumed
- Seek to estimate *f* as close as possible to the data points
- Advantage: Could work well for a wider range of possible shapes of *f*
- Disadvantage:
  - Needs large number of data points
  - Difficult to understand relationship between output and features
- Example: K-Nearest Neighbors (discussed later)

Non-Parametric Approach
More Flexible/Complex Models

- More flexible models is used to estimate *f*
  - E.g. thin-plate spline
- Fit to simulated/training data very well
- May not work well on new data!
  - Why?!
  - This is known as overfitting (discussed later)

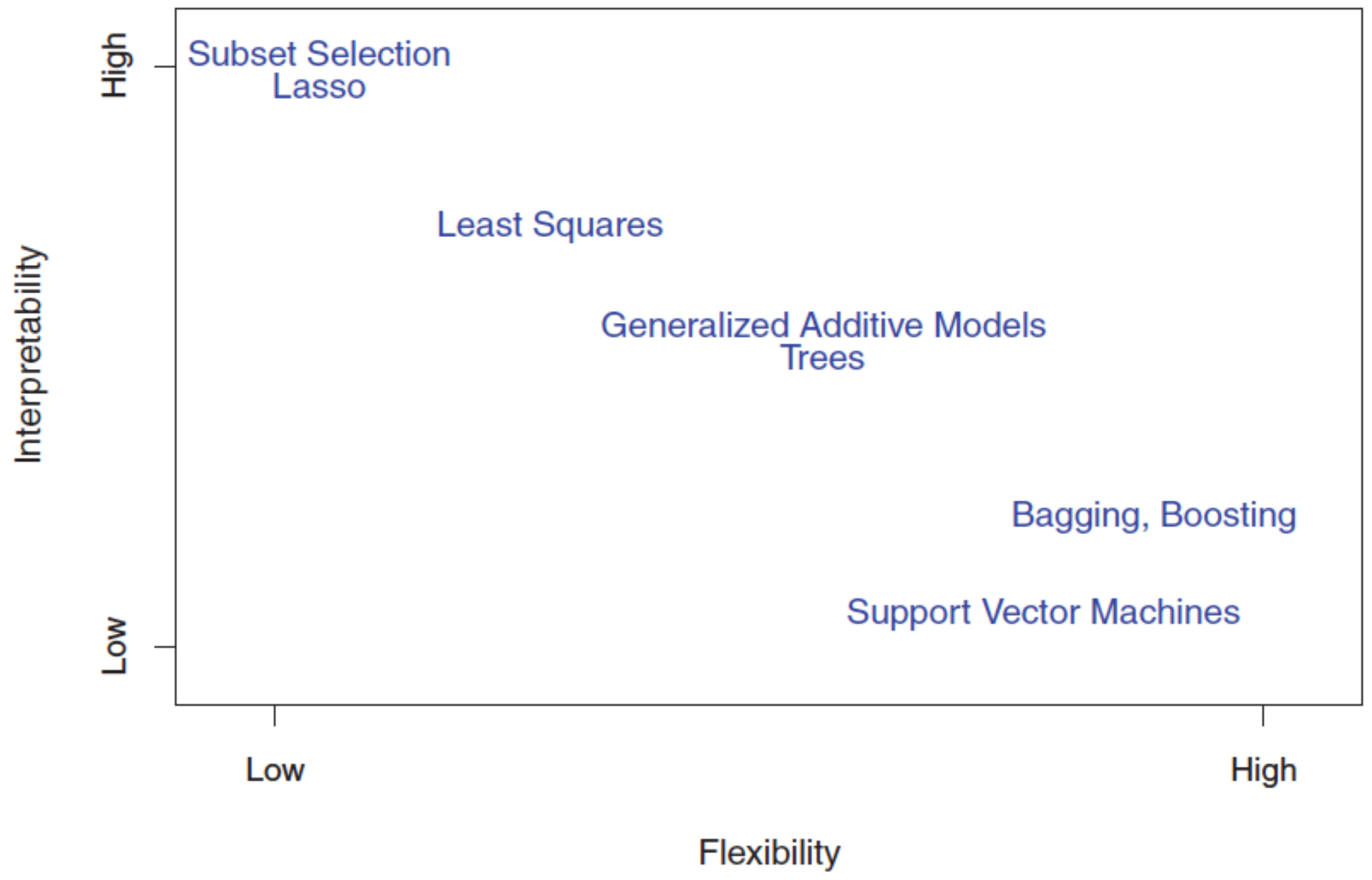


Trade-off: Model Flexibility vs Model Interpretability

- **Less flexible models**, **more restrictive, less complex:** Produce relatively small range of shapes to estimate the function *f*
  - *Simple and **easy to interpret,*** e.g. *Linear regression*
  - *Use when inference is the objective*

- **More flexible models**, **more complex**: generate wide rang of shapes for *f*
  - ***Harder to interpret,** e.g. SVM, thin splines,*

- **More flexible** model would be better for **prediction**
  - But this is not always the case! Overfitting should be avoided
  - It is possible that simpler models lead to higher accuracy

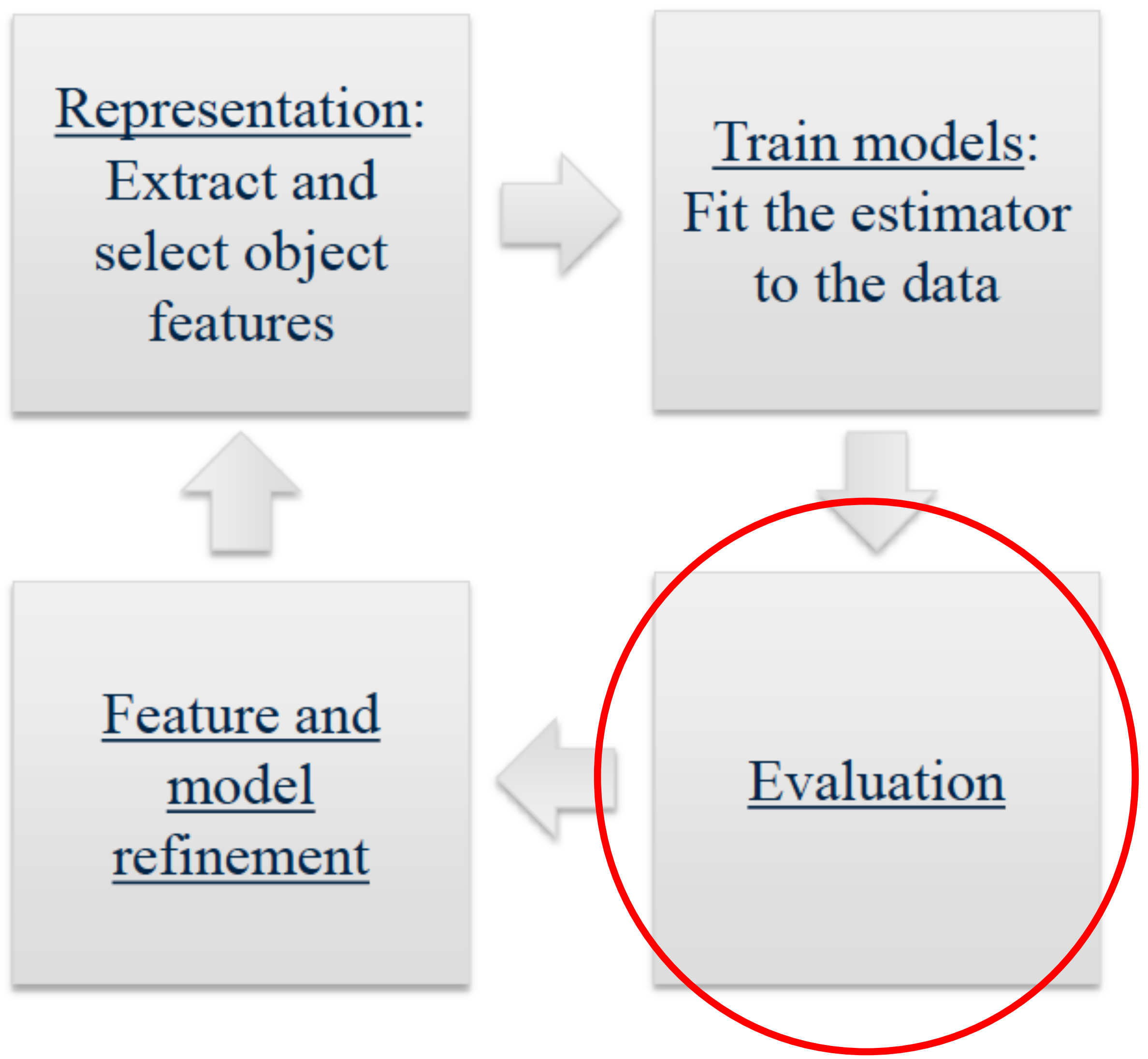

Subset Selection
Lasso
Least Squares
Generalized Additive Models
Trees
Bagging, Boosting
Support Vector Machines
Interpretability
High
Low
Flexibility
Low
High

Assessing the Model Accuracy

How does model flexibility (complexity) affect the accuracy?

- We will focus on supervised learning in this course
- Will cover unsupervised learning later in the semester

Machine Learning Process


Representation: Extract and select object features
Train models: Fit the estimator to the data
Evaluation
Feature and model refinement

Assessing the Model Accuracy – Regression Setting

- Measuring the **Quality of Fit**
  - How good the predictions match the actual data

- In **regression setting, a common measure is mean squared error (MSE)**

- Recall: **n training data points** are (x_1; y_1), (x_2; y_2), …, (x_n; y_n) = $\{x_i; yi\}_{i=1}^{n}$
  - True model $is\ y = f(x) + \epsilon$, and model estimates *f(x)* as $\hat{f}(x)$

- Then **training MSE** is

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$$

where $\hat{f}(x_i)$ is the prediction of the *i*th training sample (y_i).. Also referred to as $\hat{y}$

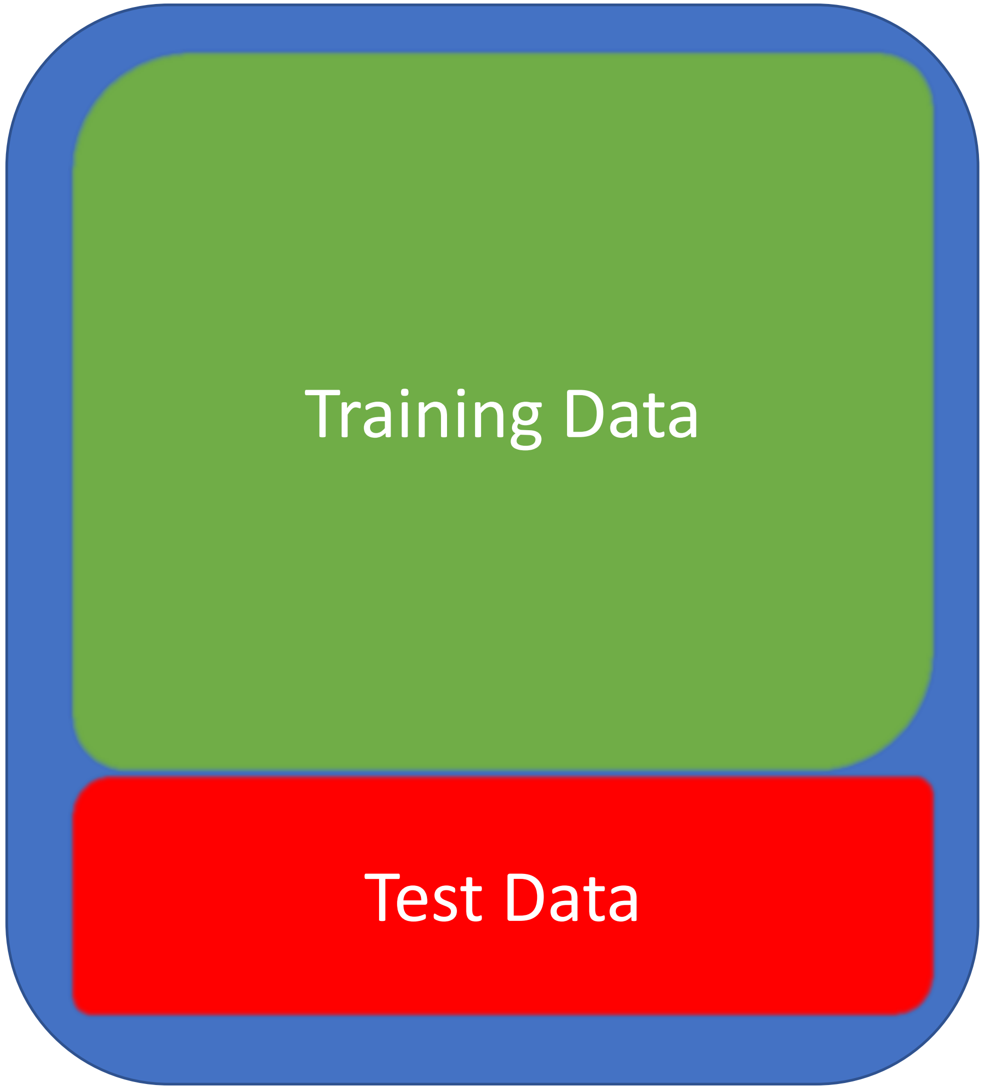
- However, we are interested in how well the model works when it is applied to previously unseen data
  - Evaluating accuracy on training data is not very helpful!
- Therefore, we evaluate **MSE on test data sets**
  - Test data set has observations that were not used to train the learning model
  - No guarantee that model with low training MSE will have low test MSE
- When (x_0, y_0) is previously unseen by the model, then **test MSE** is

$$average\left(y_0 - \hat{f}(x_0)\right)^2$$

  - This is the average square prediction error for the test observations

Training and Test Sets

- Measure the model ability to generalize well, i.e., performs well on new data
  - Thus, we can not use the training data for testing
- Typical approach: split the available data (features-response pairs) into two parts:
  - **PART I - Training set**: for **building/fitting** the machine learning algorithm
  - **PART II - Test set**: for **testing the accuracy** of the model
- Typically, data is split with 75% training set, and 25% test set
- **Choose the model that has lowest test MSE**



Overfitting and Underfitting

How does model flexibility (complexity) affect the accuracy?

Two thing we need to avoid:

- **Overfitting**: Building a model that is too **complex**, fits training data very well, but **fail to generalize** to new data (e.g. large test MSE)

- **Underfitting**: build **simple model** that is **unable to capture variability in data**
  - Does not work well even on the training set

Example

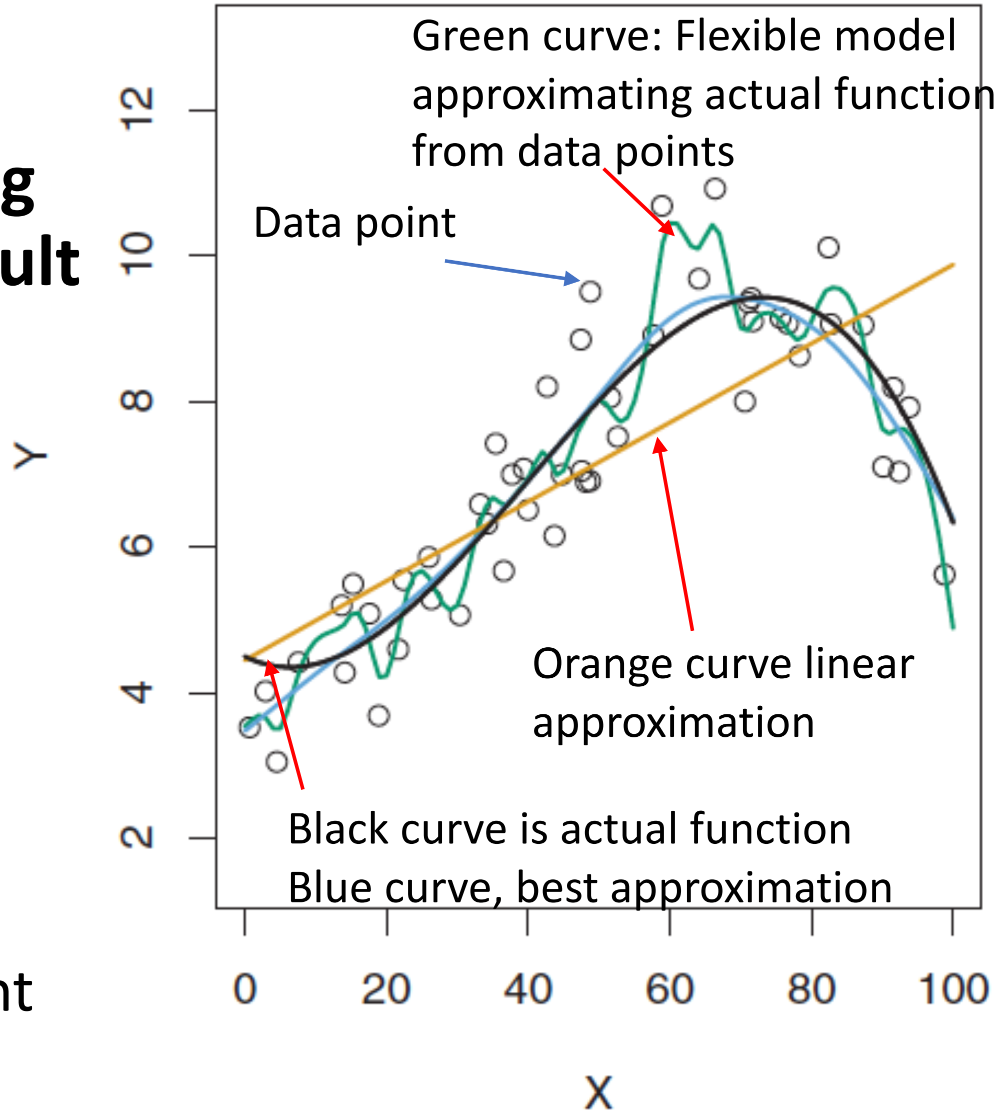
- Predict whether a customer will buy a boat to send promotions
- Given records of previous buyers
- Complex rules that work well on training: Buy a boat if
  - Age 66, 52, 53, 58,..
- **Complex** rules supported by little data results in **Overfitting**
- Over **simplified** rules result in **underfitting**
  - E.g. All who has a house buy a boat

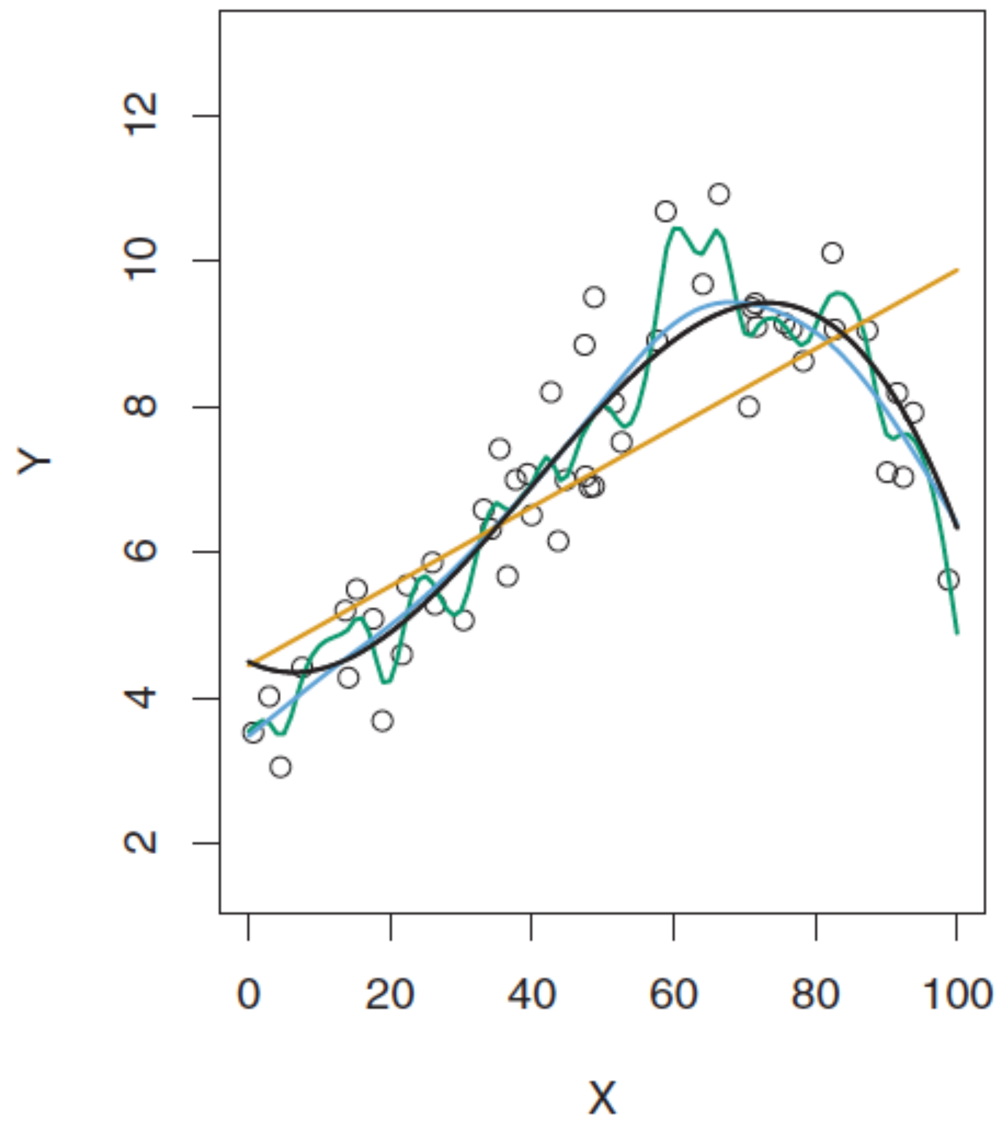
Age	Number of cars owned	Owns house	Number of children	Marital status	Owns a dog	Bought a boat
66	1	yes	2	widowed	no	yes
52	2	yes	3	married	no	yes
22	0	no	0	married	yes	no
25	1	no	1	single	no	no
44	0	no	2	divorced	yes	no
39	1	yes	2	married	yes	no
26	1	no	2	single	no	no
40	3	yes	1	married	yes	no
53	2	yes	2	divorced	no	yes
64	2	yes	3	divorced	no	no
58	2	yes	2	married	yes	yes
33	1	no	1	single	no	no

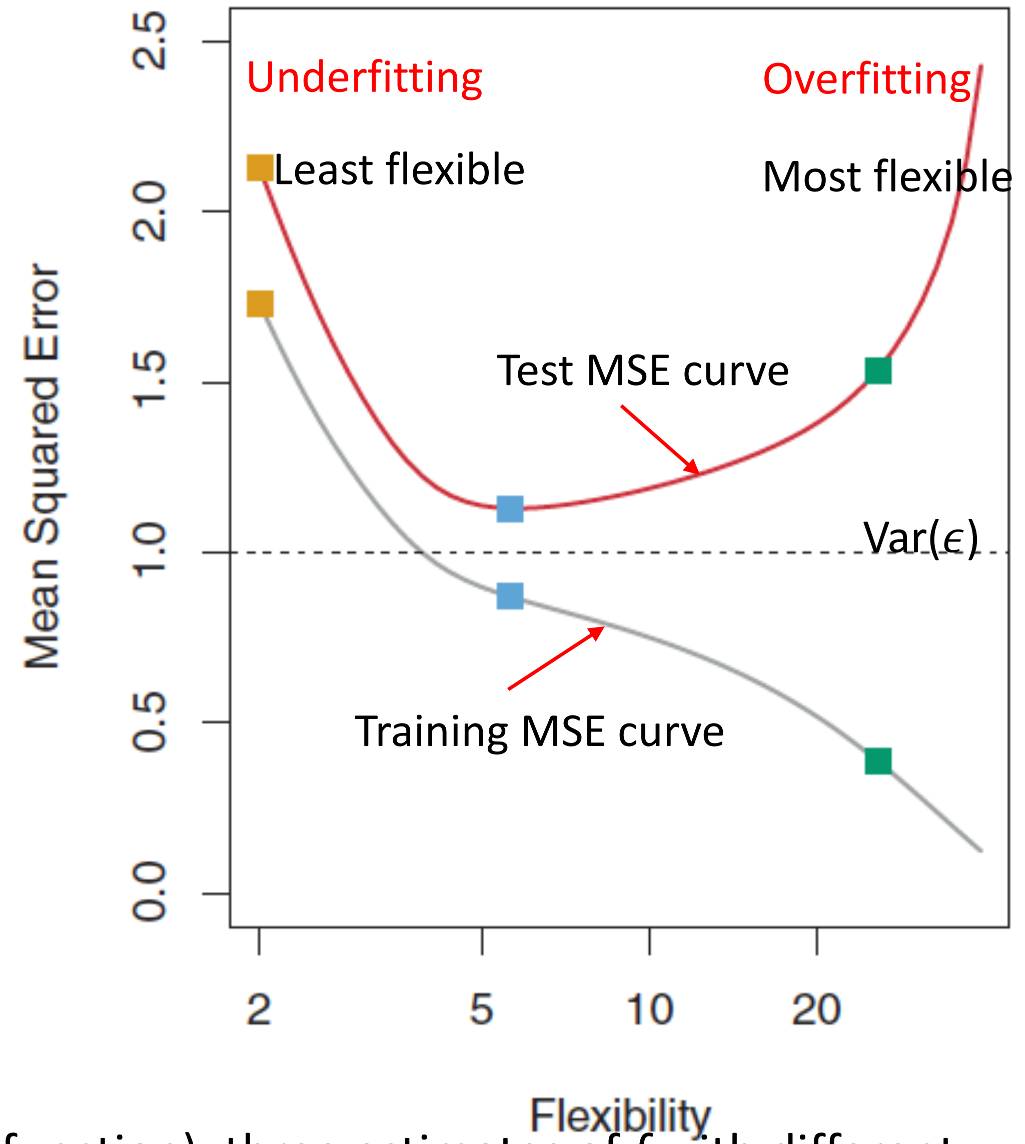
- **More flexible** models tend to **have low training MSE**, but if not carefully designed they can **result in high test MSE.**

- **Less flexible model or more flexible model works well?**
  - **Depending on application and the data**

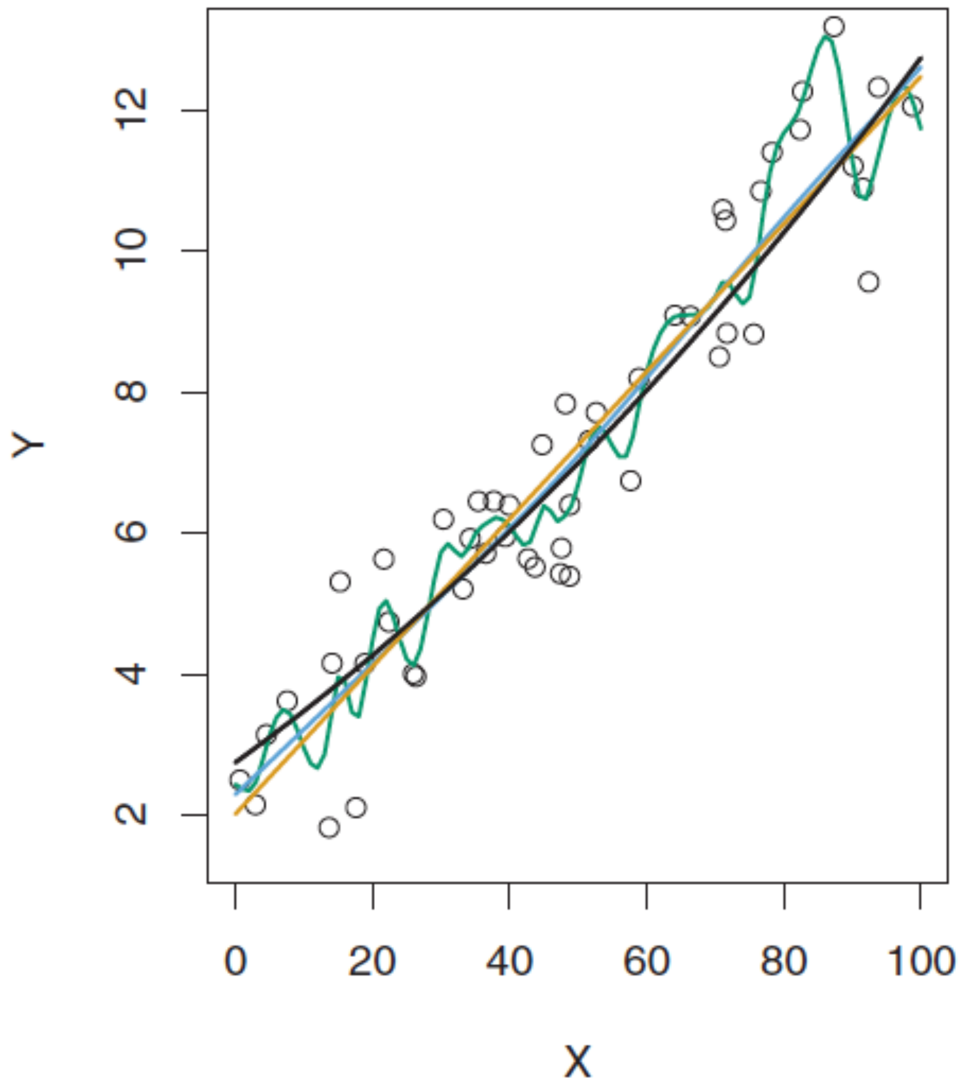
Example in fig.: Data point drawn from black curve, and there are three models (with different flexibilities) are used to fit curve

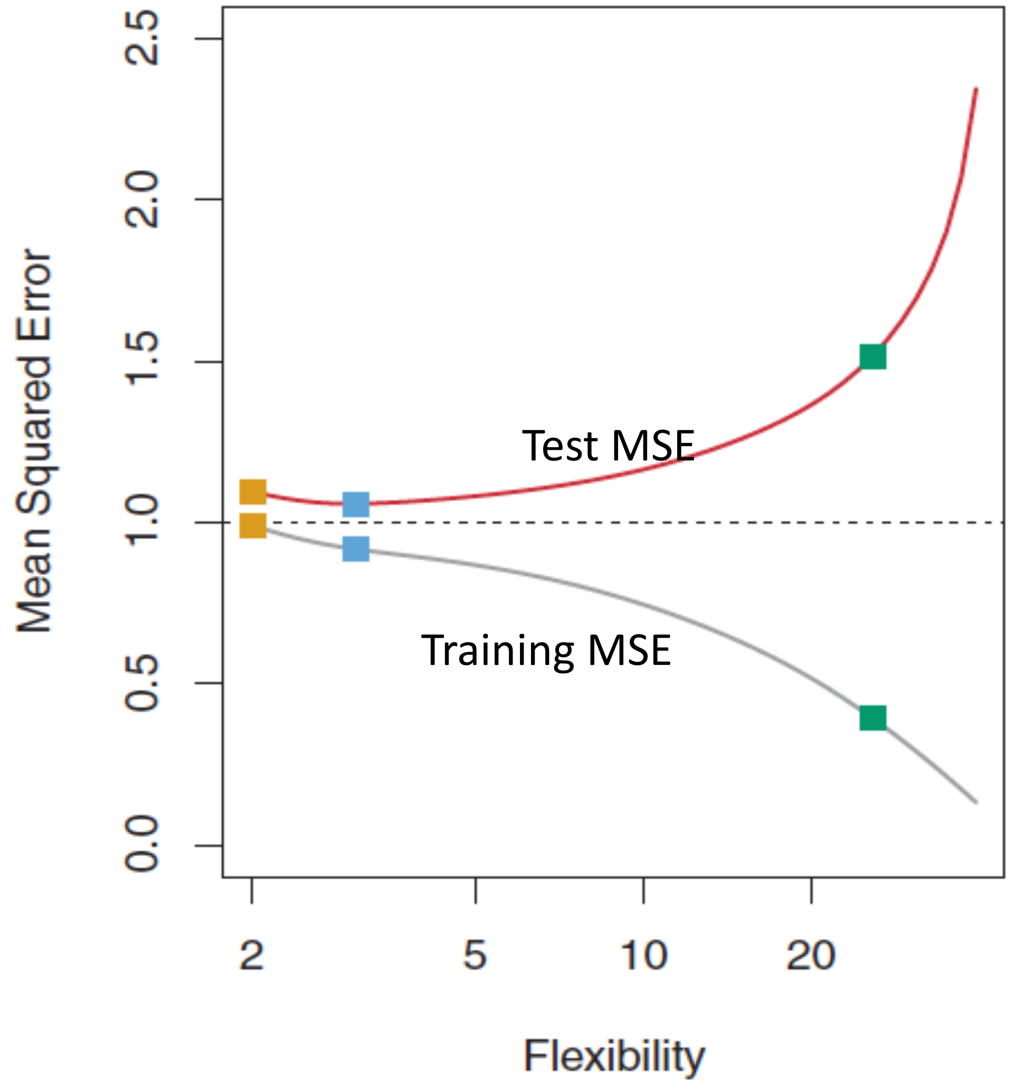




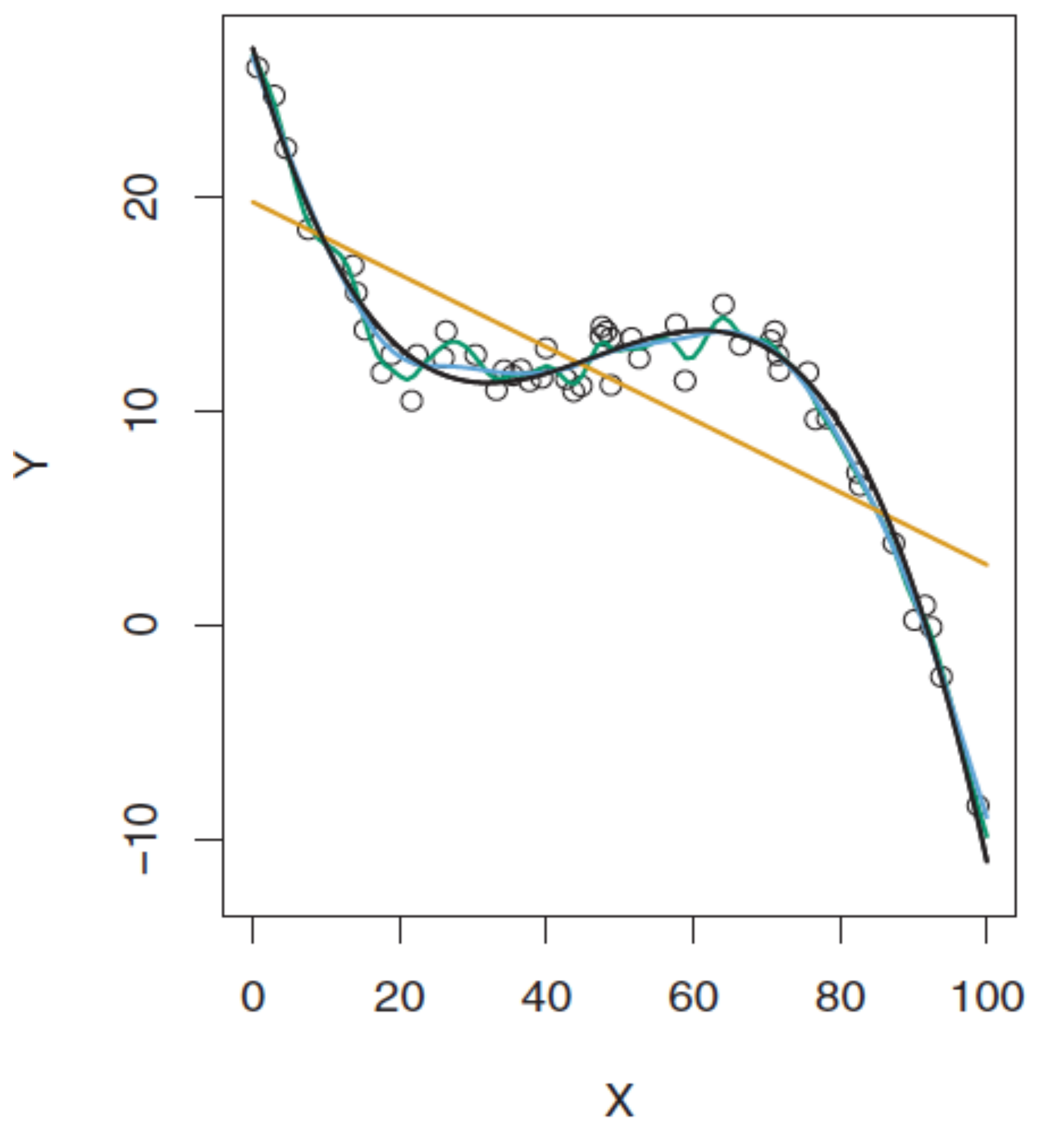


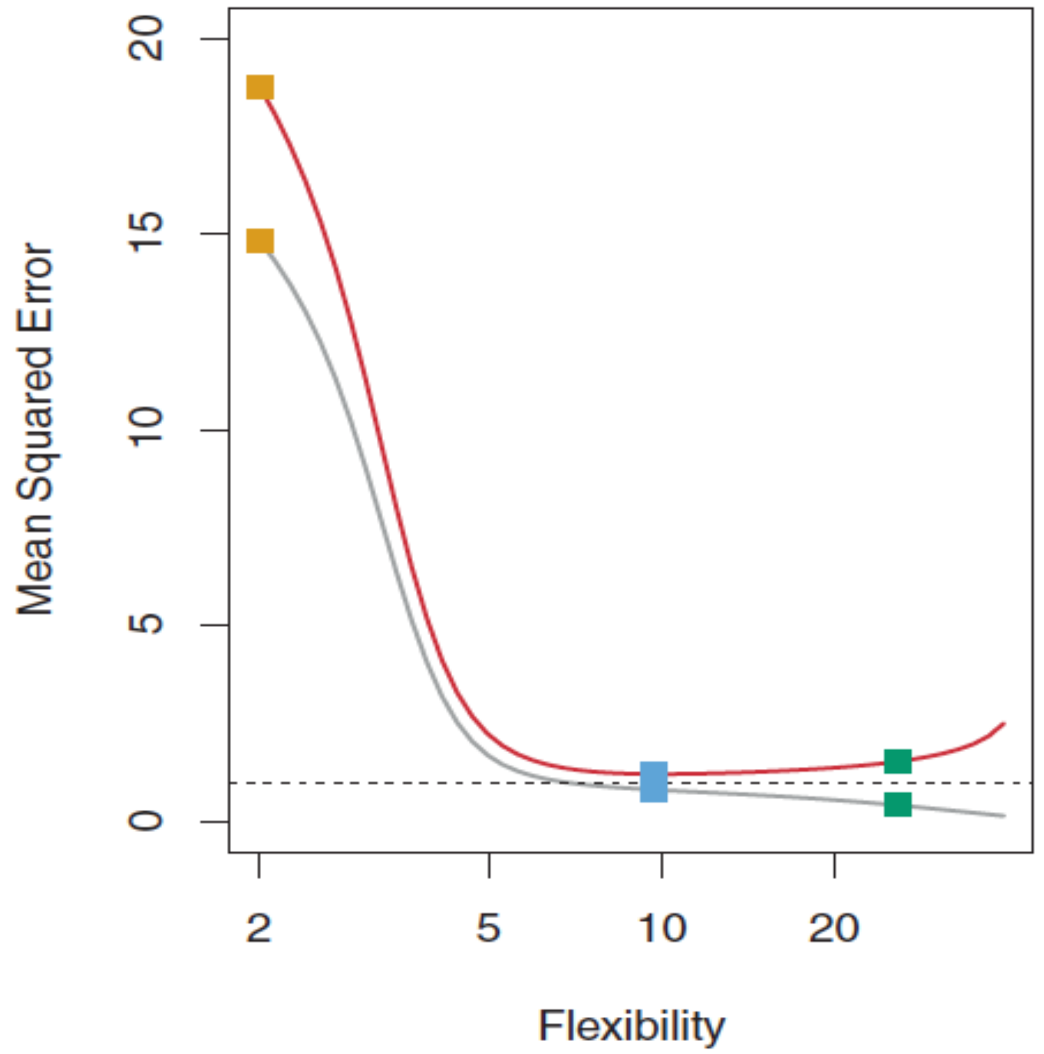
In this example, actual *f* in black (data points drawn from this function), three estimates of *f* with different flexibility: the linear regression line (orange), and two more flexible (smoothing spline) fits (blue and green)
The most flexible fit (green) is wiggly, which fits the training data very well, but has high test MSE





In this example, the linear model (least flexible) fits data very well, more flexibility results in more test MSE





The actual function is wiggly, so more flexible models do better.

Comments

- Flexibility level of the model with high accuracy (low test MSE) depends on the data set
- It is critical to have **sufficient and reliable data** to train your machine learning model
- Objective is to find the best model with lowest test MSE => Cross validation techniques (ch. 5 in textbook)

Bias-Variance Tradeoff

- The U-shape of the test MSE is a result of the bias-variance tradeoff
  - Two competing properties of statistical learning methods
- Test MSE is composed of sum of quantities: variance of $\hat{f}(x_0)$, bias of $\hat{f}(x_0)$, and variance of error term ϵ

$$E\left(y_0 - \hat{f}(x_0)\right)^2 = \mathrm{Var}(\hat{f}(x_0)) + [\underbrace{\mathrm{Bias}(\hat{f}(x_0))}_{E[\hat{f}(x_0)] - f(x_0)}]^2 + \mathrm{Var}(\epsilon)$$

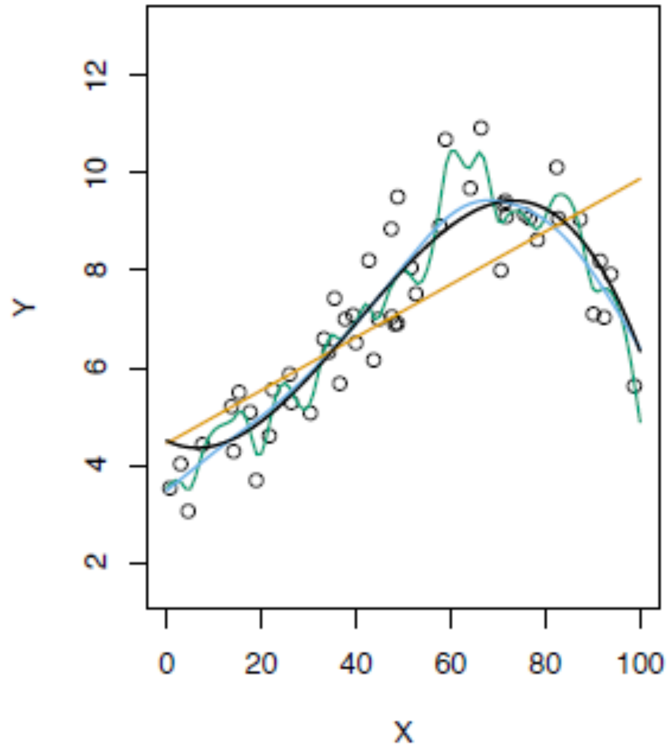
Expected value of MSE

  - The expected value of MSE: the average MSE if we repeatedly estimate *f* using large training set and test each at x_0, then average result over all test data
- Proof is beyond the scope of the course

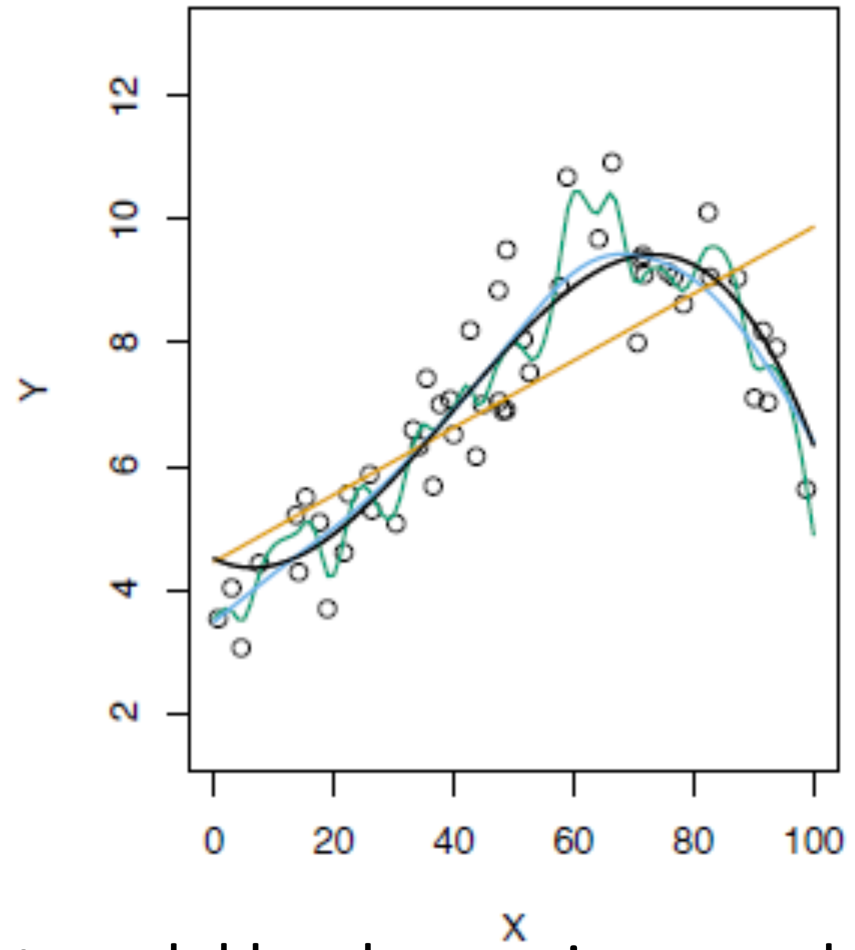
Bias-Variance Tradeoff

$$E\left(y_0 - \hat{f}(x_0)\right)^2 = \mathrm{Var}(\hat{f}(x_0)) + [\mathrm{Bias}(\hat{f}(x_0))]^2 + \mathrm{Var}(\epsilon)$$

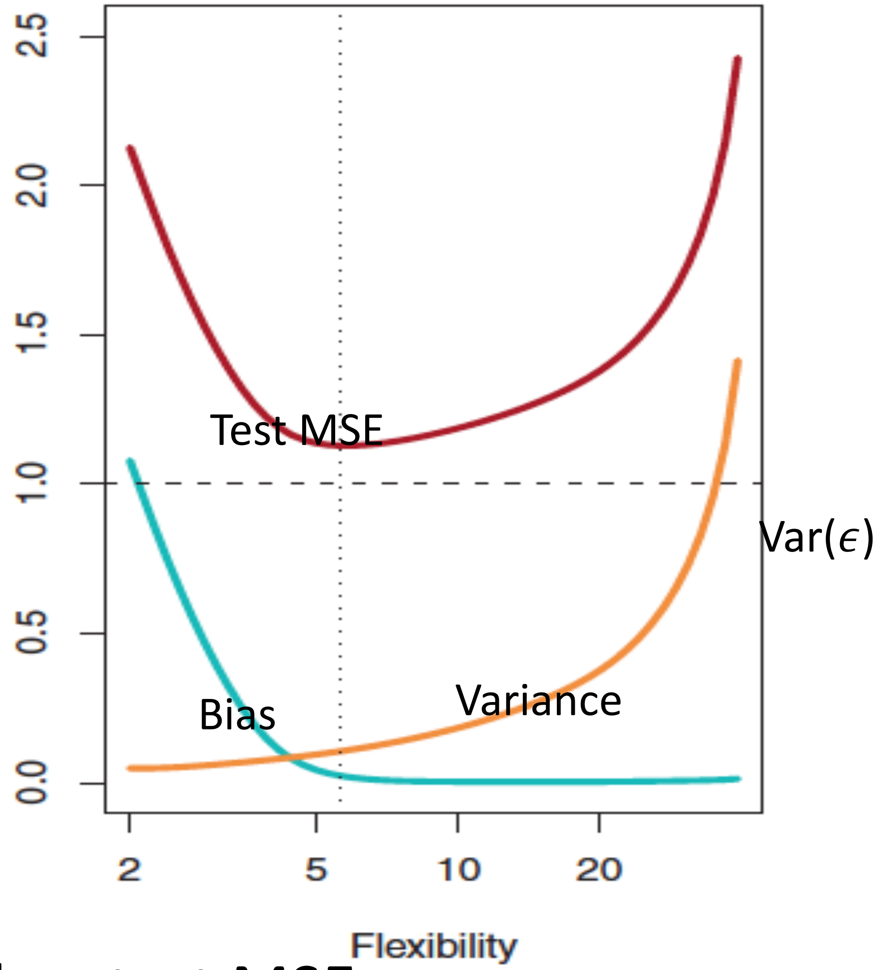
- Bias-variance trade off:
  - Variance: amount by which $\hat{f}$ changes if we made the estimation by different training set
    - High variance: small changes in training data leads to large changes in $\hat{f}$
    - **More flexible models typically have high variance**
    - E.g. Green curve in the figure has high variance, while orange curve has low variance
  - Bias: Errors from approximating real-life problems by a simpler model
    - How far the estimated model is from the actual function
    - More flexible models can result in less bias



Test MSE, Bias and Variance



Test MSE = Bias + model variance + error variance



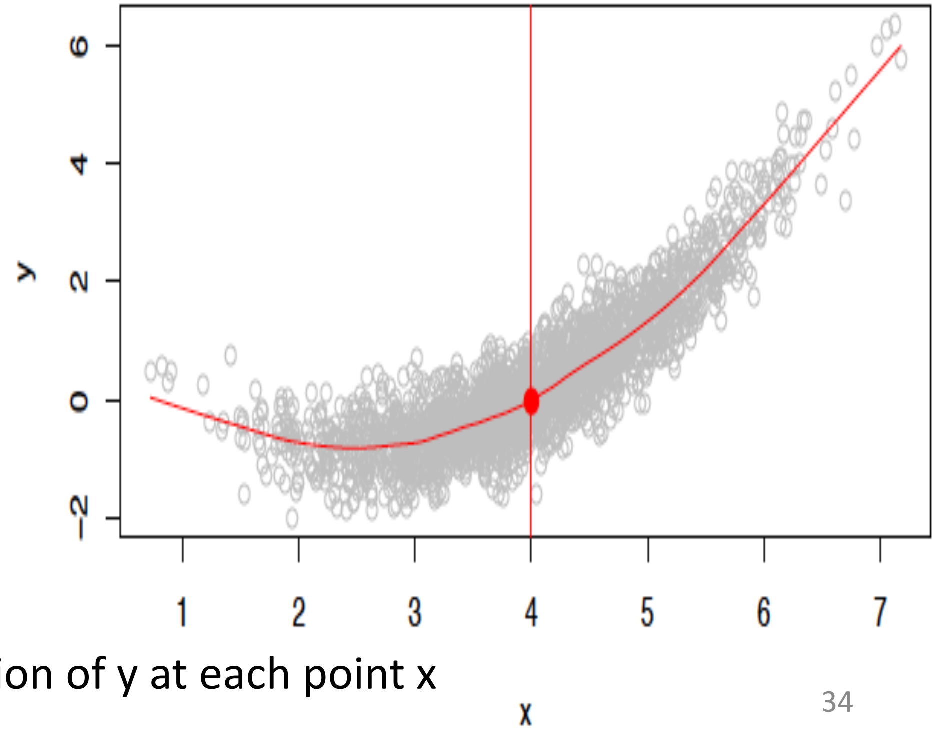
- Best model has low variance and low bias ➔ low test MSE
- **Flexible models (complex models) have low bias but high variance**
- **Less flexible models have high bias and low variance**

Reducible and Irreducible Error

- Test MSE

$$E\left(y_0 - \hat{f}(x_0)\right)^2 = \underbrace{\mathrm{Var}(\hat{f}(x_0)) + [\mathrm{Bias}(\hat{f}(x_0))]^2}_{\text{Reducible error}} + \underbrace{\mathrm{Var}(\epsilon)}_{\text{Irreducible}}$$

- Reducible error: can be improved by better estimate of *f(x)*
- Irreducible error: cannot be reduced with better estimate of *f(x)*



ϵ= Y - f(x) is the irreducible error

Measurement errors or due to distribution of y at each point x

Classification Setting

- The response in is qualitative
  - Email spam or not spam:
- Build classifier that assigns class label to a future unlabeled observation
  - y_i belongs to a finite set of possible classes: $y_i \in C, C = \{1,2,..,m\}$
- To assess the model accuracy, we typically evaluate the **error rate**
  - **Accuracy =1 – error rate**
- $\hat{y}_o$= $\hat{f}(x_0)$ is the predicted output class
- Test error rate associated with **test** observations (x_0, y_0):

$$Average(I(y_o \neq \hat{y}_o))$$

$I(y_o \neq \hat{y}_o)$ is indicator variable that is equal to 1 when $y_o \neq \hat{y}_o$, and zero when $y_o = \hat{y}_o$

Classification Setting

- The error rate is minimized by a simple classifier, called **Bayes classifier**
- Bayes classifier assigns each observation to the **most likely class given the feature values**.
  - Assign x_0 to class *j* that has **largest** $Pr\,(Y= j\,|\,X= x_0)$
  - $Pr\,(Y= j\,|\,X= x_0)$ *is the conditional probability*
    - *Example, spam email: Y=1 (spam), Y=2 (not spam)*
      - *Pr(Y=1) is the probability that Y is spam email*
      - *Pr(Y=1|X) is the conditional probability that Y is spam given features of an email, e.g. given it contains the word "free", size of email*

Bayes Classifier

Figure shows two features of 100 simulated observations of two classes

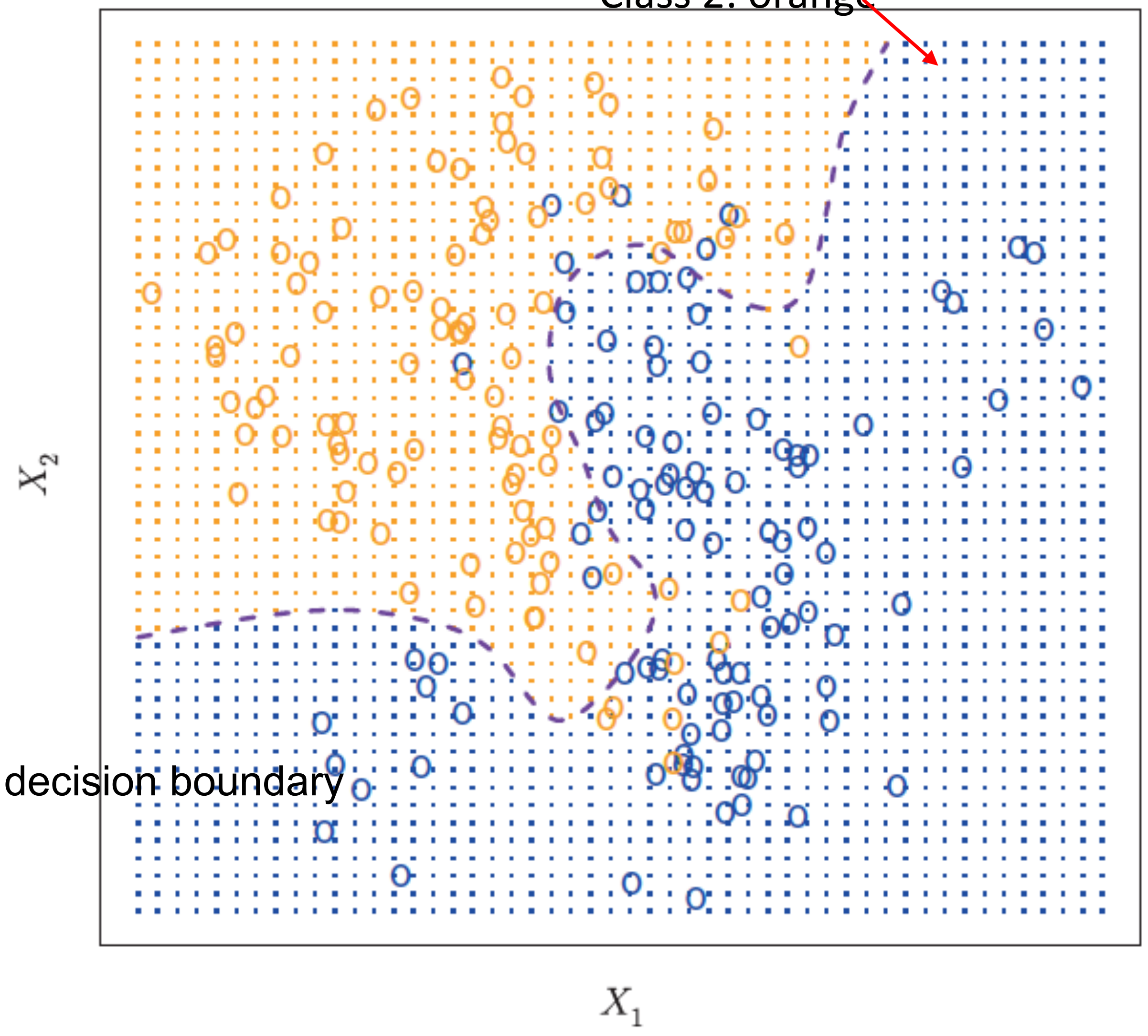
- For each value of X_1 and X_2 there is a probability to be in one of the 2 classes
  - Here the **conditional distribution is known**
- The dashed line is called **decision boundary**, where the probability is exactly 50%

p(y=orange|x)=p(y=blue|x) when on the decision boundary

- Decisions are based on this boundary
  - Each side of the decision boundary belongs to a different class

Feature Space 2-D

Class 1: blue
Class 2: orange



Simulated data 100 observations

K-Nearest Neighbors

- Typically we do not have the distribution and it is hard to get conditional probabilities
  - We have few points, if any, at each X

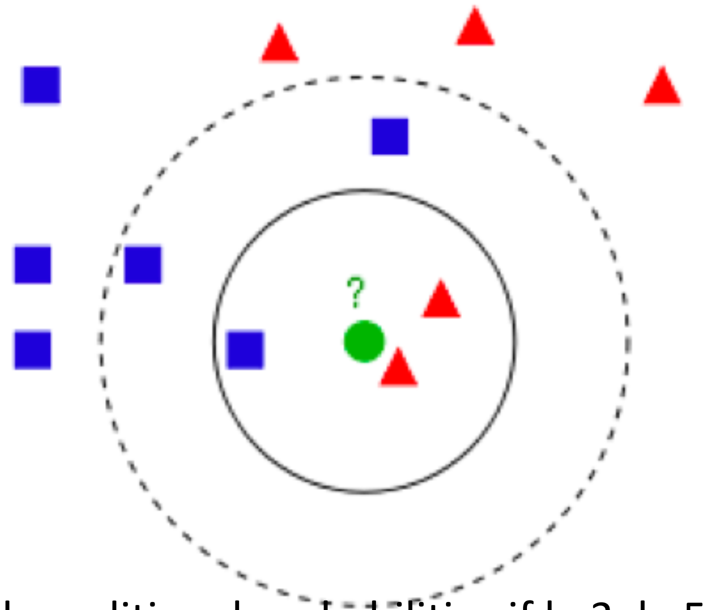
- Many methods tries to **estimate the conditional distribution**, one of these methods is K-nearest neighbor classifier (abbreviated as K-NN or KNN): .

K-Nearest Neighbors

- K-nearest neighbor (KNN):
  - Define a positive integer K
  - For each test observation x_0 , identify **K points in the training data that are closest to $\mathbf{x_0}$** referred to as $\mathcal{N}_0$
  - **Estimate the conditional probability** for class j as **fraction of points in $\mathcal{N}_0$ whose response values equal to j**

$$\Pr(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j)$$

  - Then assign to the class with largest conditional probability
  - Decision depends on the choice of K



k=3, p(red|x)=2/3, p(blue|x)=1/3----red
k=5, p(red|x)=2/5, p(blue|x)=3/5---blue

What is the estimated conditional probabilities if k=3, k=5?

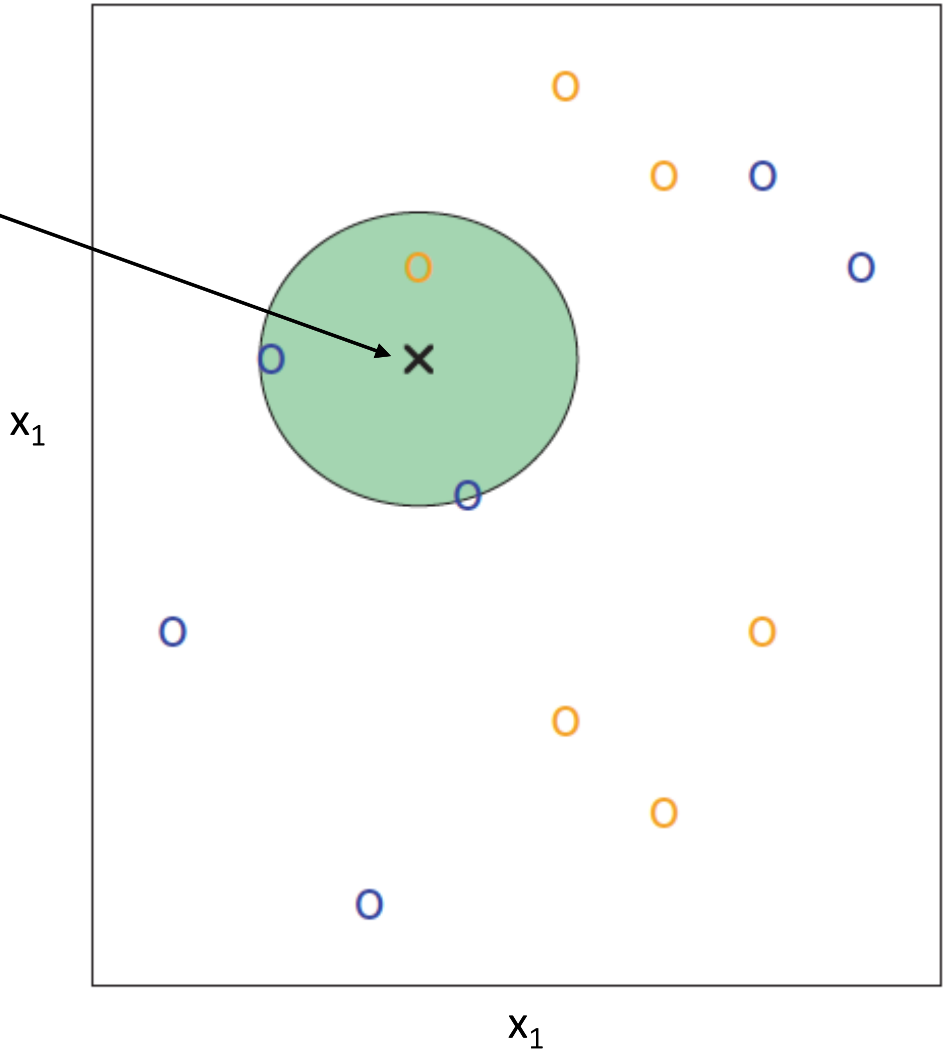
K-Nearest Neighbors - Simplified

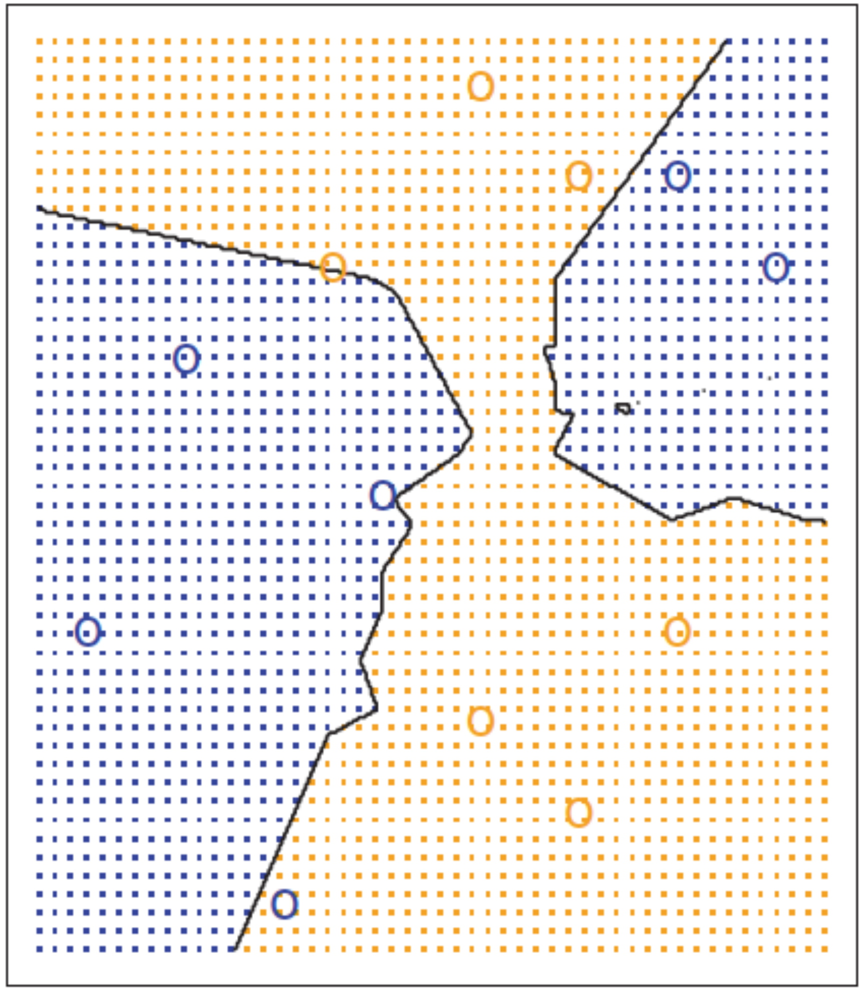
- For any given test data point, we find the **K closest neighbors** to this point in the **training data**, and examine their corresponding class (y).

- Assign the data point to the class from which the **majority of neighbors** belong

K= 3, 2-D Feature Space Example

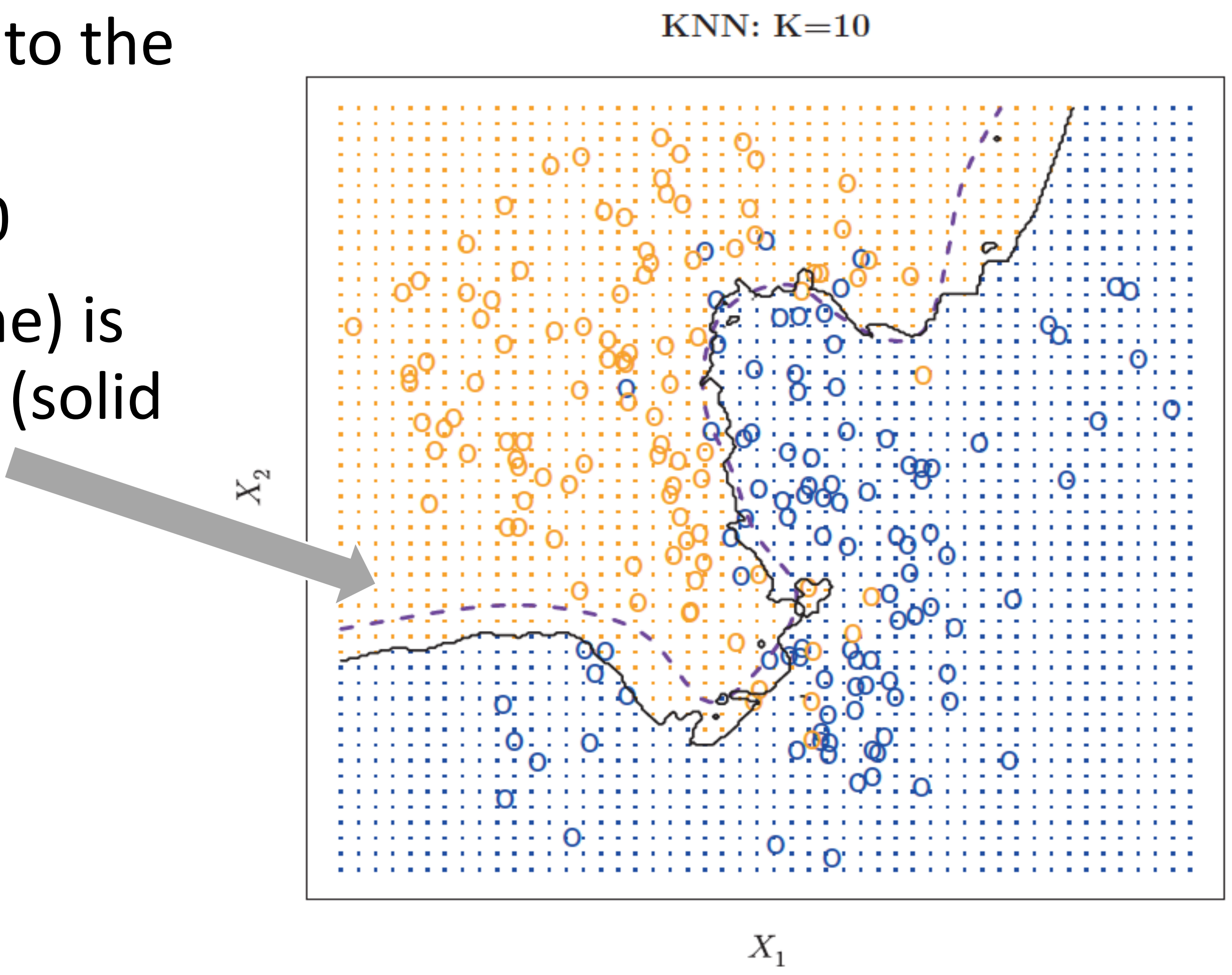
Within the 3 nearest neighbors, two of them belong to the blue class (majority). Thus, we classify x as blue





2 classes: blue and orange circle

- KNN is simple, sometimes it is close to the optimal Bayes classifier
- Fig. shows an example of using K=10
- Bayes decision boundary (dashed line) is close to the KNN decision boundary (solid line)

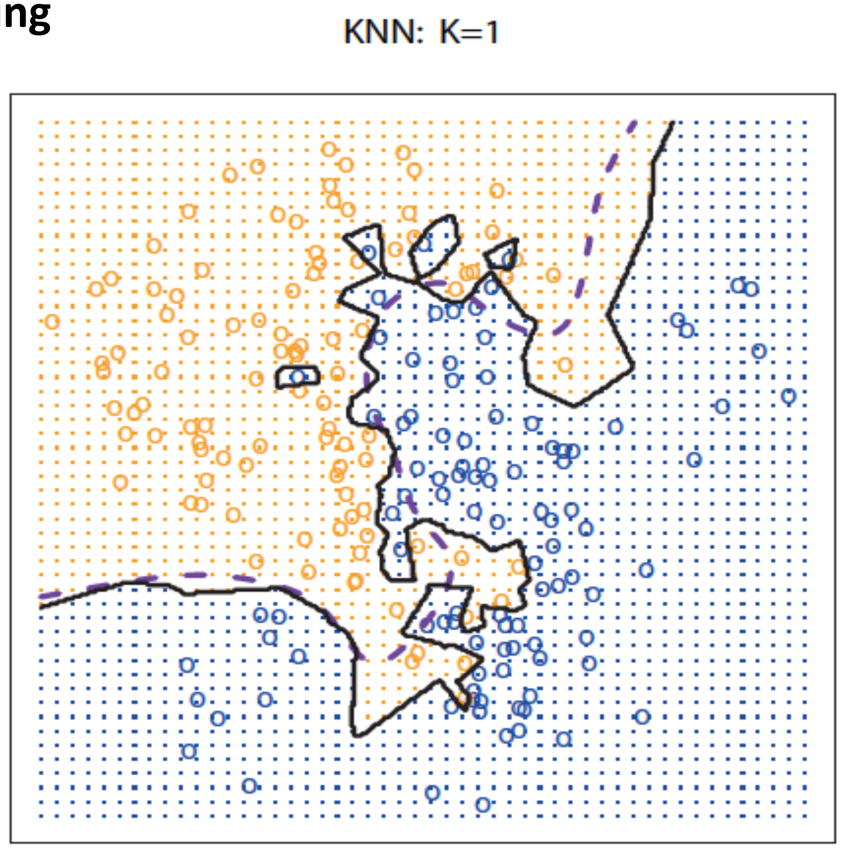


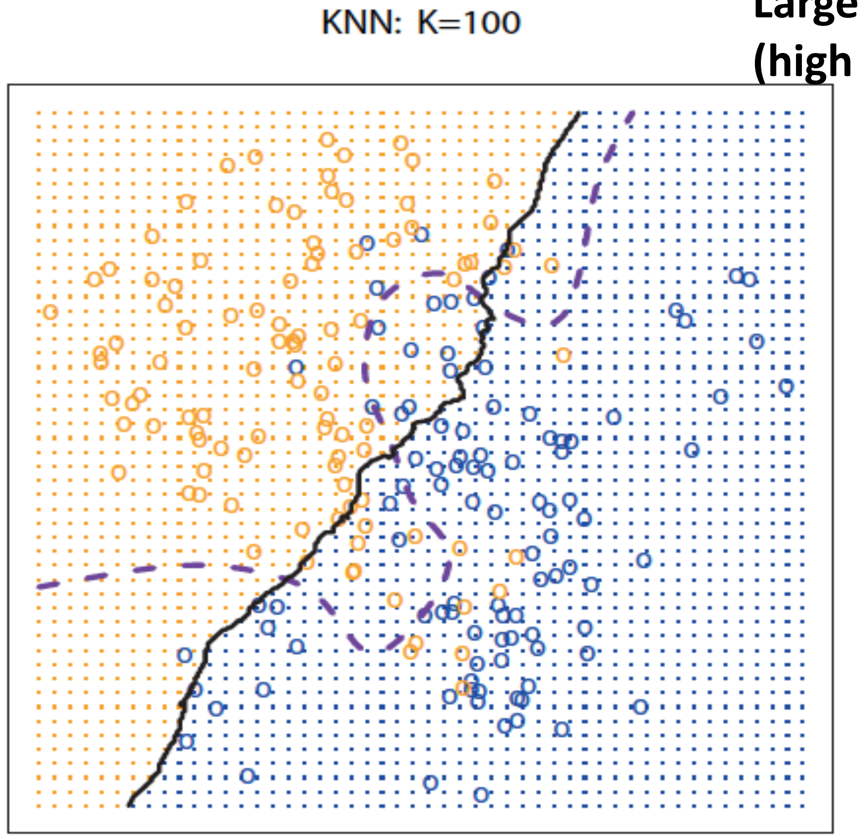
K-Nearest Neighbors

- Choice of K has huge impact on the performance
  - Bias-variance trade-off applies

K=1 => overfitting (high variance)

Large K =>underfitting (high bias)

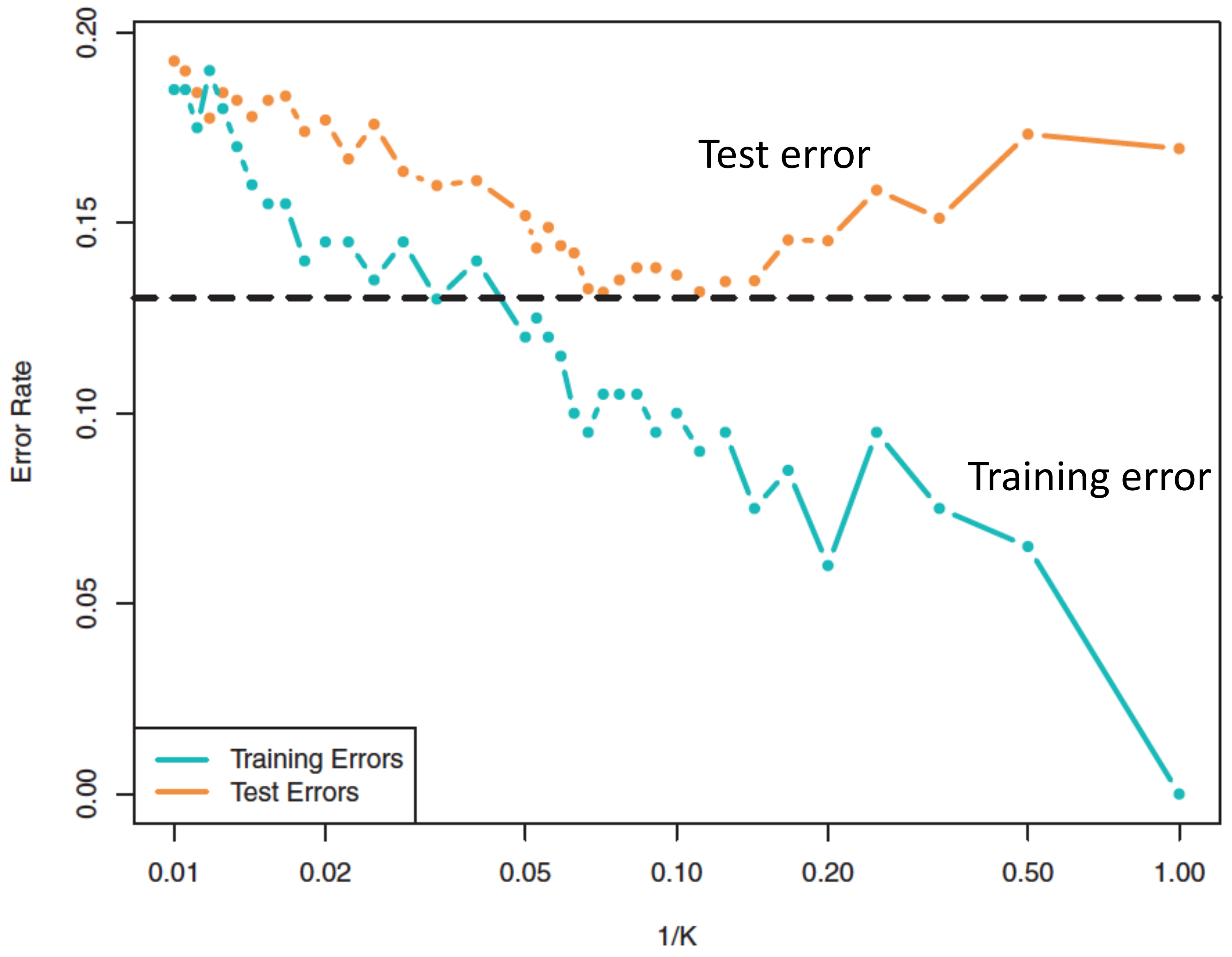




Dashed line: Bayes decision boundary

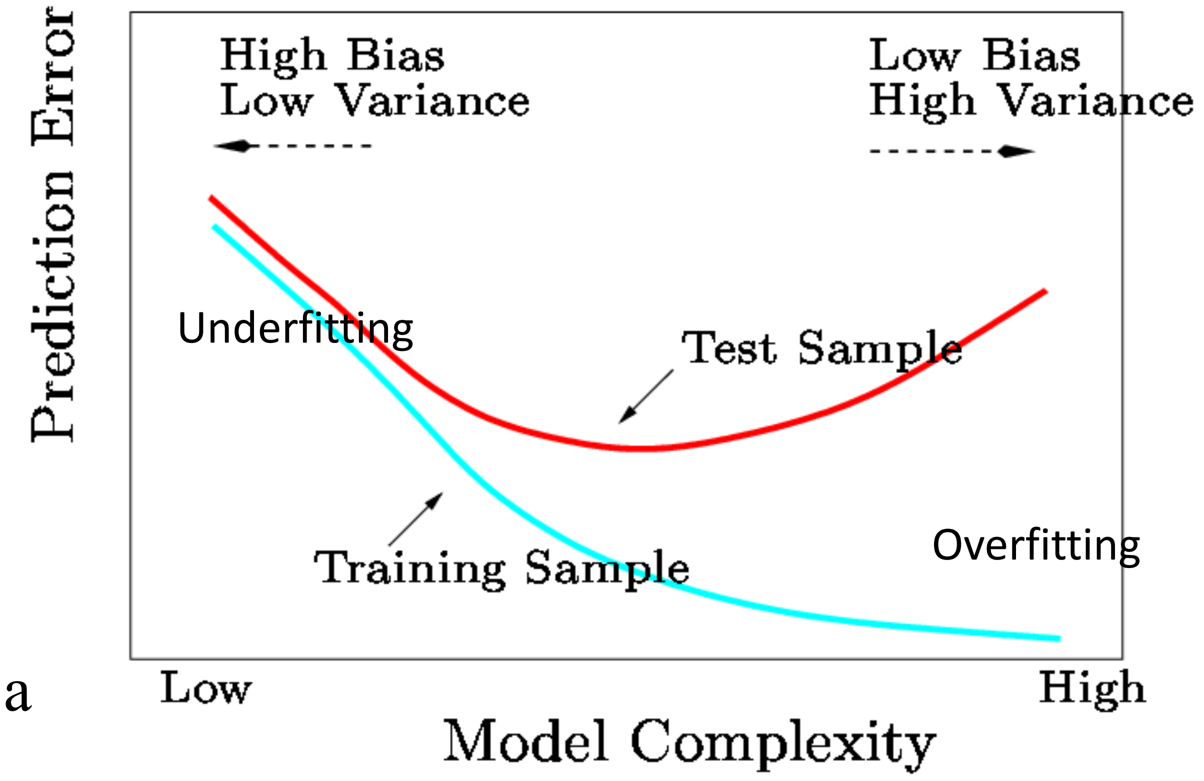
Different K different decision boundaries

- Same principles apply to classification problems



Trade-offs

- Prediction accuracy versus model complexity (flexibility)
  - Bias-variance trade-off
- Good fit versus over-fit or under-fit



Keep this picture in mind when choosing a learning method.

More flexible/complicated model is not always better!