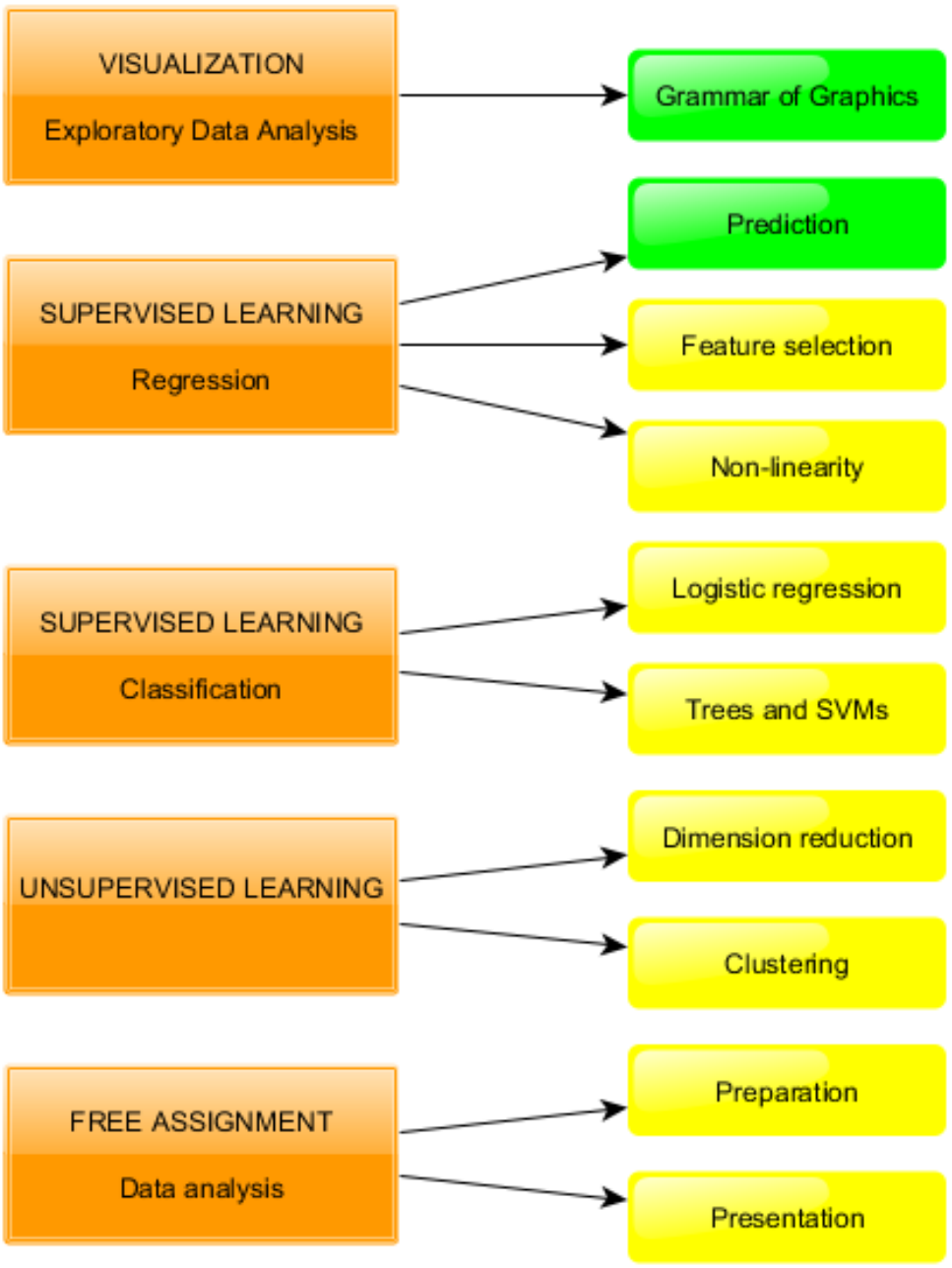
Supervised Learning: Regression

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

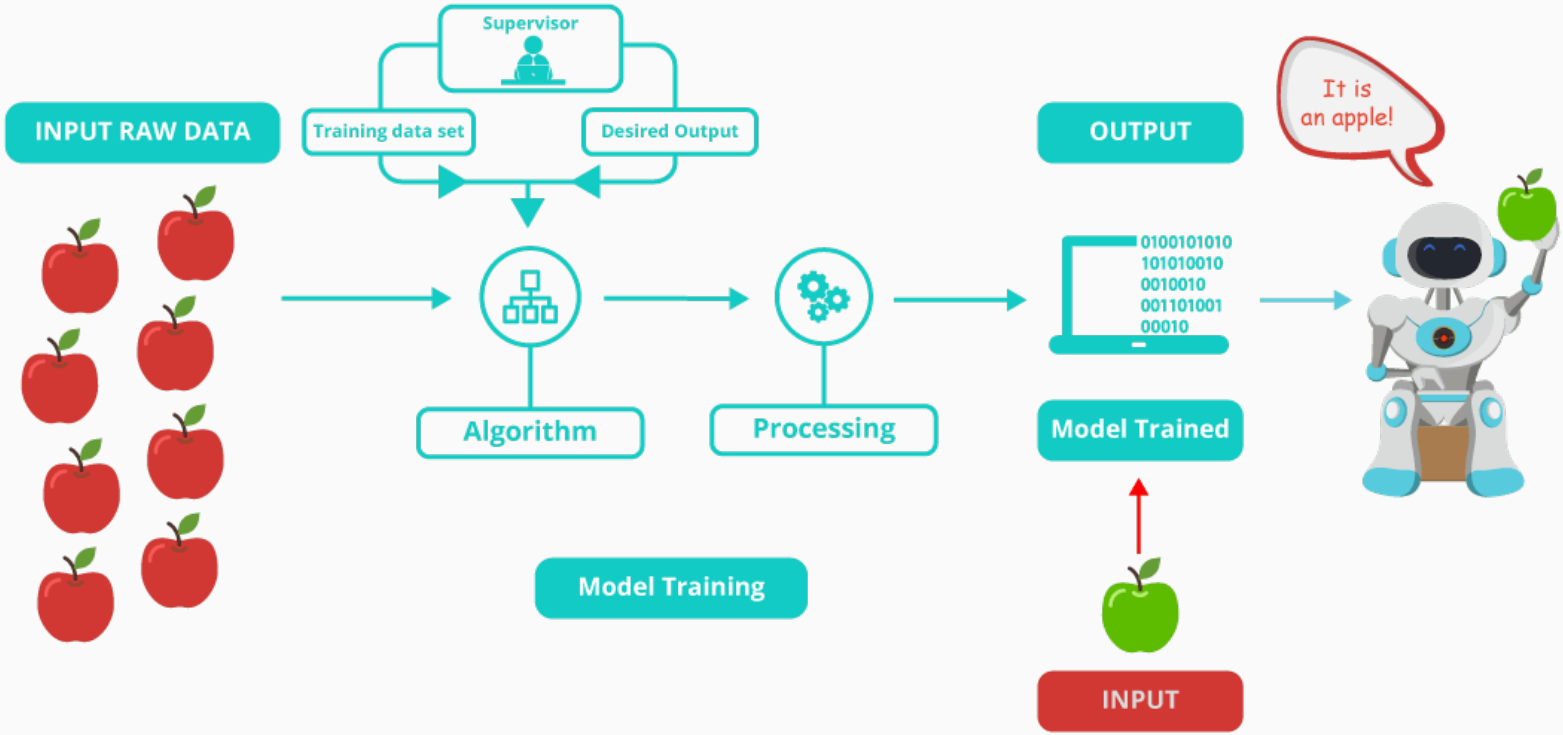
Content

1. Supervised learning
2. KNN vs regression
3. Bias-variance tradeoff
4. Train/dev/test paradigm
5. The `caret` package

Supervised learning

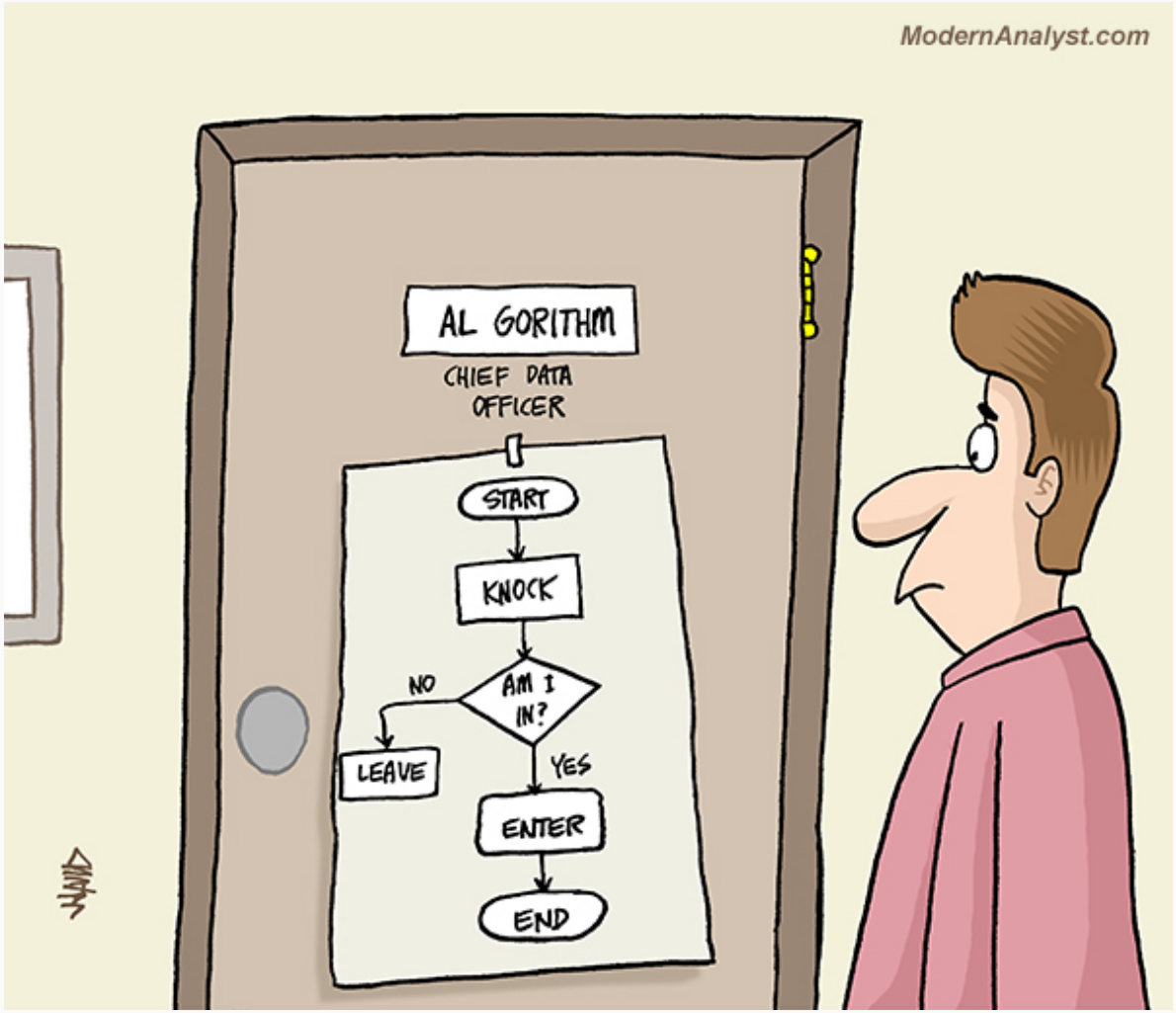
Presence outcome variable

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



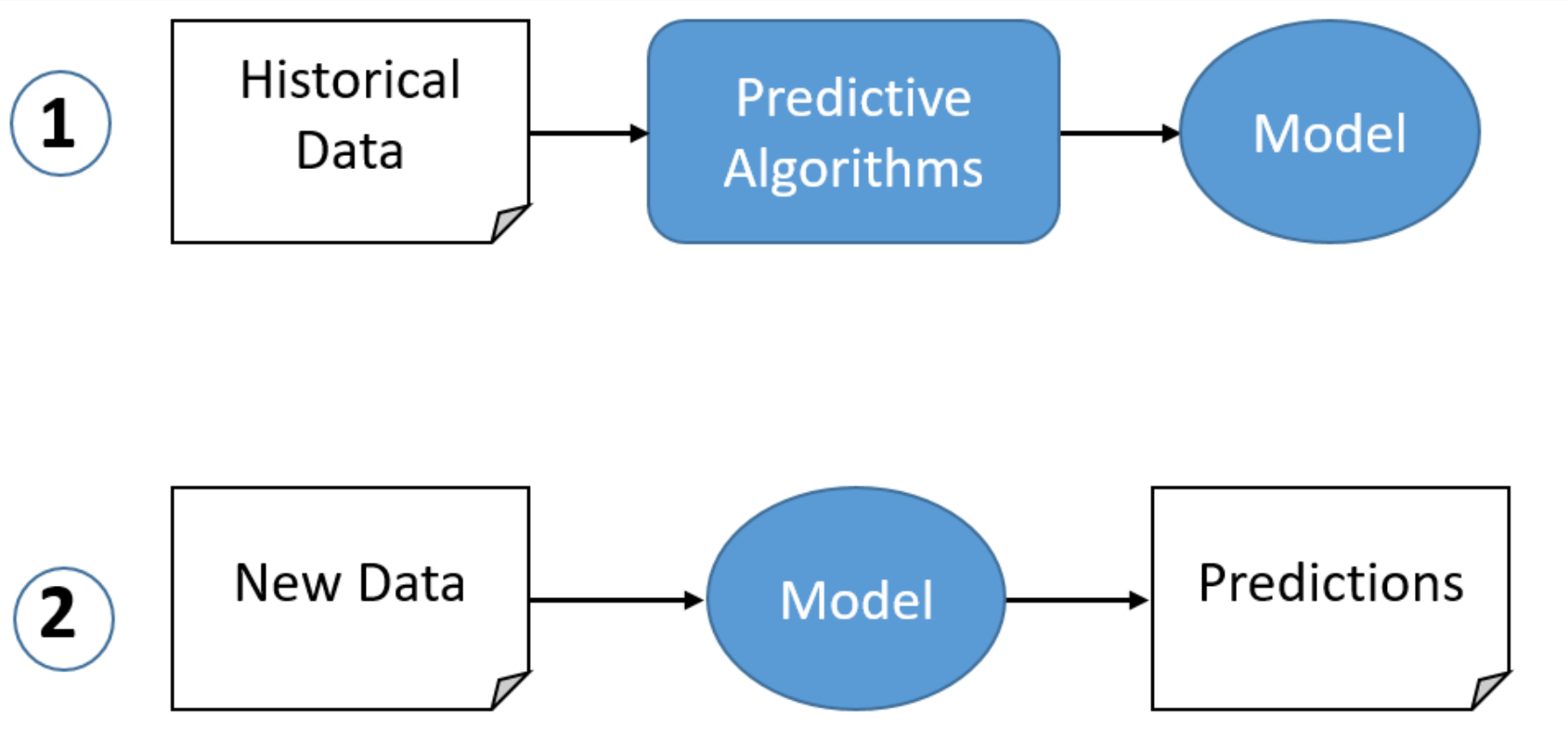
Algorithm

- a set of instructions to solve a problem



Model training

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



Learning by training?

Not really, there is a systematic approach!

Statistal models

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

$\sum_i \epsilon_i^2$ is error variance or Mean Squared Error (MSE)

Choice of $f(x)$

Many competing models

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

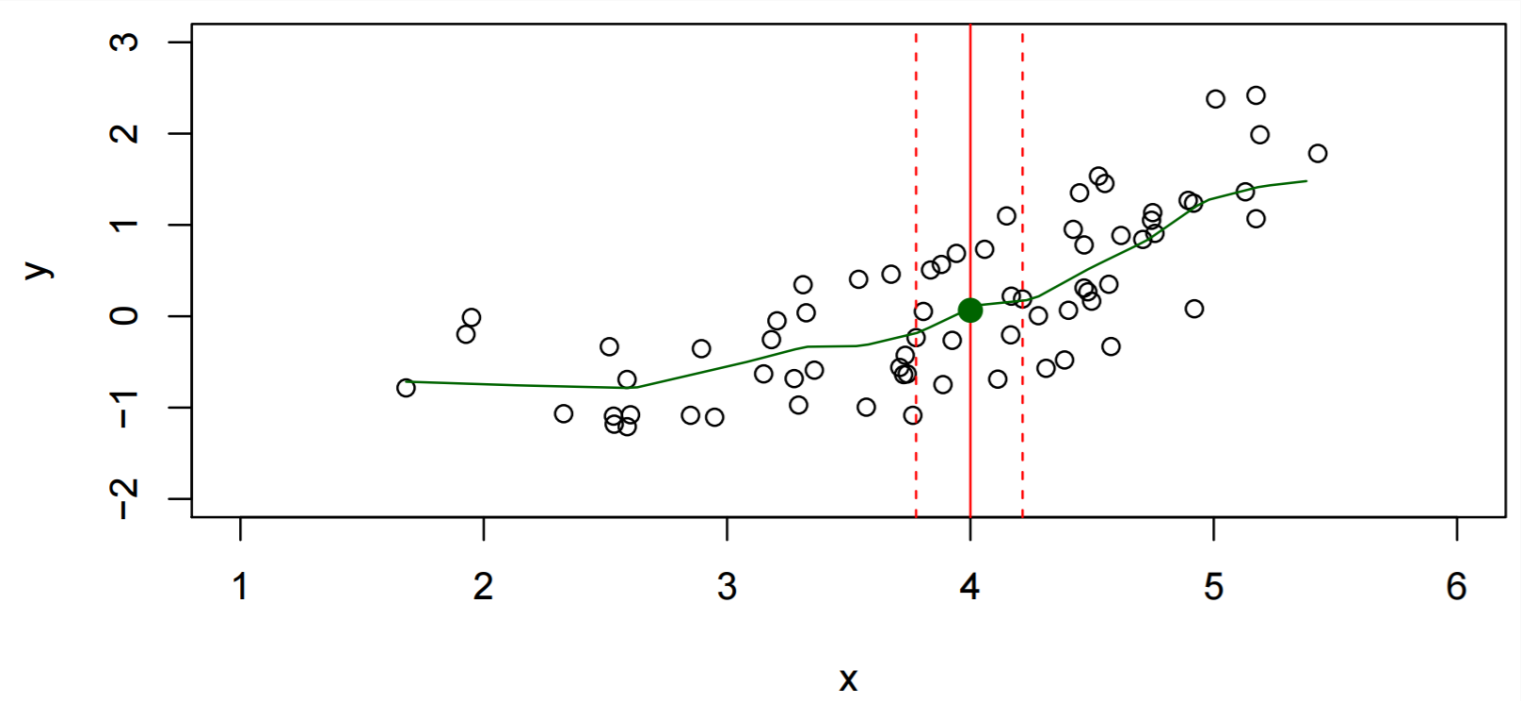
These models vary in complexity and interpretability

KNN versus regression

k-nearest neighbors (KNN)

Nonparametric model (distribution y unspecified)

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

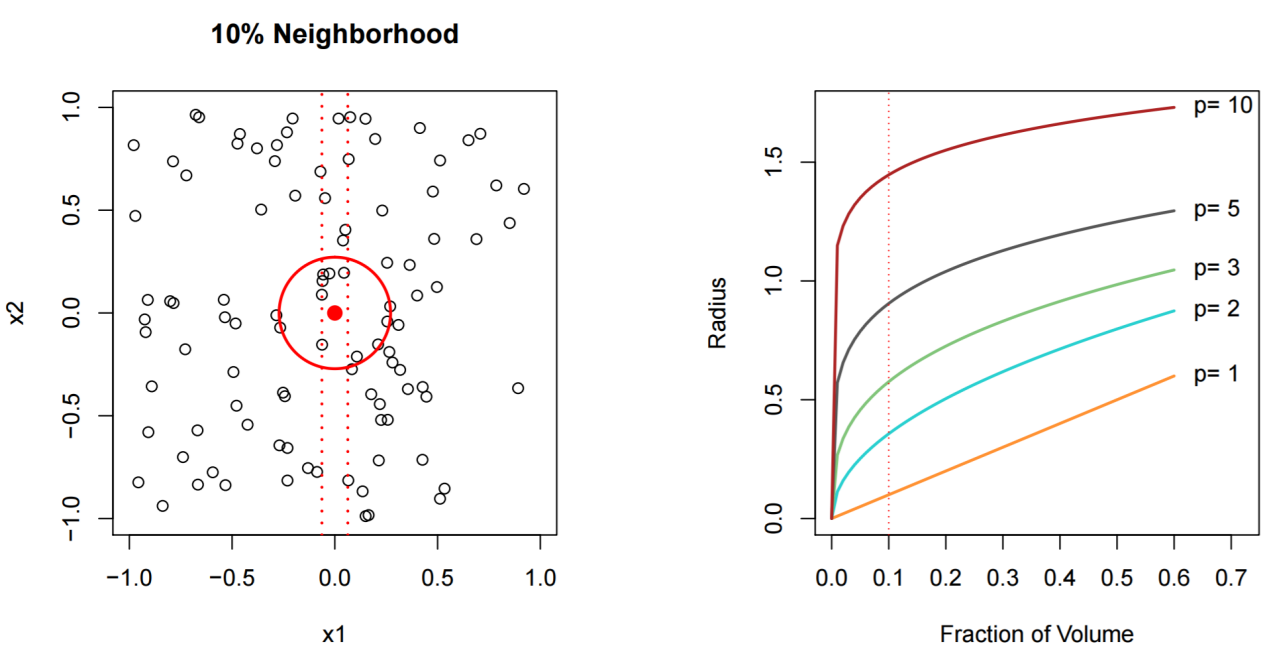


Prediction for $x = 4$ is mean y of k nearest data points

Curse of dimensionality

Limited to low-dimensional data

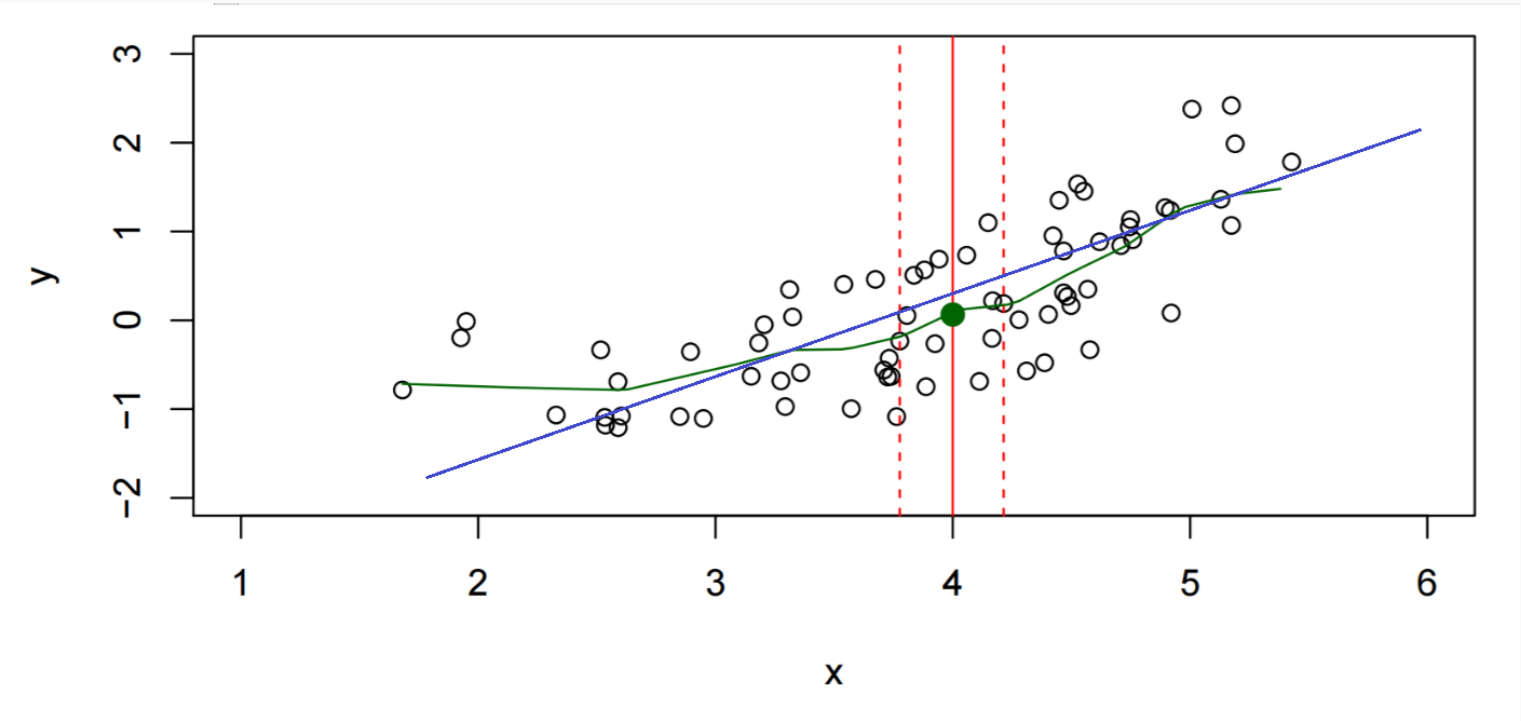
- KNN breaks down with increasing number of predictors
- distance between points increases and neighborhood shrinks



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 regression 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?

Minimizing MSE

Fit m competing functions $\hat{f}_j(x)$ to the data

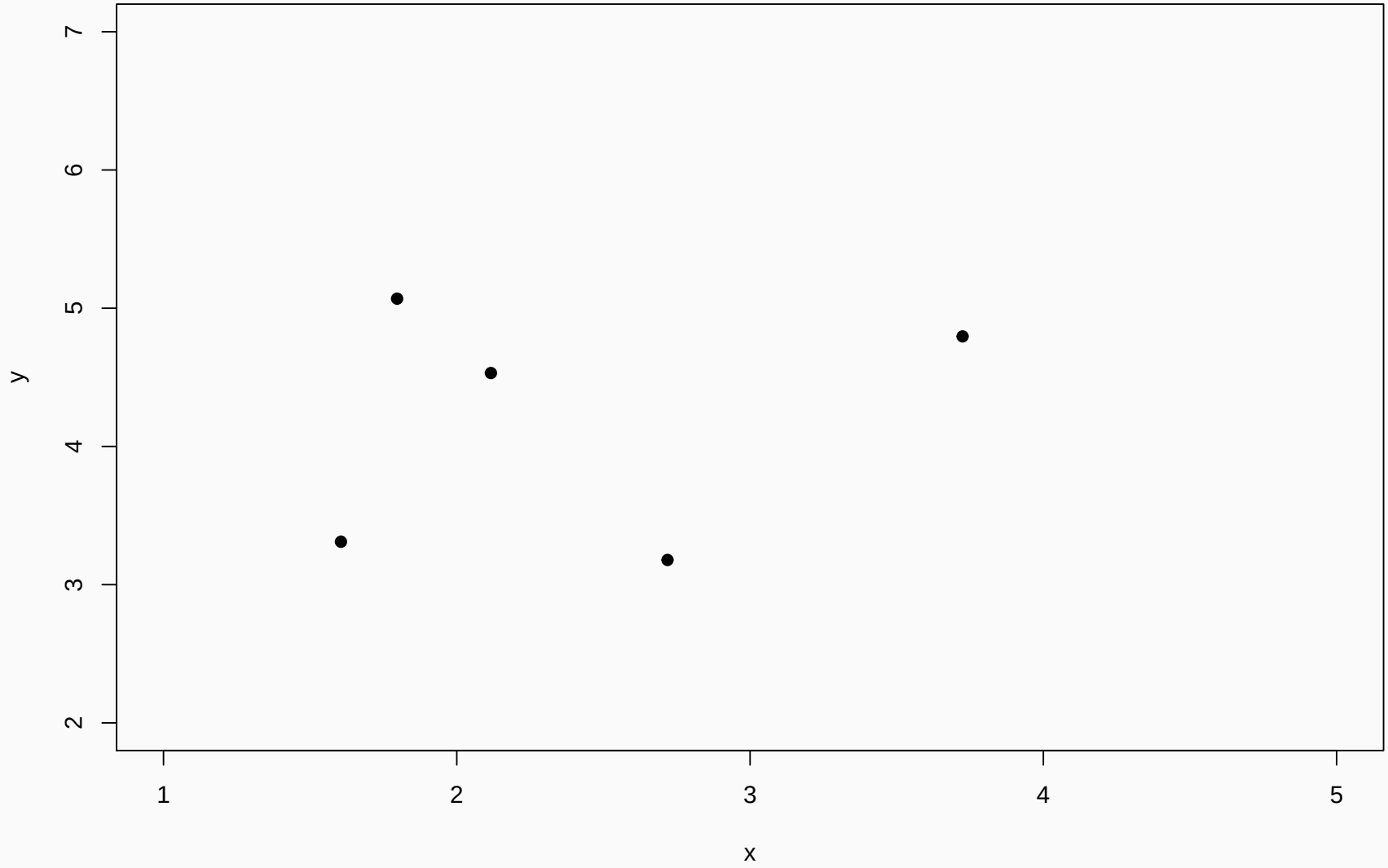
- for each model compute the MSE

The model with the smallest MSE ios closest to the true $f(x)$

TRUE?

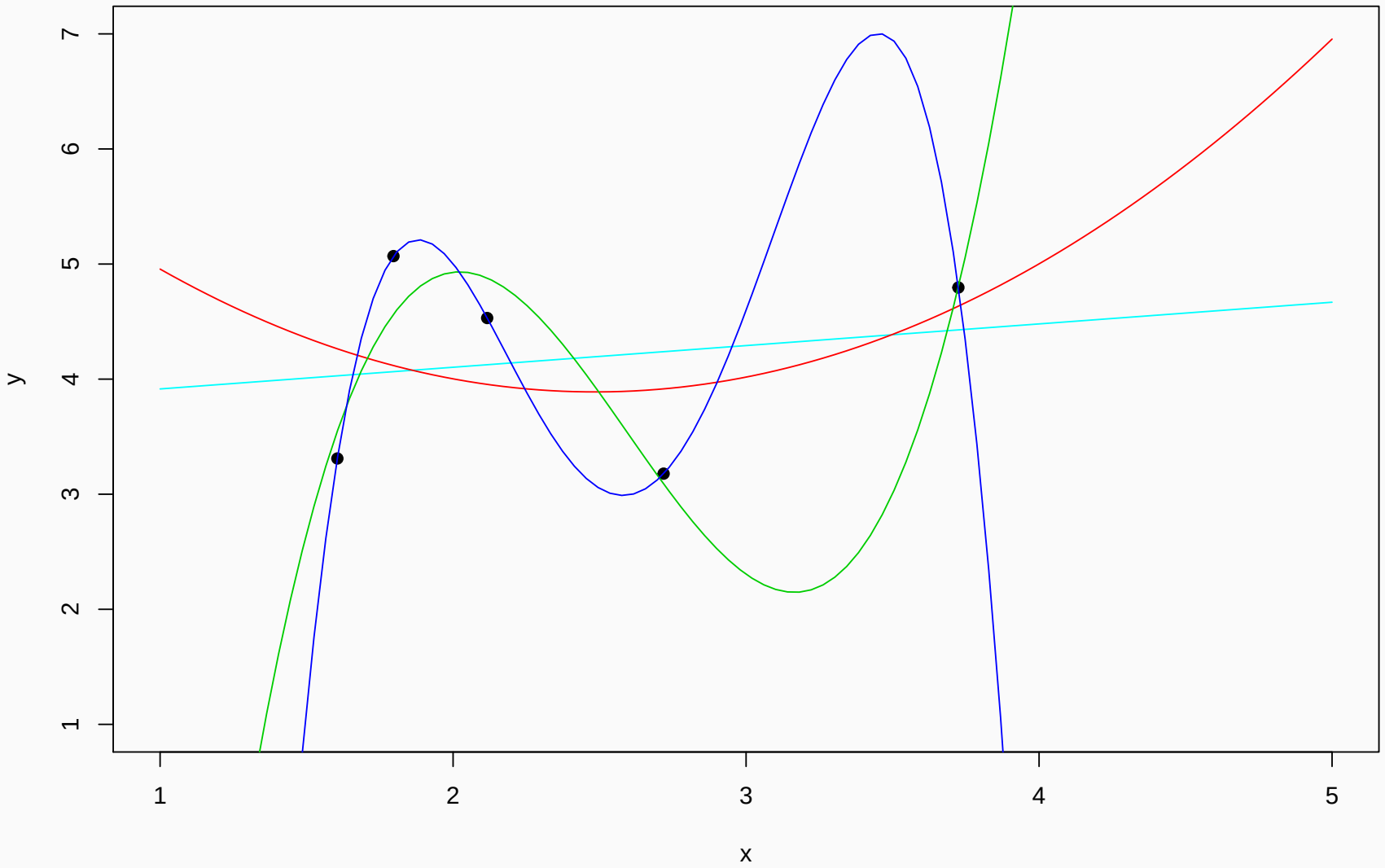
Data points generated from $f(x)$

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

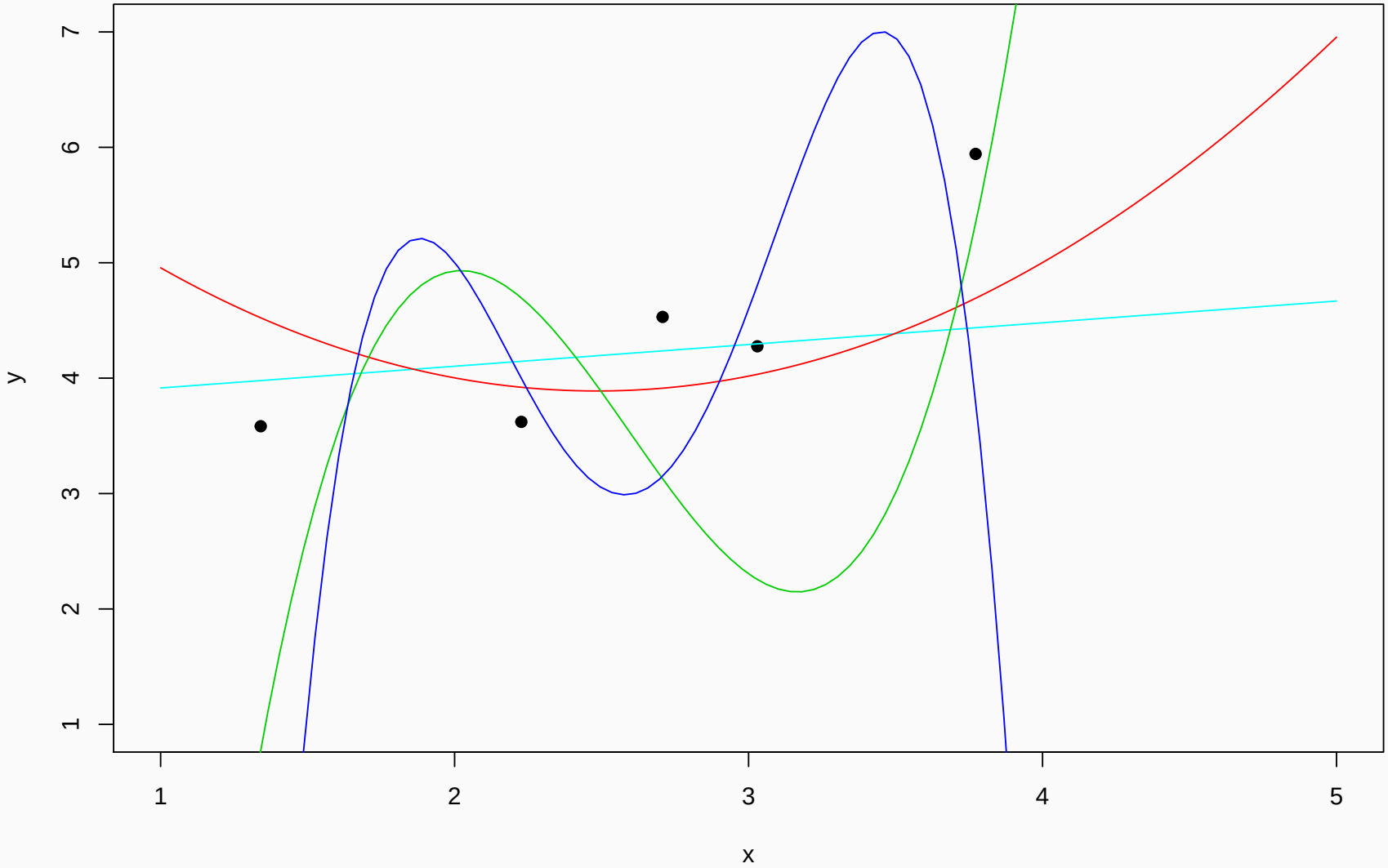
Comparison of the MSE of the fitted models

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!!!

Bias-variance tradeoff

Bias vs variance

Unbiased model

- predictions correct when averaged over (infinite) data sets

High variance model

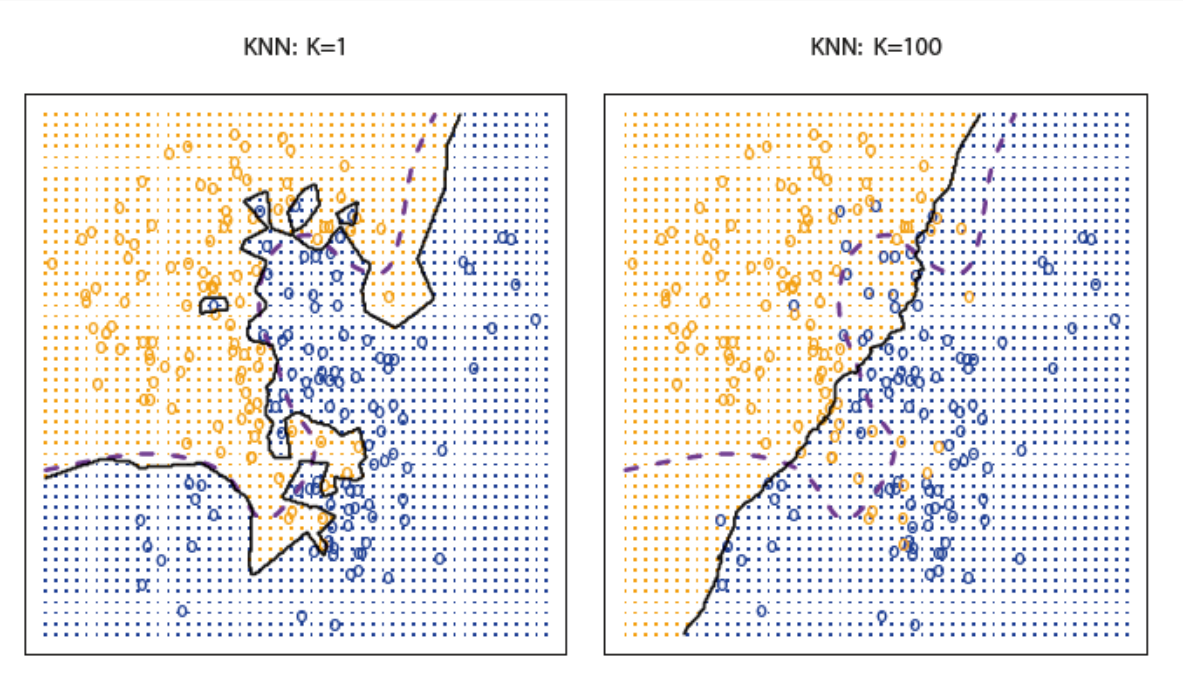
- predictions vary wildly over diffferent data sets

So what if $f(x) = \beta_0 + \beta_x x$, and $\hat{f}(x) = \beta_0 + \beta_1 x + \beta_2 x$?

Example KNN

Dotted line is true border blue/orange data points

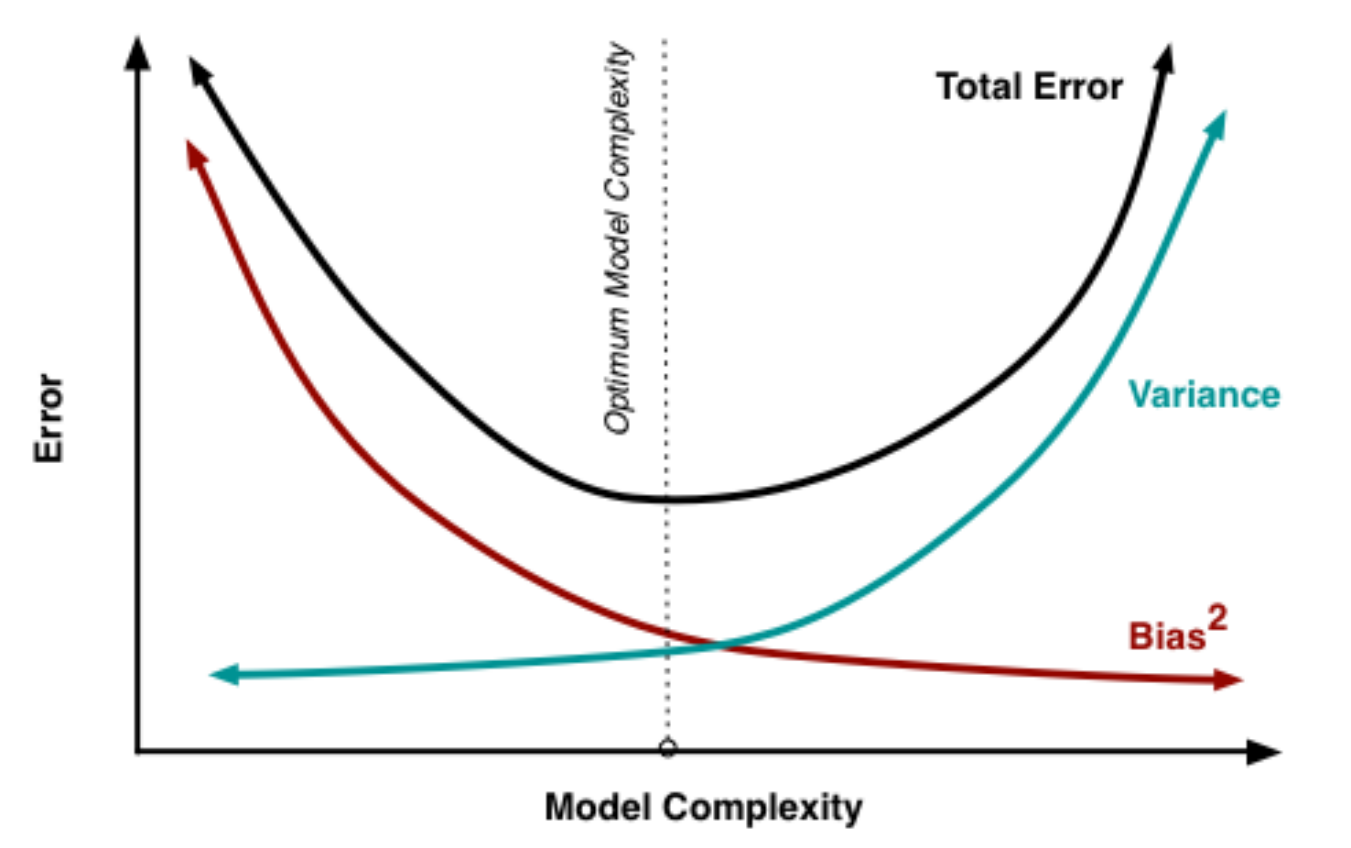
- low bias, high variance for $k = 1$
- high bias, low variance for $k = 100$



Bias-variance tradeoff

Optimum corresponds to expected MSE, i.e. the MSE of $f(x)$ instead of $\hat{f}(x)$

- how to find this optimum?



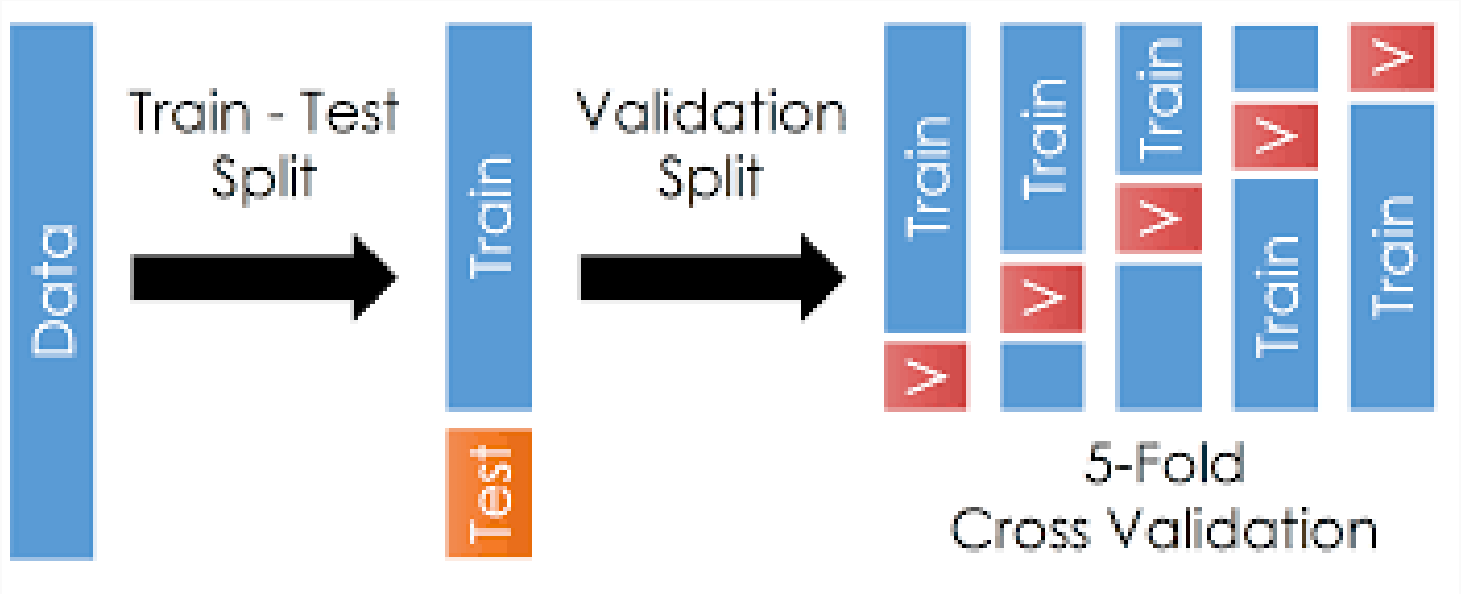
Train/dev/test paradigm

Data partitioning

Training set: estimate parameters $\hat{f}(x)$

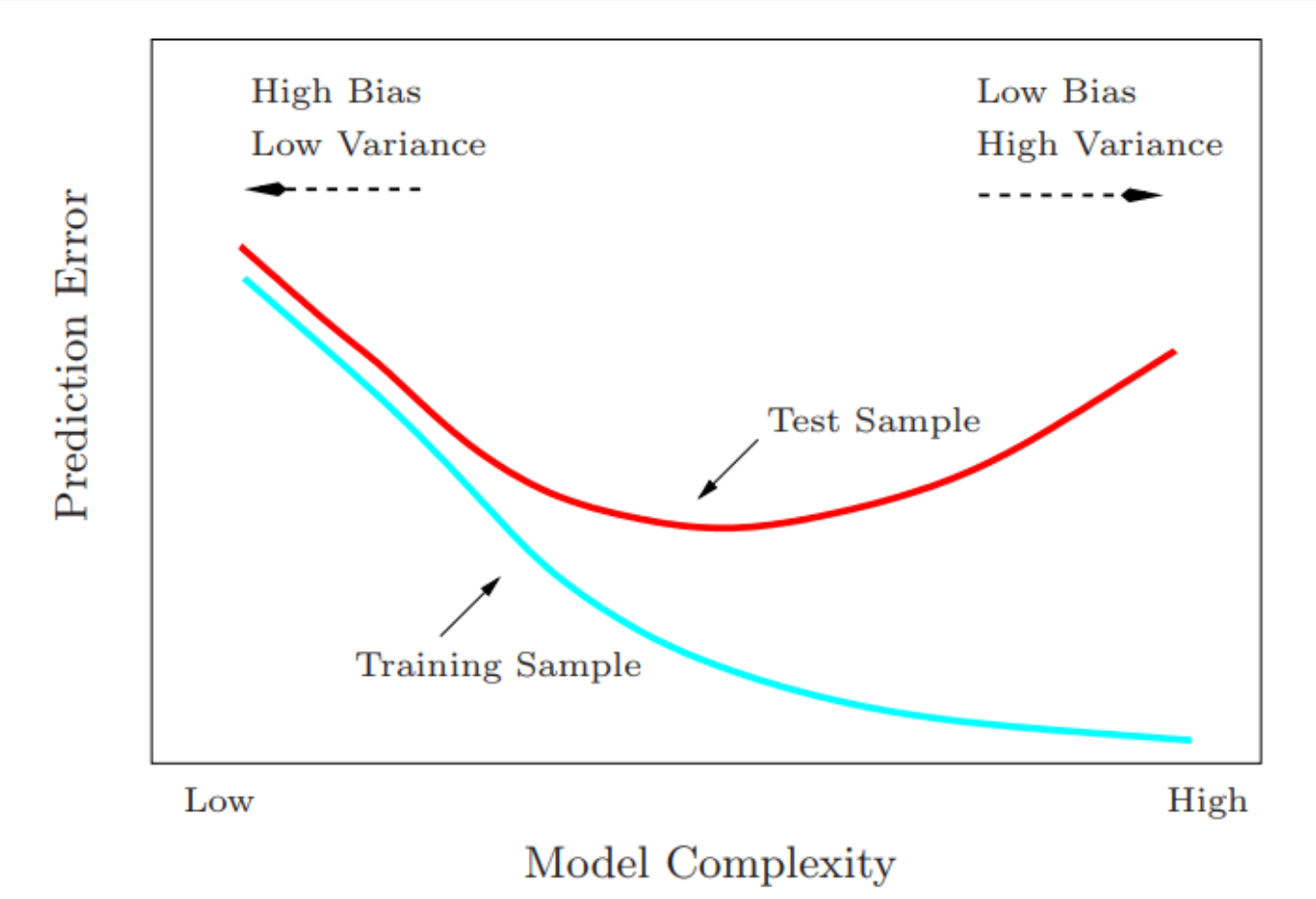
Validation set: tune hyperparameters $\hat{f}(x)$ (if any)

Test data: estimate expected MSE



Train-test MSE

- training data prefer model complexity
- test data punish too much model complexity



Estimate expected MSE

Train/test

1. Train model on training data
2. Compute MSE test data as estimate expected MSE

Train/dev/test

1. Cross-validate hyperparameters on training/validation data
2. Train model with tuned hyperparameter on training data
3. Compute MSE test data as estimate expected MSE

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

```r
# 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