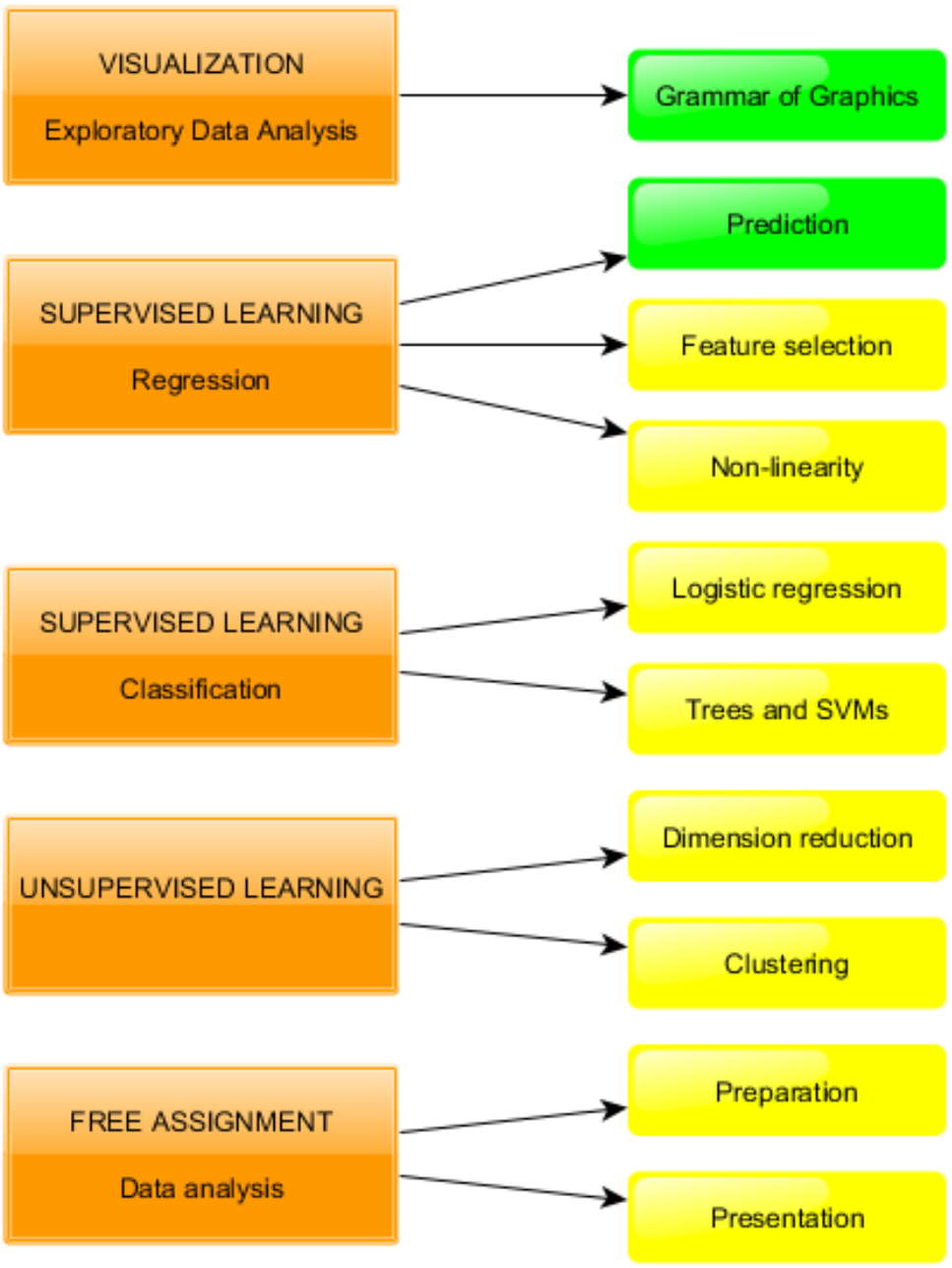
Supervised Learning: Regression

Prediction

Program


VISUALIZATION
Exploratory Data Analysis
Grammar of Graphics
SUPERVISED LEARNING
Regression
Prediction
Feature selection
Non-linearity
SUPERVISED LEARNING
Classification
Logistic regression
Trees and SVMs
UNSUPERVISED LEARNING
Dimension reduction
Clustering
FREE ASSIGNMENT
Data analysis
Preparation
Presentation

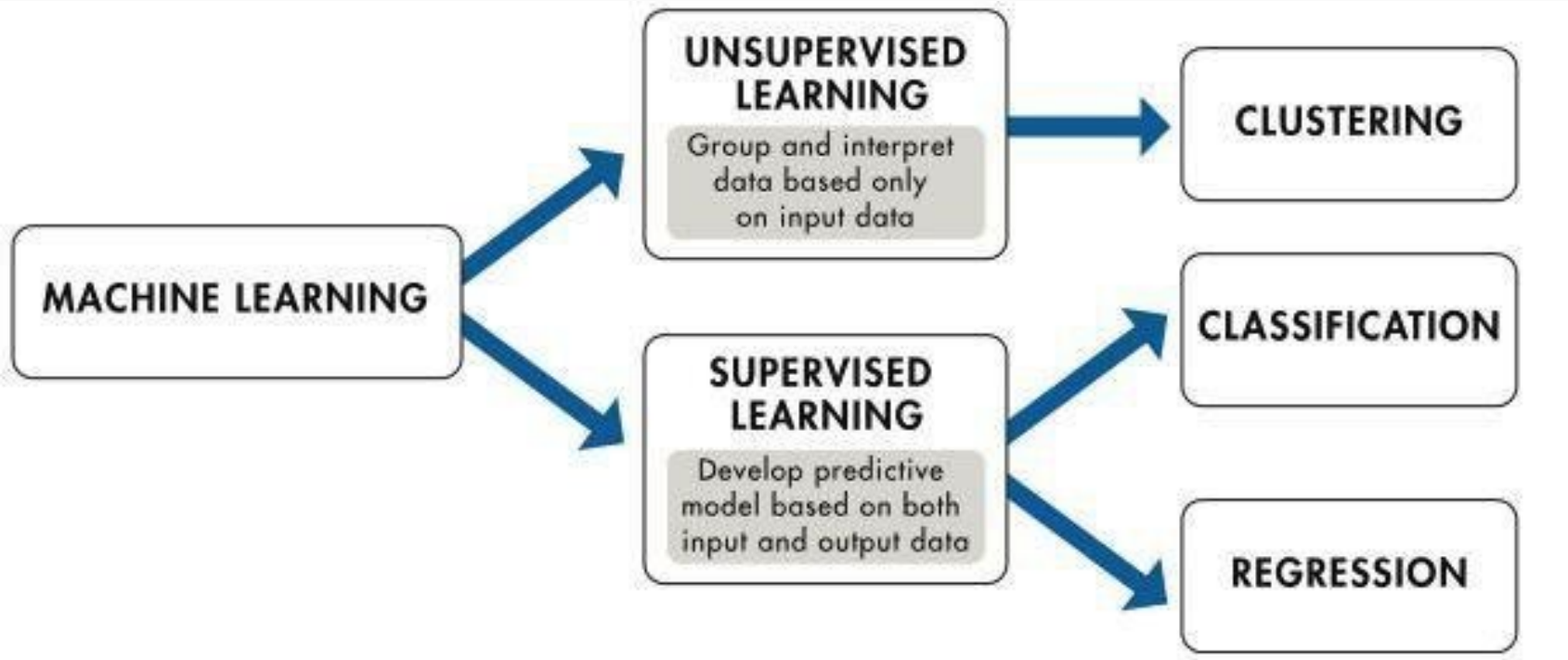
Content

1. Statistical learning
2. Supervised learning
3. Prediction models
4. Bias-variance tradeoff
5. Training-validation-test set paradigm (or "Train/dev/test")
6. The `caret` package

Statistical learning

Supervised and unsupervised

- difference is in presence of output data



Machine learning

Not entirely true:

- there is a systematic approach

Supervised learning

Statistical inference vs supervised learning

Statistical inference

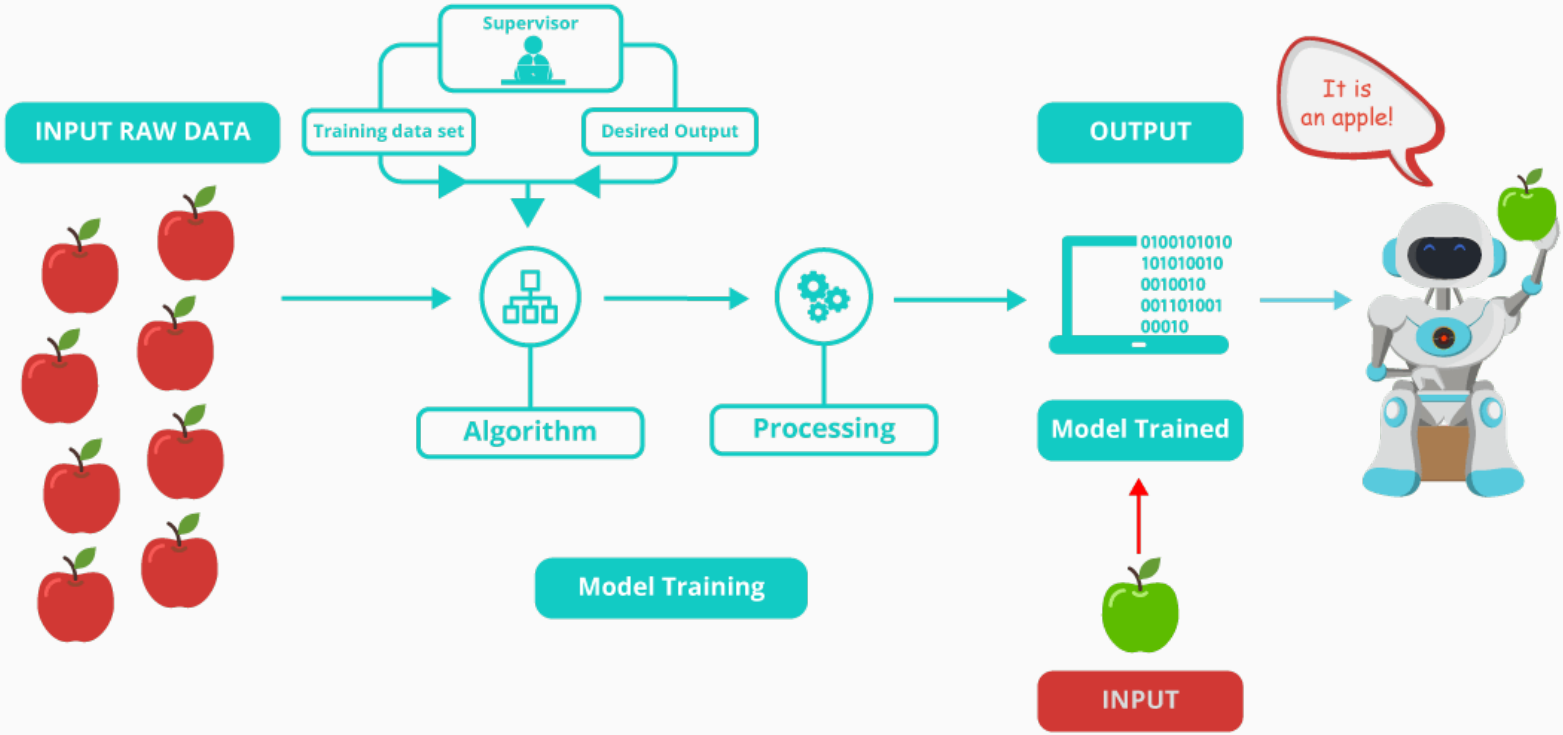
- fit model to entire data set
- find model that minimizes MSE
- aim is interpretation parameter estimates

Supervised learning

- train model on training data
- find model that minimizes MSE on test data
- aim is prediction on new data

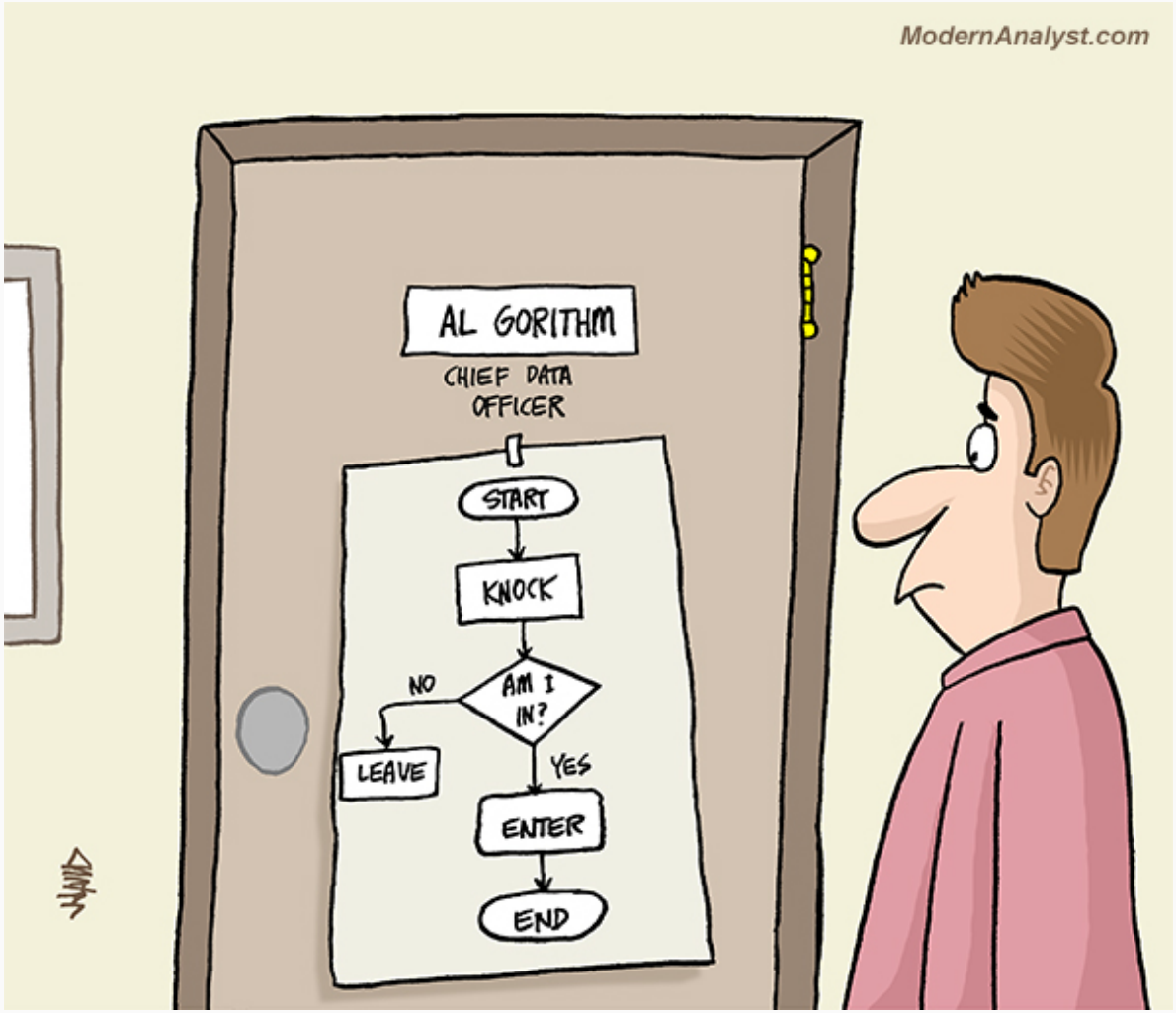
Presence outcome variable

- predict outcome variable (supervisor)
- train model using some algorithm



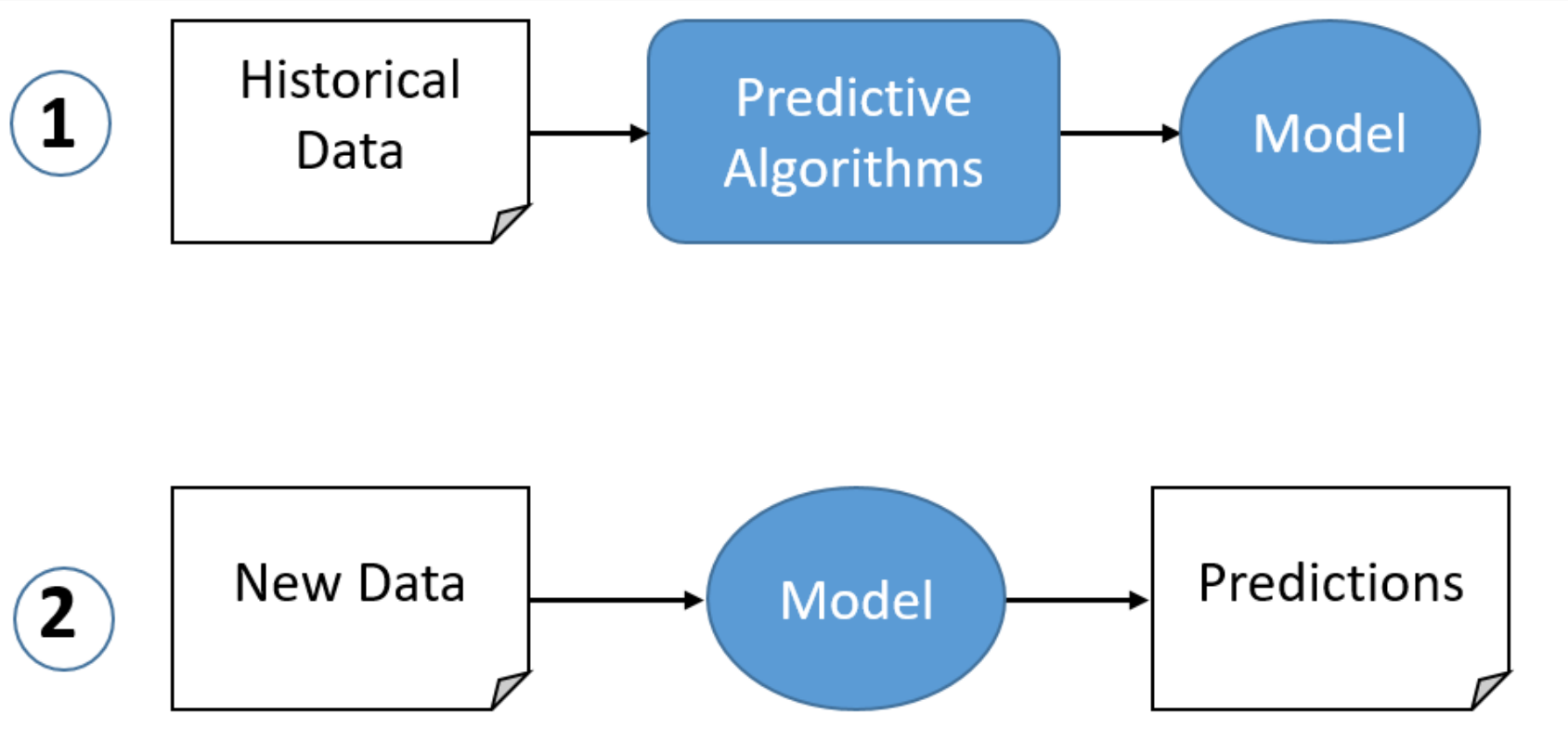
What's an algorithm?

- a set of instructions to solve a problem



What's model training?

- distinction between training data and new data
- aim is to predict outcome in new data set



What's a model?

Mathematical representation of a theory

Model relating output y to input x

$$y = f(x) + \epsilon$$

- y outcome
- x predictor(s)
- $f(x)$ prediction function
- ϵ irreducible error

Choice of $f(x)$

Many competing models

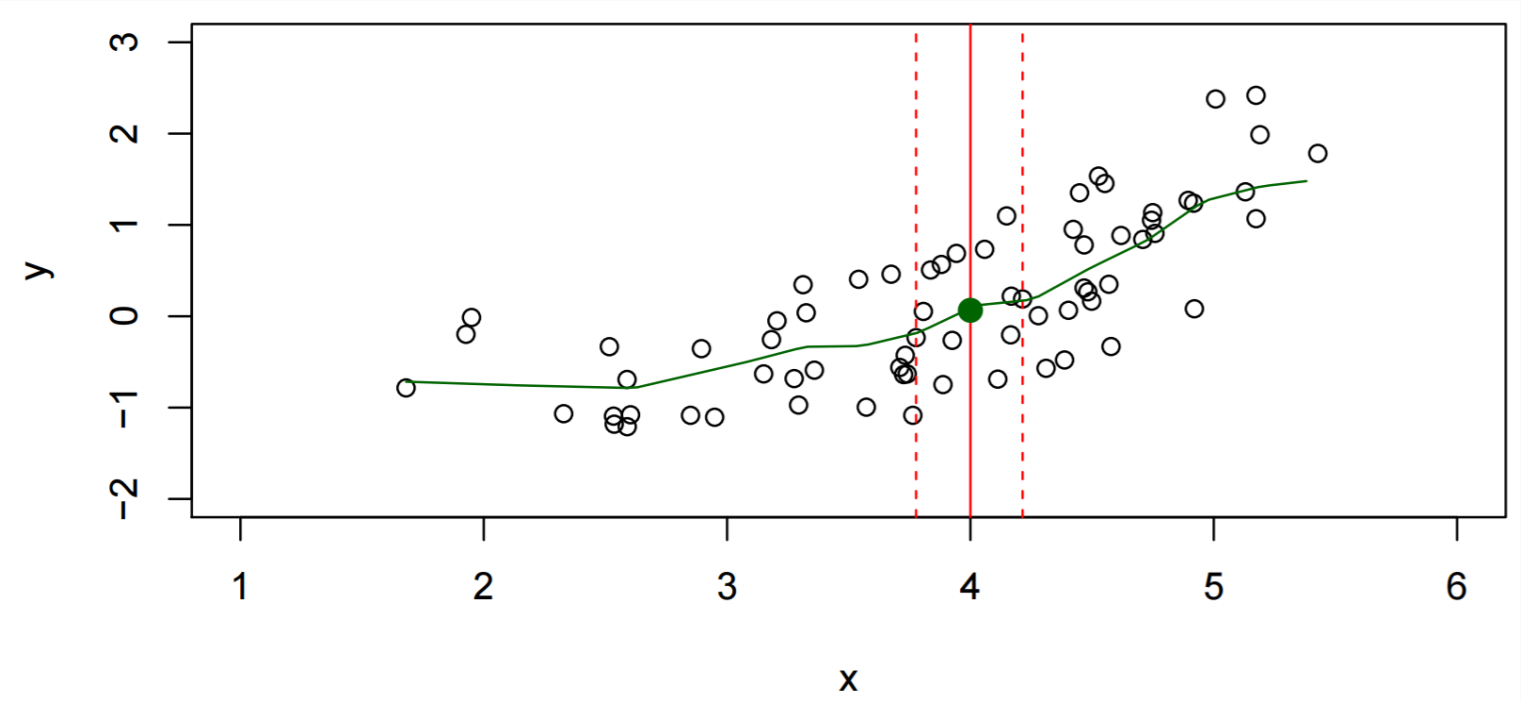
- k-nearest neighbors
- linear regression
- nonlinear regression
- tree-based methods
- support vector machines
- neural networks
- etc.

These models vary in complexity and interpretability

k-nearest neighbors (KNN)

Nonparametric model (distribution y unspecified)

$$f(x) = \frac{1}{k} \sum_{i=1}^{k} (y | x_i \in \text{neighborhood of } 4)$$

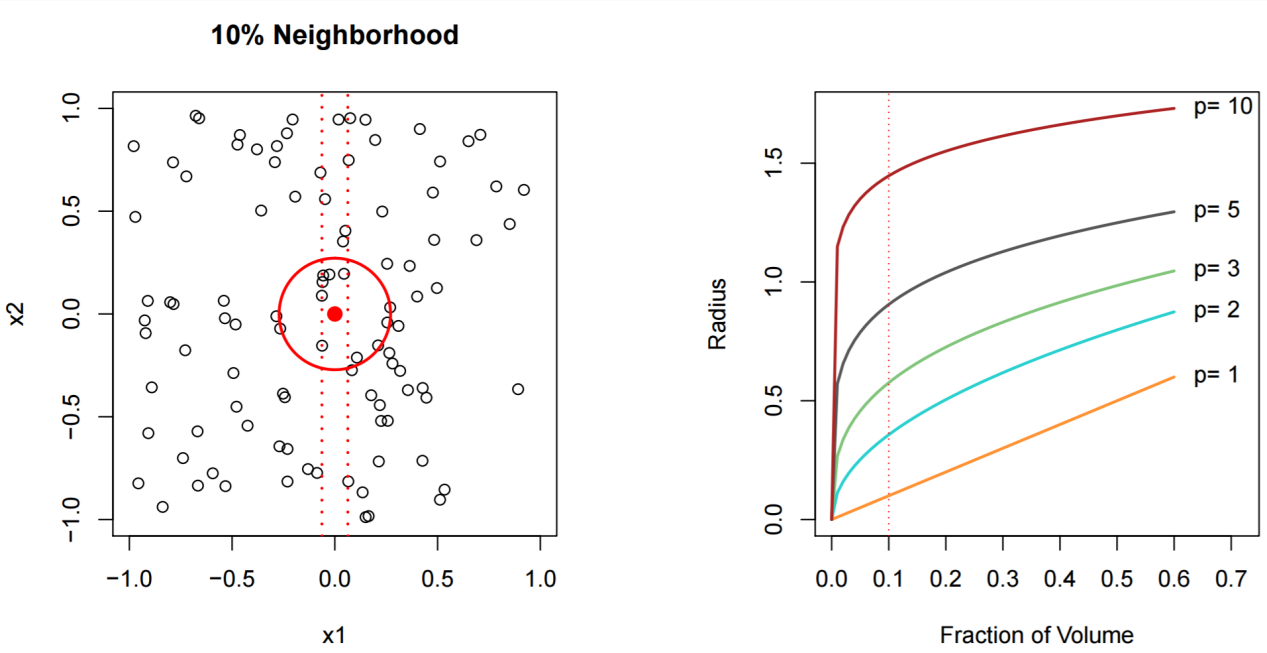


Prediction for $x = 4$ is mean y of k nearest data points

Curse of dimensionality

Limited to low-dimensional data

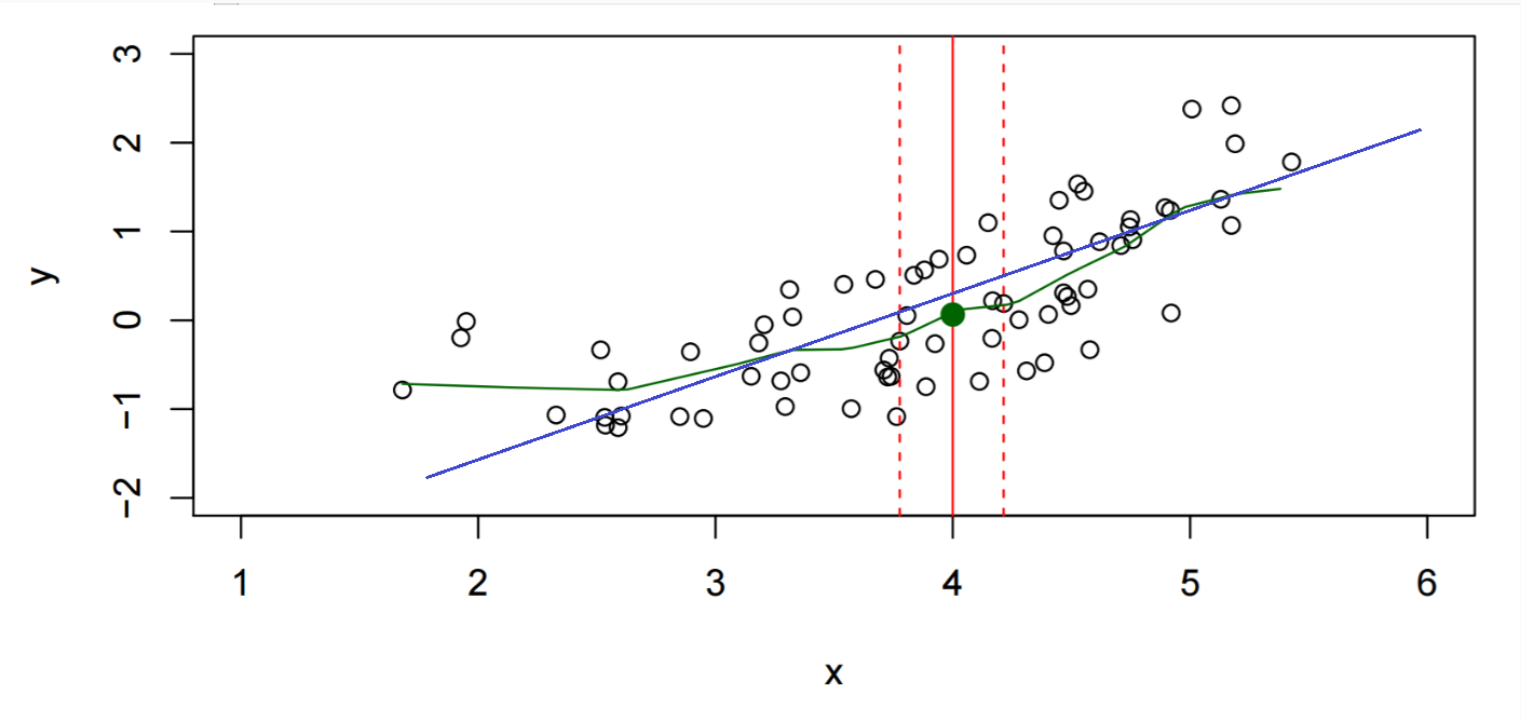
- KNN breaks down with increasing number of predictors
- distance between points increases and neighborhood shrinks



Linear regression

Parametric model (linear relationship y, x and normal ϵ)

$$f(x) = \beta_0 + \beta x, \qquad \epsilon \sim N(0, \sigma_\epsilon^2)$$



- distance between points no problem for drawing straight line

More complex models

Polynomial models allows for non-linearity

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \dots$$

- model is linear combination of the polynomials of x

Estimation problem

Usually do not know $f(x)$, it could be

- a linear function
- a polynomial function
- or many other functional forms (logarithm, exponent, etc.)

How do we find $\hat{f}(x)$ that comes closest to $f(x)$?

- we estimate it! But how?

Mean Squared Error (MSE)

Choose J competing functions $\hat{f}_j(x)$ as estimates of $f(x)$

- fit the models to the data

$$\hat{y}_j = \hat{f}_j(x) + \epsilon_j, \quad j = 1, 2, \ldots, J$$

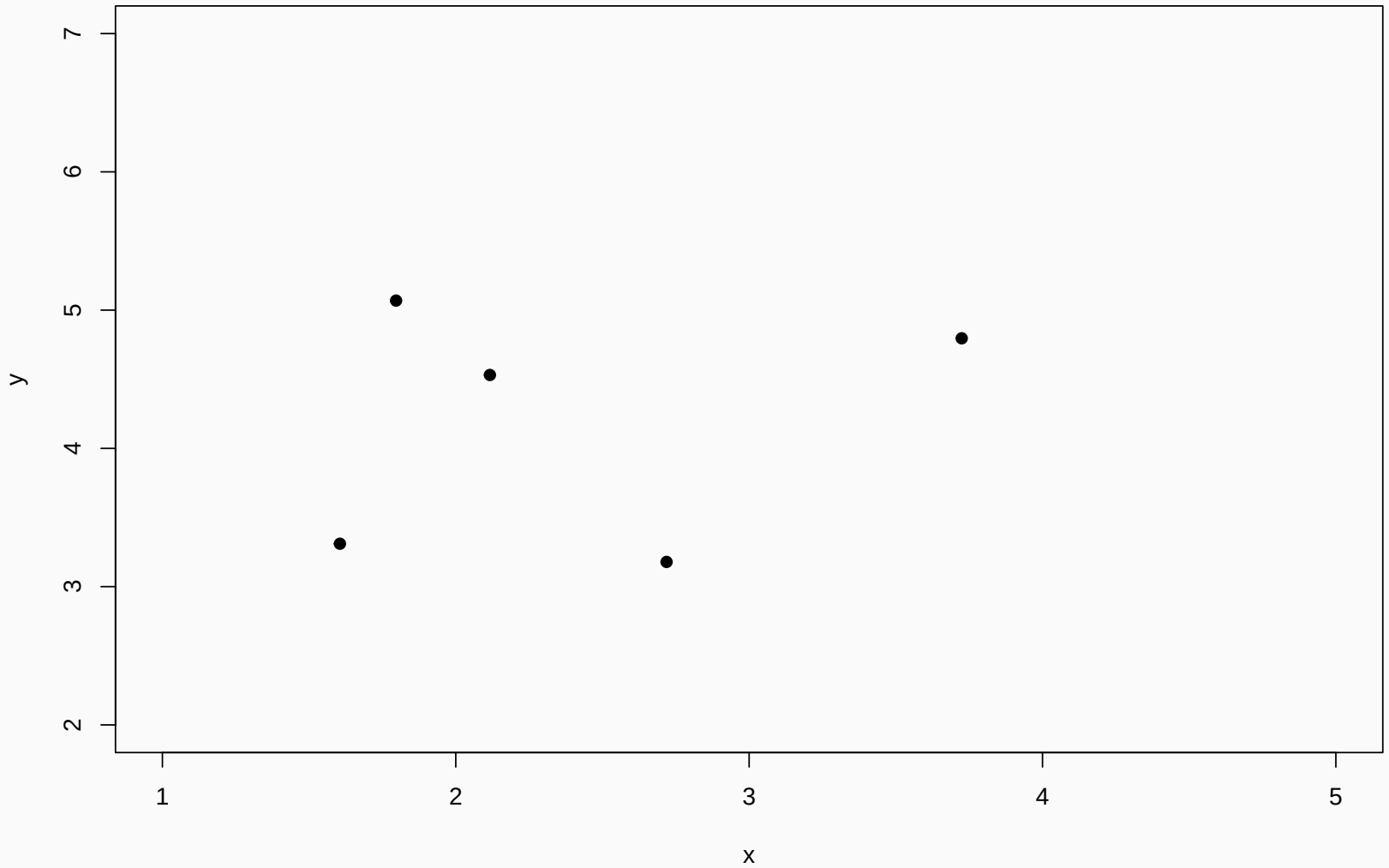
- choose j with the smallest MSE (error variance)

$$MSE_j = \frac{1}{n} \sum_{i=1}^{n} \sigma_{ij}^2$$

TRUE?

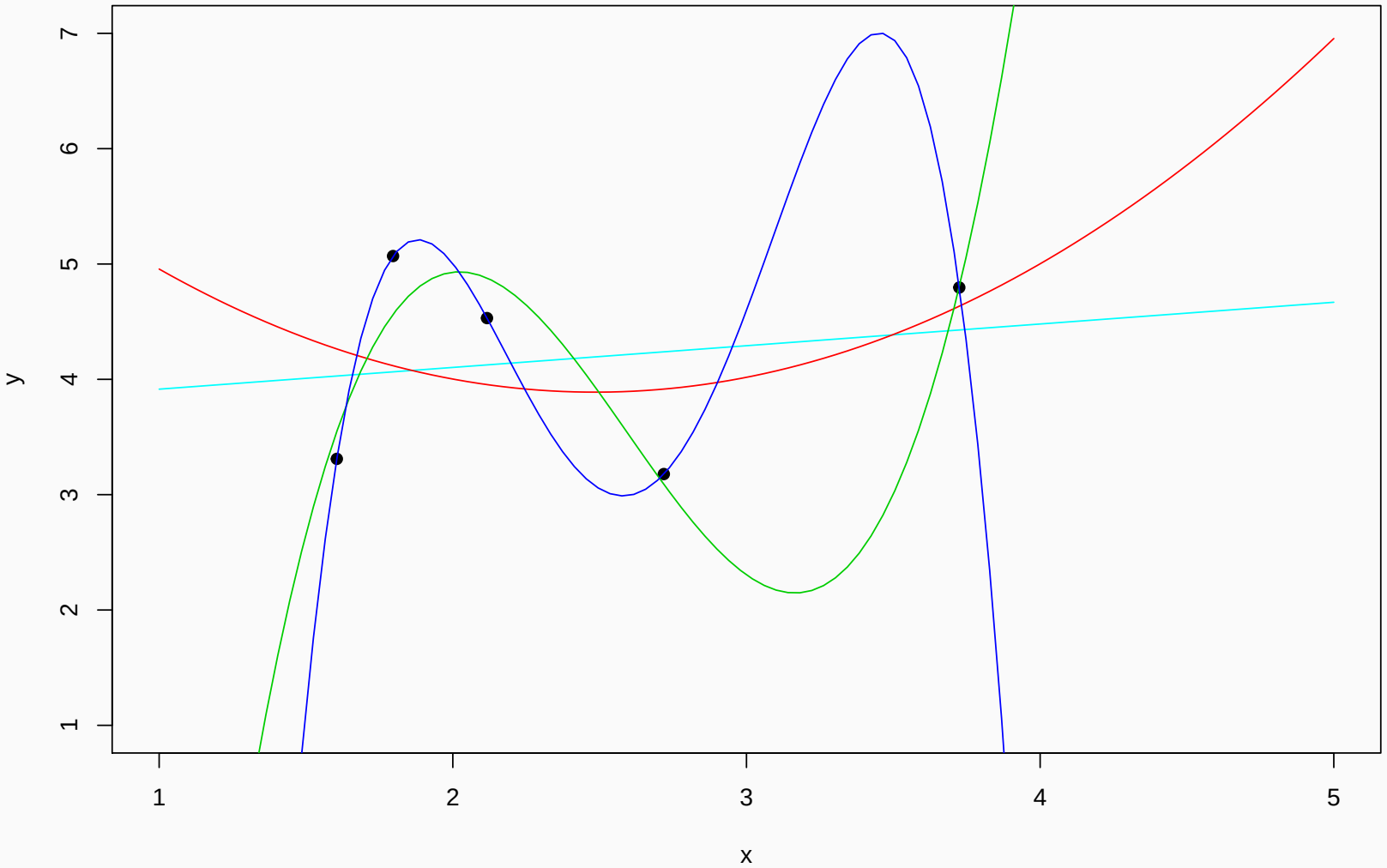
Example with data generated from $f(x)$

What is the best choice $\hat{f}(x)$ given these data points?



Predicted values for y

Regression lines for linear, quadratic, cubic and quartic models



MSE

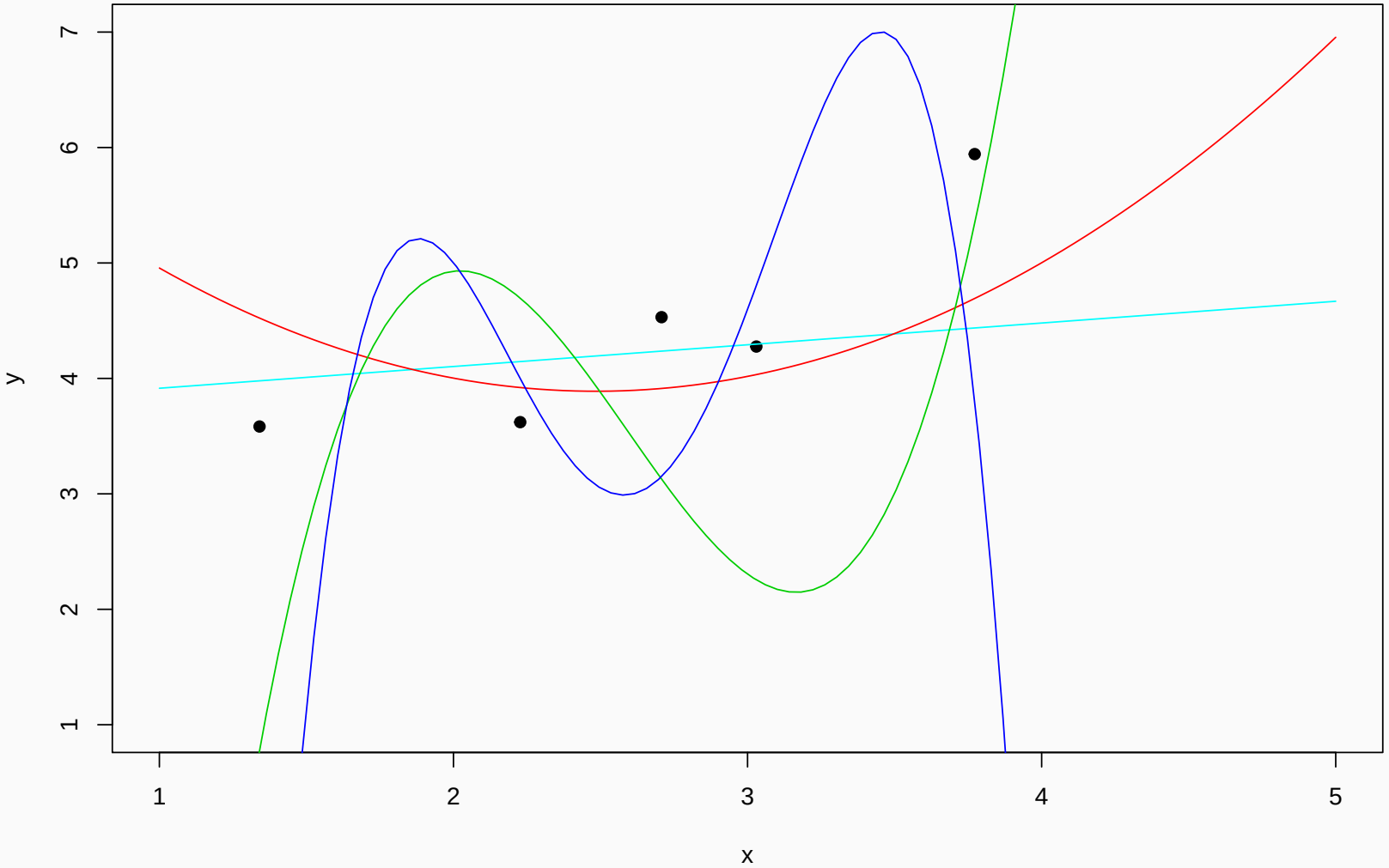
Comparison of the MSE of the fitted models

Fitted models	MSE
linear	0.5896
quadratic	0.5428
cubic	0.0881
quartic	0

Which model do you choose, and why????

Five new data points from $f(x)$

- What are the MSE's for these five new data points?



MSE for new data points

MSEs of the new data points show different picture!

Fitted models	MSE
linear	0.557
quadratic	0.595
cubic	3.086
quartic	14.285

Which model do you choose now, and why????

True $f(x)$

Model from which the data point were generated is

$$f(x) = 2 + 2x + \epsilon, \qquad \epsilon \sim N(0, 1)$$

so the true model is the linear model!!!

Over- and underfitting

The more complex the model, the better the fit

- complex models adapt to nonlinear patterns in the data
- and thus reduce the MSE

Too much complexity results in poor predictions

- complex models also adapt to random error
- in that case new data points are poorly predicted

Too little complexity results in poor predictions

- simple model do not adapt to nonlinear patterns in the data
- and predict poorly when the true model has nonlinearity

Bias-variance tradeoff

Bias and variance

Unbiased model

- model that predicts correctly on average

Model with high variance

- model that easily adapts to accidental patterns

In the five-point example, which models are unbiased, and which have high variance?

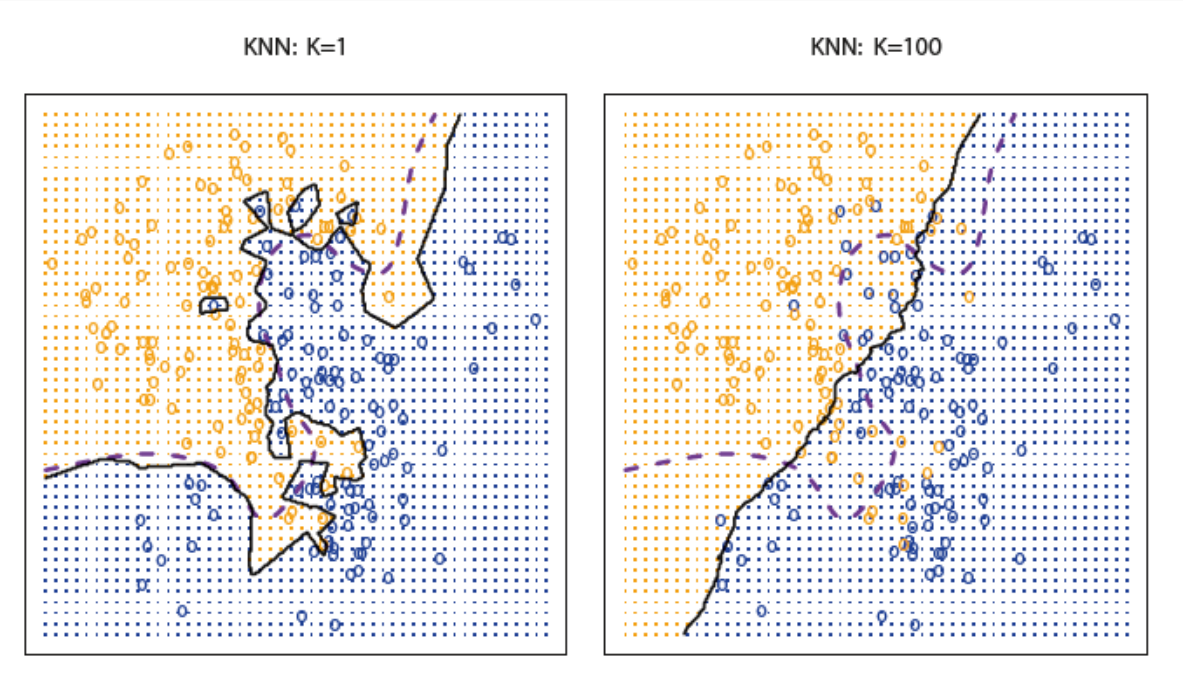
Some examples

Bias and variance in regression models

$f(x)$	$\hat{f}(x)$	bias	variance
$\beta_0 + \beta_1 x + \beta_x^2$	$\beta_0 + \beta_1 x$	medium	low
$\beta_0 + \beta_1 x + \cdots + \beta_x^5$	$\beta_0 + \beta_1 x$	high	low
$\beta_0 + \beta_1 x$	$\beta_0 + \beta_1 x + \beta_x^2$	none	medium
$\beta_0 + \beta_1 x$	$\beta_0 + \beta_1 x + \cdots + \beta_x^5$	none	high

Model complexity

- number of parameters
  - regression coefficients, polynomials, interactions
- hyperparameters (tuning parameters)
  - number of neighbors, budgets, shrinkage parameters



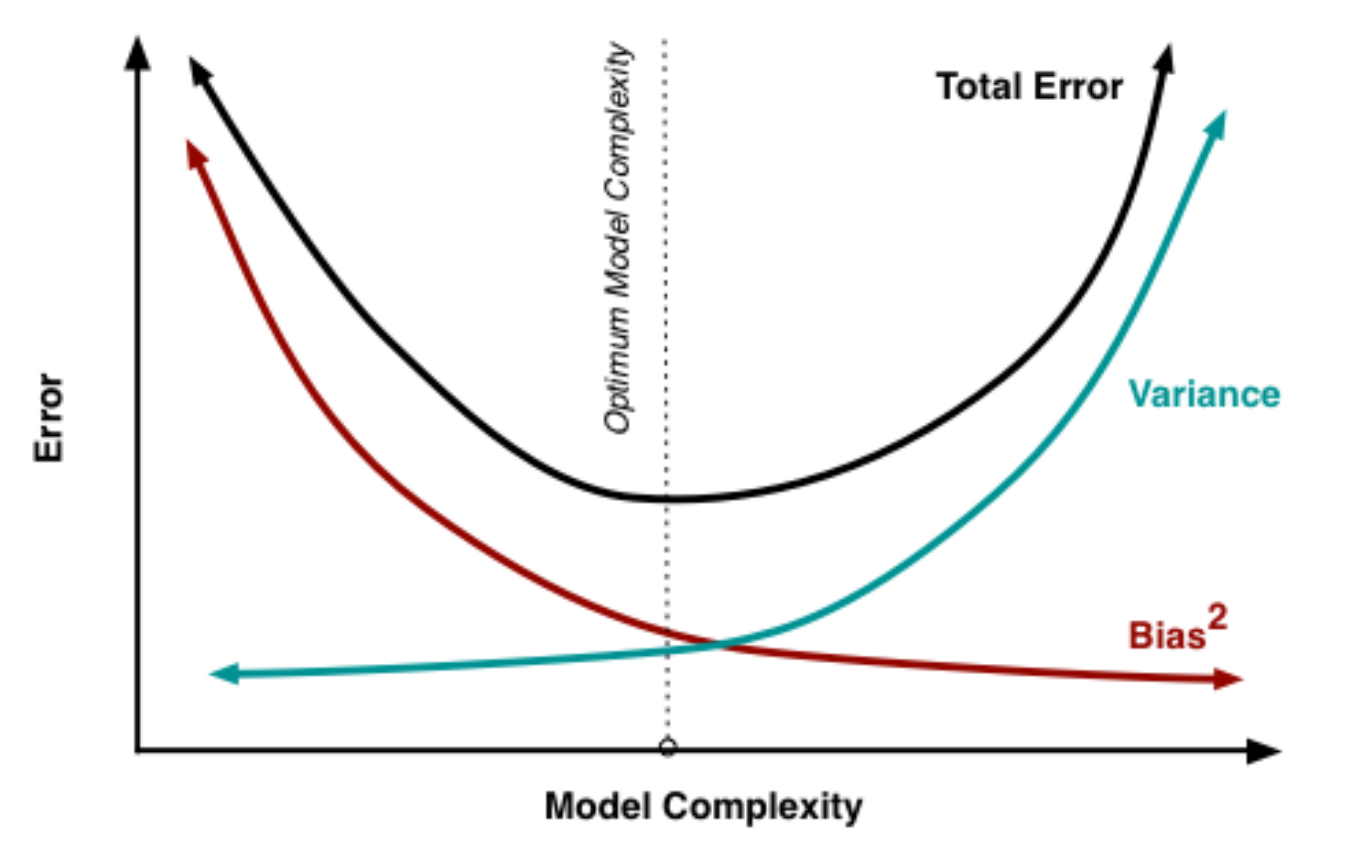
Expectation of MSE

$$E(\text{MSE}) = \text{Bias}^2 + \text{Variance}$$

- $E(MSE)$ is MSE averaged over infinite samples from target population
- we want the model that minimizes $E(MSE)$, and not MSE!!!

The tradeoff

How to find the optimum?



Train/dev/test paradigm

Estimation of E(MSE)

If we would know the true $f(x)$ we could calculate E(MSE)

- but we don't know $f(x)$
- so we need to estimate it

How to estimate E(MSE)?

A. average over observations used to fit $\hat{f}(x)$

B. average over new observations from same data source

C. average over new observations from intented prediction setting

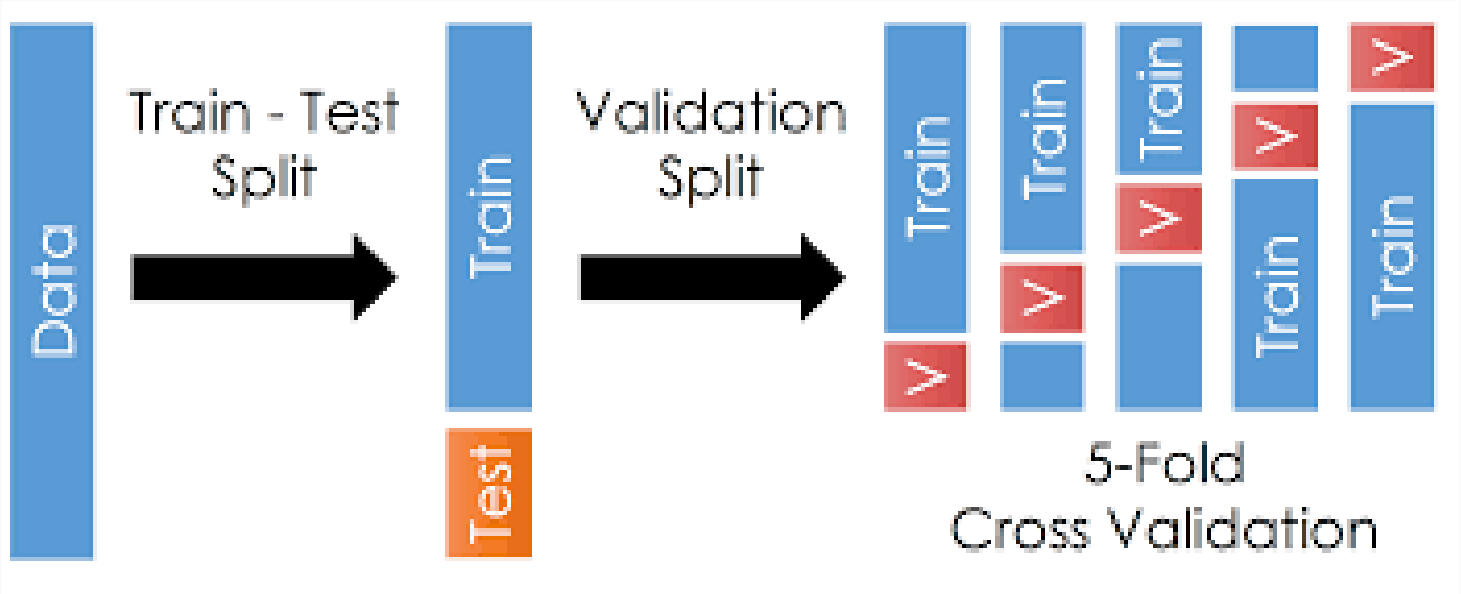
D. or . . .?

Train/dev/test paradigm

Training set: to fit $\hat{f}(x)$ and estimate its parameters

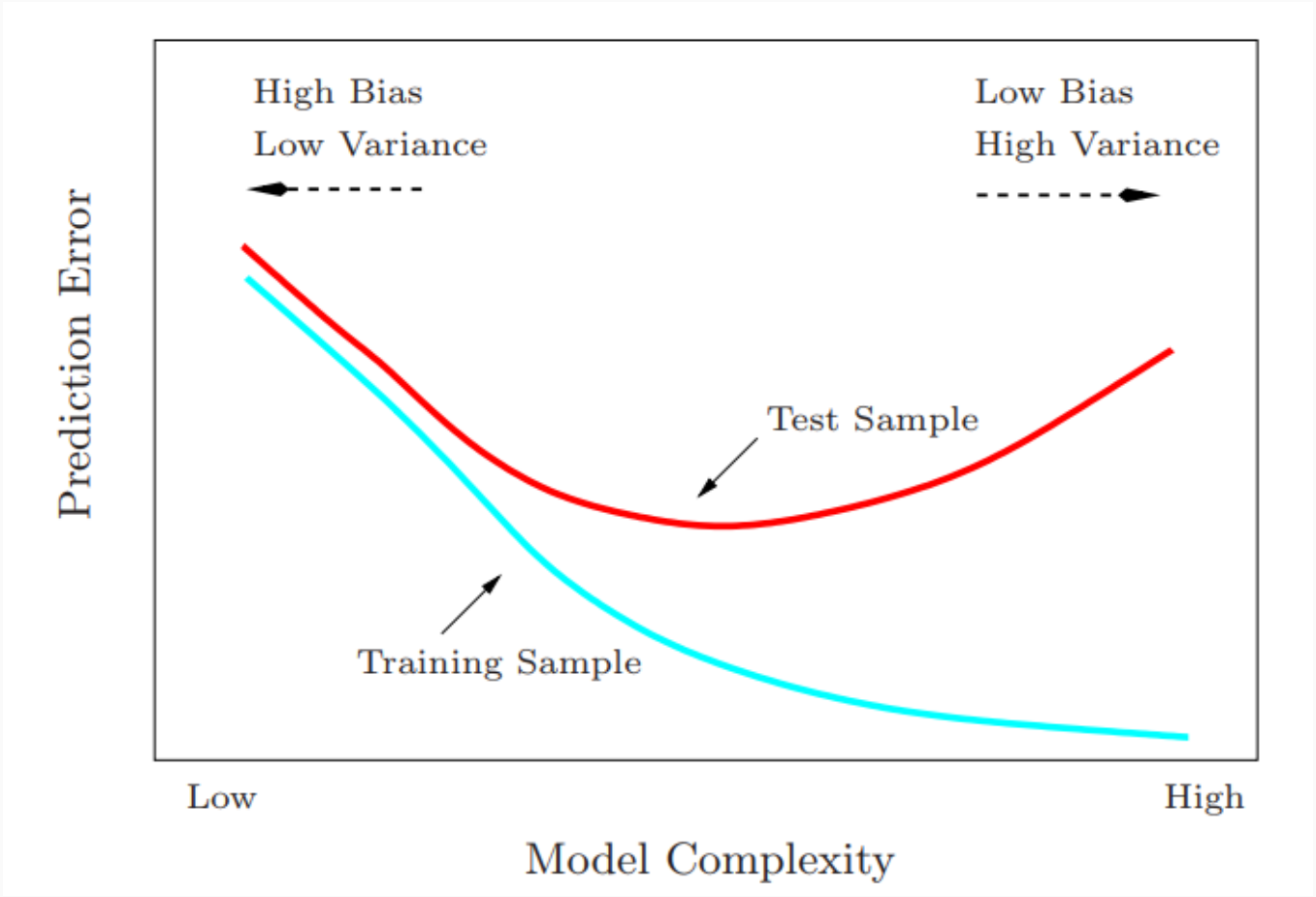
Validation set: to tune hyperparameters of $\hat{f}(x)$

Test data: to estimate $E(\text{MSE})$



Train-test MSE

- training data prefer complexity
- test data punish complexity



k-fold cross validation

Algorithm

1. Train model on training data
2. Compute MSE on validation data
3. Repeat k times and average MSE's as estimate of $E(\text{MSE})$

- If hyperparameter(s) are present:
  - repeat 1 to 3 for different hyperparameters
  - obtain MSE on test set as estimate of $E(\text{MSE})$

Optimal number of folds

- 5 to 10: small bias and low variance
- $k = n$ is Leave One Out CV (LOOCV): no bias but high variance

Recap

- simple models underfit the data (high bias, low variance)
- complex models overfit the data (low bias, high variance)
- search for optimum in bias-variance tradeoff
- train/dev to tune hyperparameters
- train/test to compare MSEs of competing models

The `caret` package

R package for training models

Short for **C**lassification **A**nd **RE**gression **T**raining

https://cran.r-project.org/web/packages/caret/vignettes/caret.html

Functions for streamline model training for over 40 models

http://topepo.github.io/caret/available-models.html

- data splitting
- pre-processing
- feature selection
- model tuning using resampling
- variable importance estimation

Cross-validation procedure

```
# split data in train/test set
inTrain <- createDataPartition(y    = data$y,
                               p    = .8,
                               list = FALSE)
Train   <- data[ inTrain, ]
Test    <- data[-inTrain, ]

# train KNN model on training set
knn_train <- train(formula,
                   data      = Train,
                   method    = "knn"
                   tuneGrid  = expand.grid(<range tuning parameters>),
                   trControl = trainControl(method = "cv",
                                            number = 5)
                   )

# get predictions for test set
knn_test <- predict(knn_train, newdata = Test)
```

Lab 1B

1. Fit linear, quadratic and KNN model
2. Compare test MSE to select bess model
3. Use `caret` data partition and cross-validation