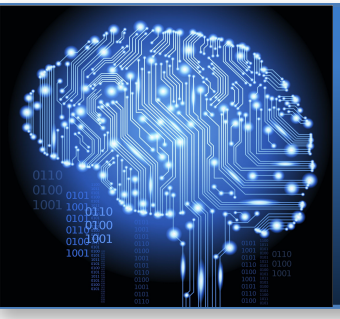


FIT1043 Lecture 6
Introduction to Data Science

Mahsa Salehi*

Faculty of Information Technology, Monash University

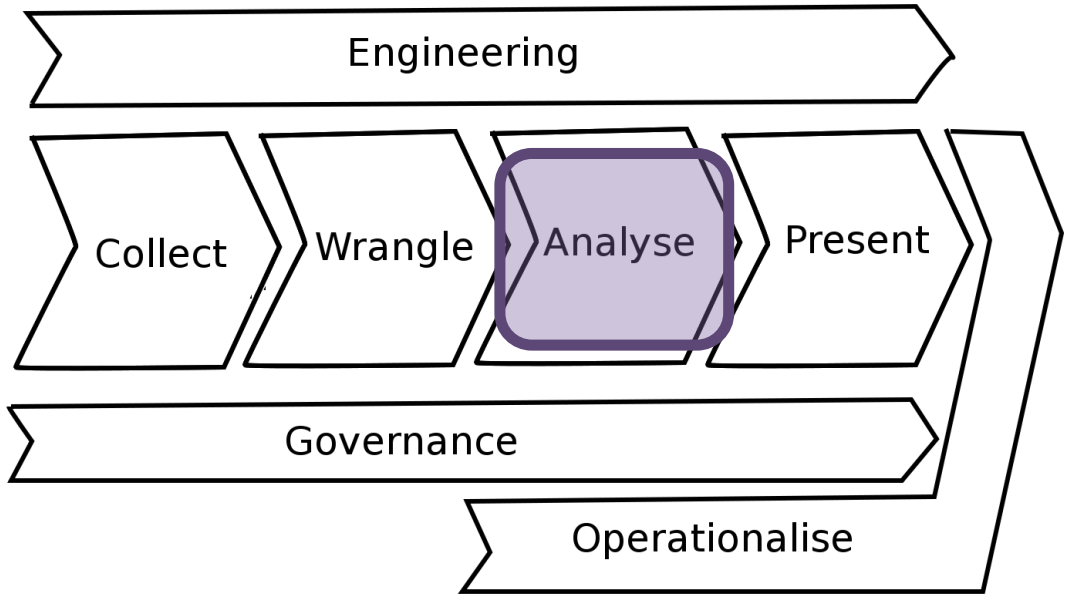
Semester 2, 2022

*with material from © Wray Buntine

Unit Schedule

Week	Activities	Assignments
1	Overview of data science	Weekly Lecture/tutorial active participation assessment
2	Introduction to Python for data science	
3	Data visualisation and descriptive statistics	
4	Data sources and data wrangling	
5	Data analysis theory	Assignment 1
6	Regression analysis	
7	Classification and clustering	
8	Introduction to R for data science	
9	Characterising data and "big" data	Assignment 2
10	Big data processing	
11	Issues in data management	
12	Industry guest lecture	Assignment 3

Our Standard Value Chain



Weeks 5-7

Outline

- Linear regression terminology
- How to calculate model parameters
- Underfitting vs Overfitting
- Bias and Variance
- No free lunch theorem
- Ensemble models

Learning Outcomes (Week 6)

By the end of this week you should be able to:

- Fit linear regression and polynomial regression models to a given dataset
- Explain overfitting and underfitting of different models
- Comprehend bias and variance trade-off
- Comprehend the importance of “No Free Lunch Theorem”
- Explain what ensemble models are

Data Analysis Algorithms
Regression

From The Elements of Statistical Learning

by T. Hastie, R. Tibshirani and J. Friedman

Regression

What is Regression?

- Look for relationships amongst variables

Real World Example:

- Identify the relation between salary and experience, education, and role

Terminology

Variables can be:

- *Independent Variables/Inputs/Predictors*
  E.g., experience, education, role

- *Dependent Variables/Outputs/Responses*
  E.g., salary of employee

Observation is a data point, row, or sample in a dataset

- **E.g.,** an employee's salary, experience, education, role.

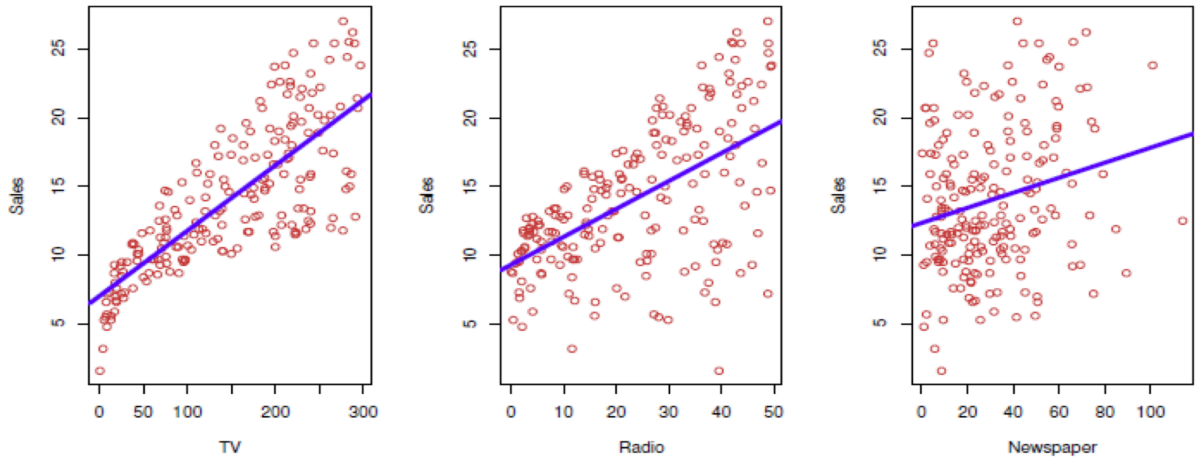
When Use Regression

- To determine how multiple variables are related

  E.g., determine *if* and *to what extent* the experience or education impact salaries

- To forecast a value

  E.g., predict electricity consumption given the outdoor temperature, time of day, and number of residents in that household



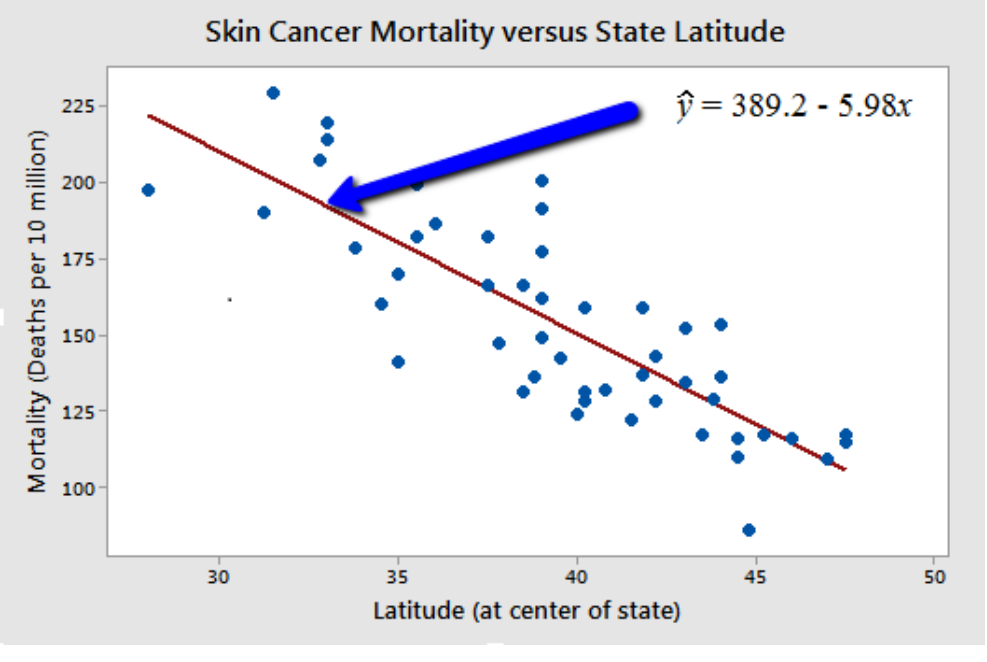
Example: Sales ~ TV, Radio, newspaper

Simple Linear Regression (two-dimensional space):

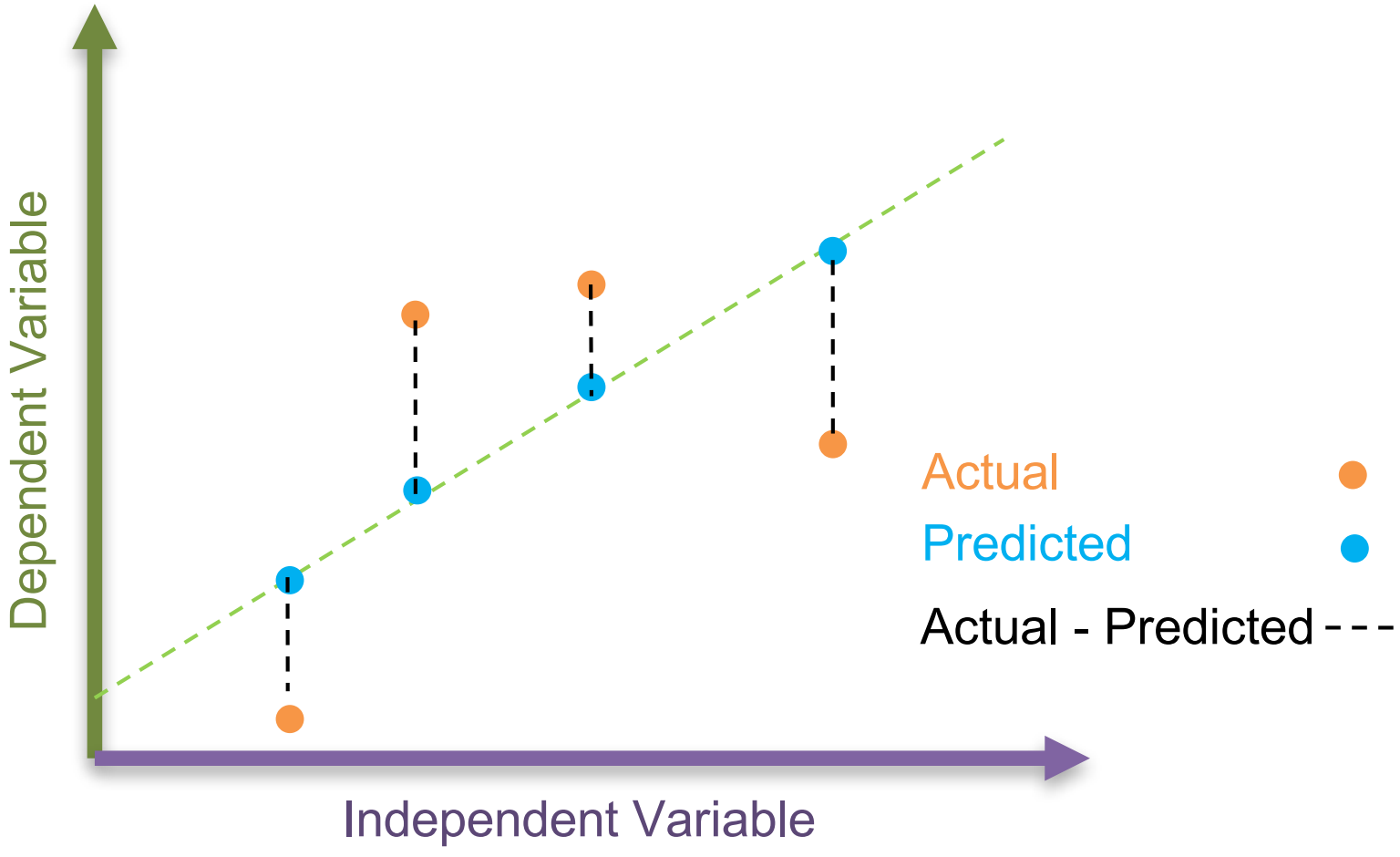
Regression fits a very simple equation to the data:

$$\hat{y}(x;\vec{a}) = a_0 + a_1 x$$

Here $\hat{y}(x;\vec{a})$ is prediction for y at the point x using the model parameters $\vec{a}$ = (a_0, a_1), i.e. the intercept and slope terms.



Best Fitting Line



Aim is that the predicted response, be as close as possible to the actual response.

Calculating Parameters- Intuition

$$\hat{y}(x;\vec{a}) = a_0 + a_1 x$$

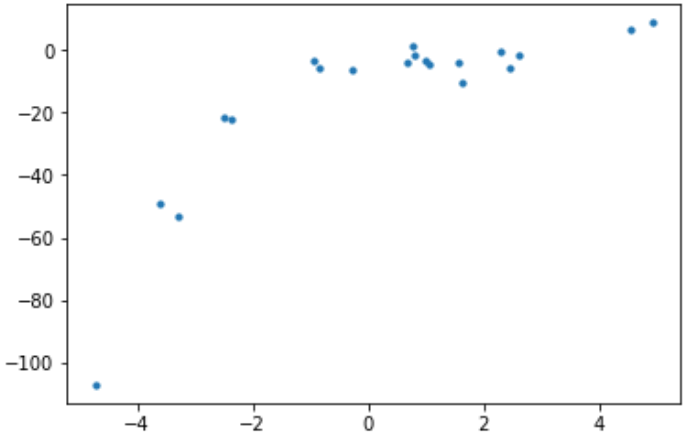
- Given some data pairs (x_1, y_1), ...,(x_N , y_N), we fit a model by finding the vector $\vec{a}$ that minimises the loss function:

$$\text{mean square error} = MSE_{train} = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{y}(x_i;\vec{a}) - y_i\right)^2$$

- Compare the derivatives to zero

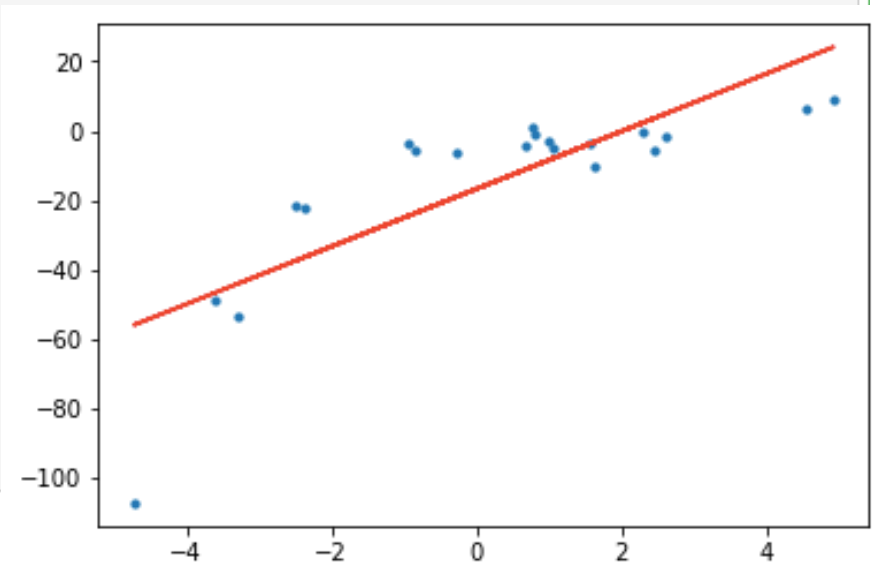
Real-world Example (Python)

```python
#import required packages
import numpy as np
import matplotlib.pyplot as plt

#provide data
np.random.seed(0)
x = 2 - 3 * np.random.normal(0, 1, 20)
y = x - 2 * (x ** 2) + 0.5 * (x ** 3) + np.random.normal(-3, 3, 20)
plt.scatter(x,y, s=10)
plt.show()
```



Real-world Example (Python)

```python
#import required packages
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

#provide data
np.random.seed(0)
x = 2 - 3 * np.random.normal(0, 1, 20)
y = x - 2 * (x ** 2) + 0.5 * (x ** 3) + np.random.normal(-3, 3, 20)

# transforming the data to include another axis
x = x[:, np.newaxis]
y = y[:, np.newaxis]

#create a liniear regression model
model = LinearRegression()
model.fit(x, y)
y_pred = model.predict(x)

#diplay the best fit line
plt.scatter(x, y, s=10)
plt.plot(x, y_pred, color='r')
plt.show()
```



- Linear regression is unable to capture the patterns in the data. This is an example of under-fitting.
- To overcome under-fitting, we need to increase the complexity of the model

Polynomial Regression

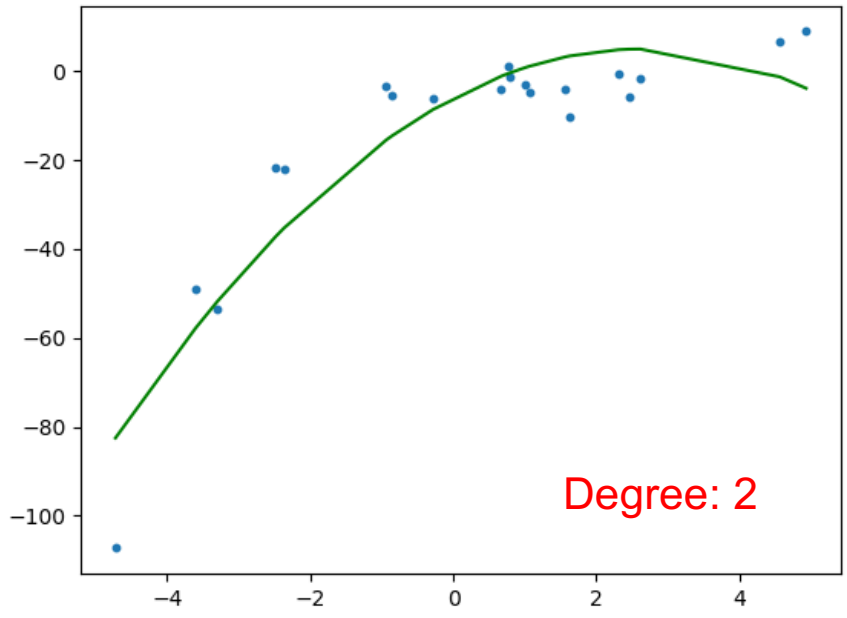
- Assume the polynomial relationship between the inputs and output.

- E.g., 10th order (aka degree) polynomial

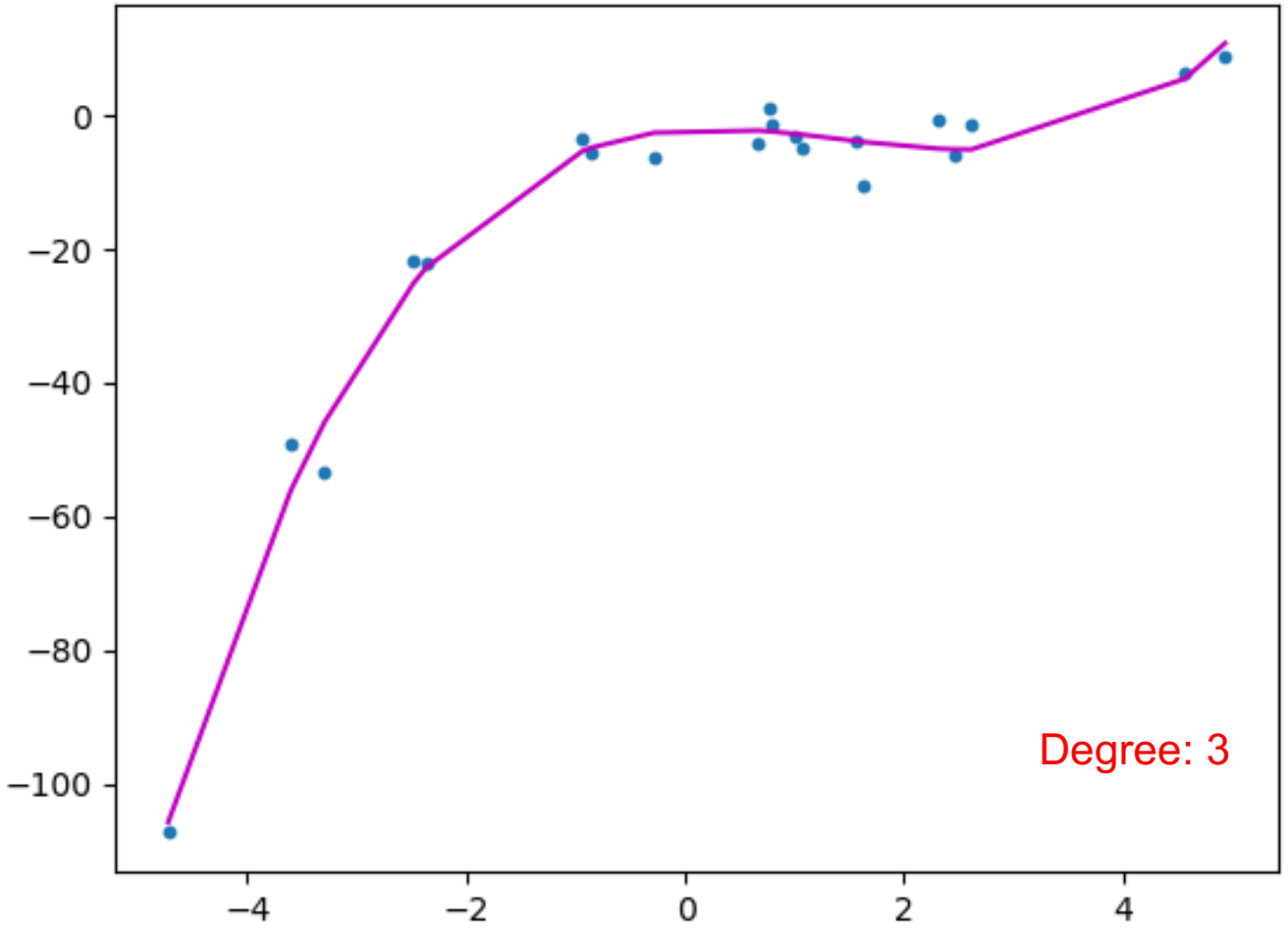
$$\hat{y}(x;\vec{a}) = a_0 + a_1 x + a_2 x^2 + \ldots a_9 x^9 + a_{10} x^{10} = \sum_{i=0}^{10} a_i x^i$$

Real-world Example (cont.)

```python
#import required packages
import operator
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import PolynomialFeatures

#provide data
np.random.seed(0)
x = 2 - 3 * np.random.normal(0, 1, 20)
y = x - 2 * (x ** 2) + 0.5 * (x ** 3) + np.random.normal(-3, 3, 20)

# transforming the data to include another axis
x = x[:, np.newaxis]
y = y[:, np.newaxis]

#create polynomial regression
polynomial_features= PolynomialFeatures(degree= 2)
x_poly = polynomial_features.fit_transform(x)

model = LinearRegression()
model.fit(x_poly, y)
y_poly_pred = model.predict(x_poly)

rmse = np.sqrt(mean_squared_error(y,y_poly_pred))
r2 = r2_score(y,y_poly_pred)
print(rmse)
print(r2)

plt.scatter(x, y, s=10)
# sort the values of x before line plot
sort_axis = operator.itemgetter(0)
sorted_zip = sorted(zip(x,y_poly_pred), key=sort_axis)
x, y_poly_pred = zip(*sorted_zip)
plt.plot(x, y_poly_pred, color='m')
plt.show()
```



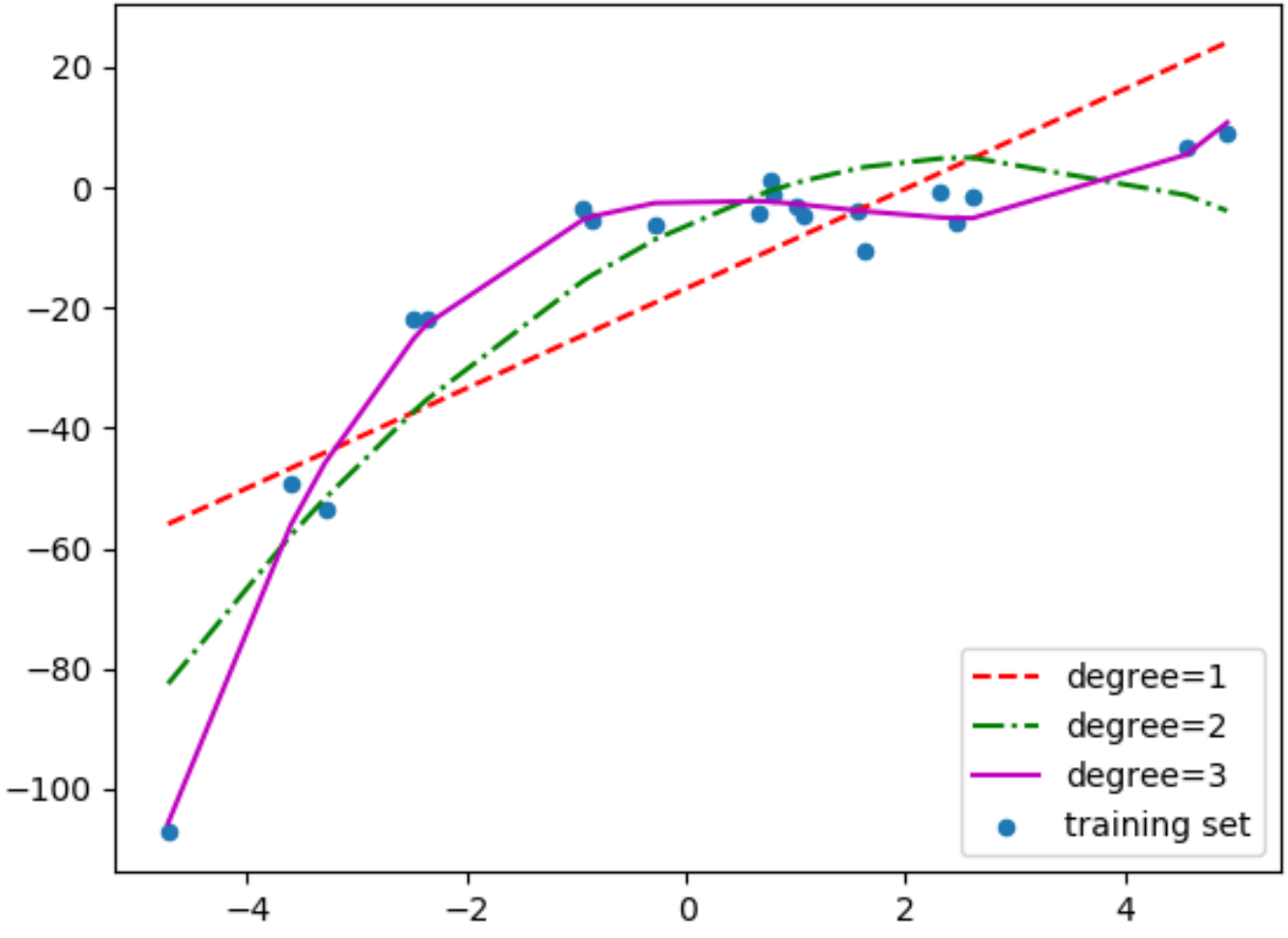
Polynomial Regression



FLUX Question

FLUX code:
5NKWED

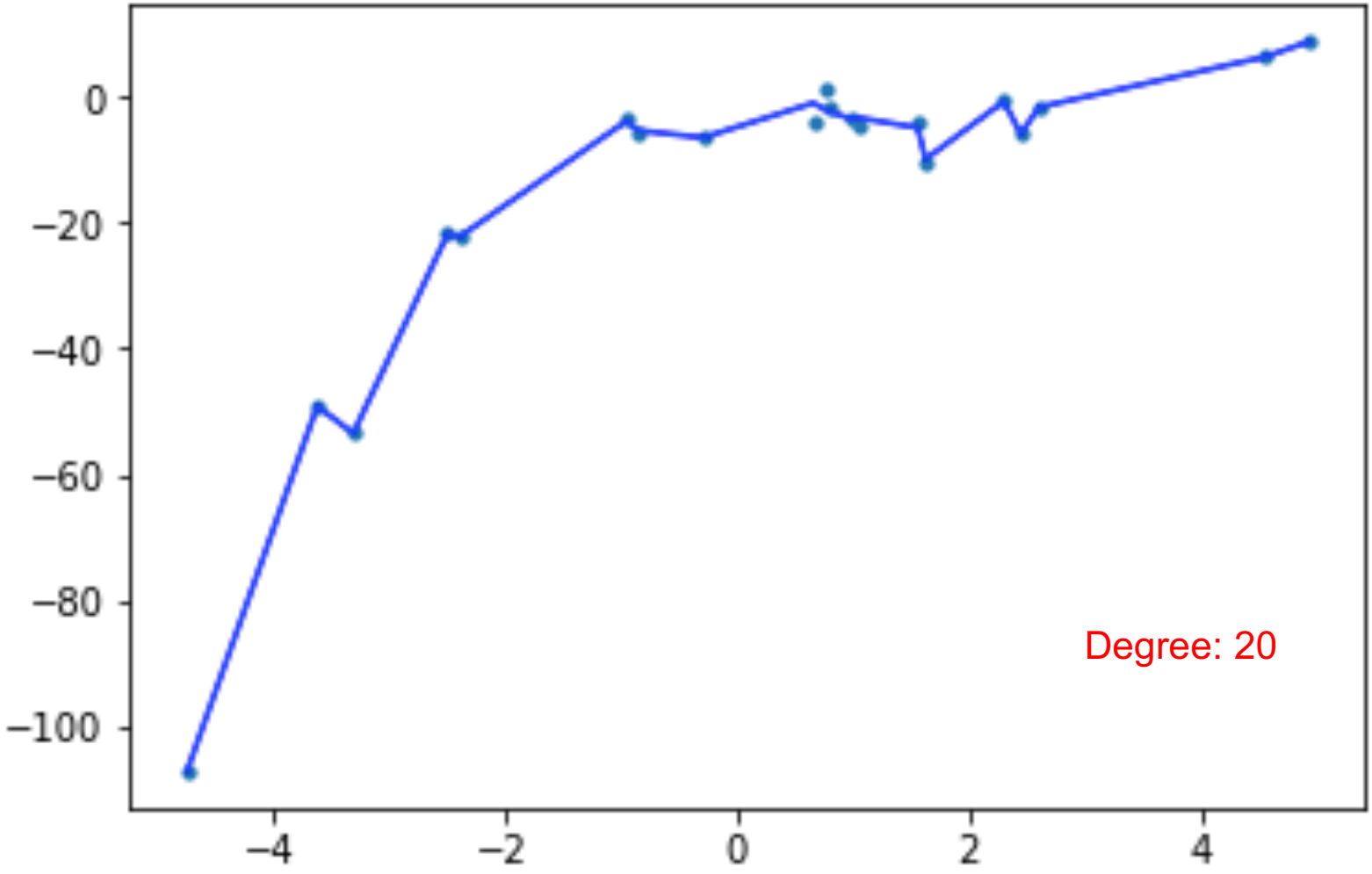
What is the best degree? 1, 2 or 3?



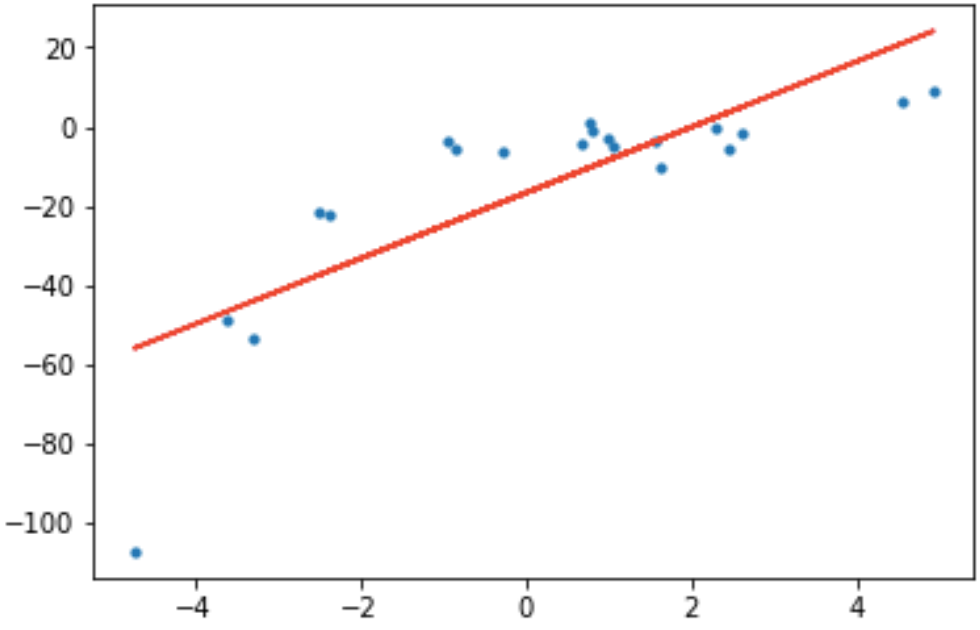
FLUX Question

FLUX code:
5NKWED

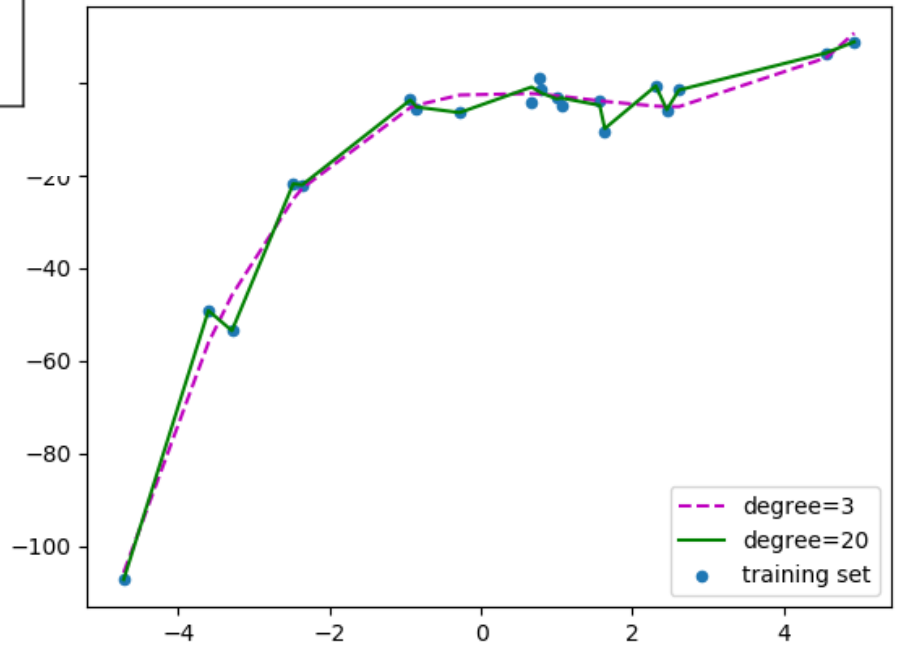
Is this fit better than previous fits?



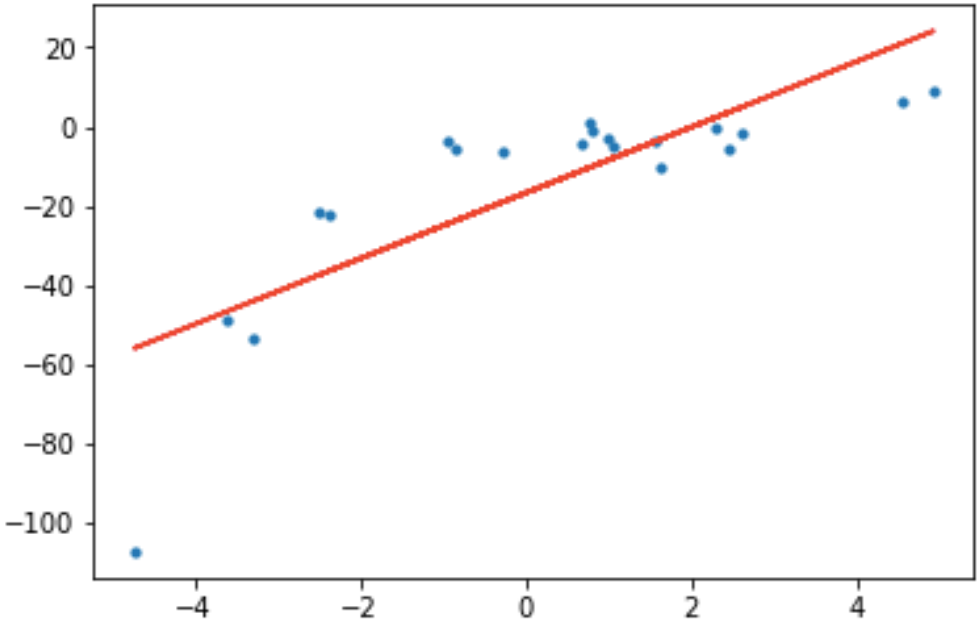
Underfitting and Overfitting



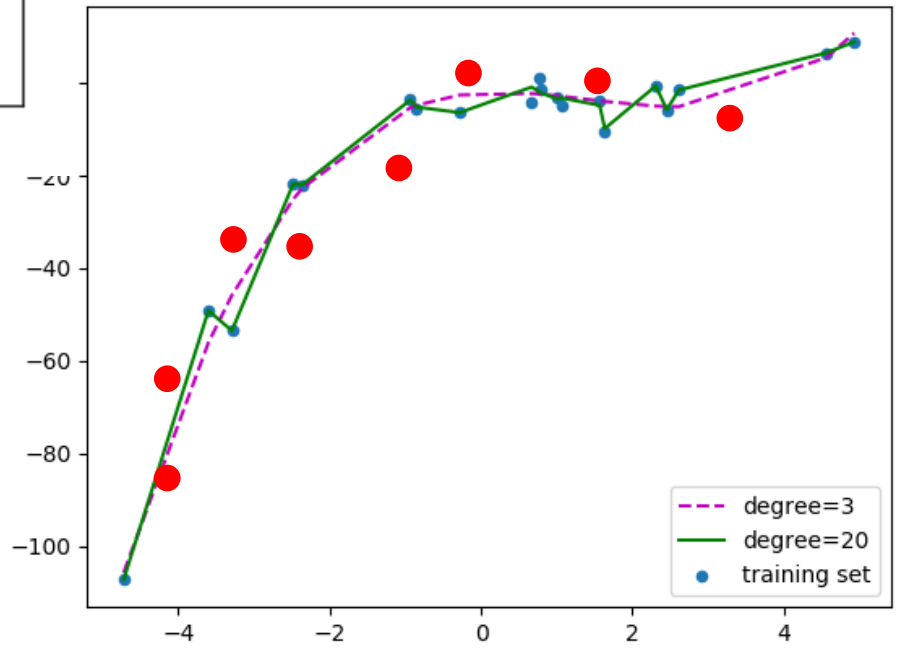
Under-Fitting



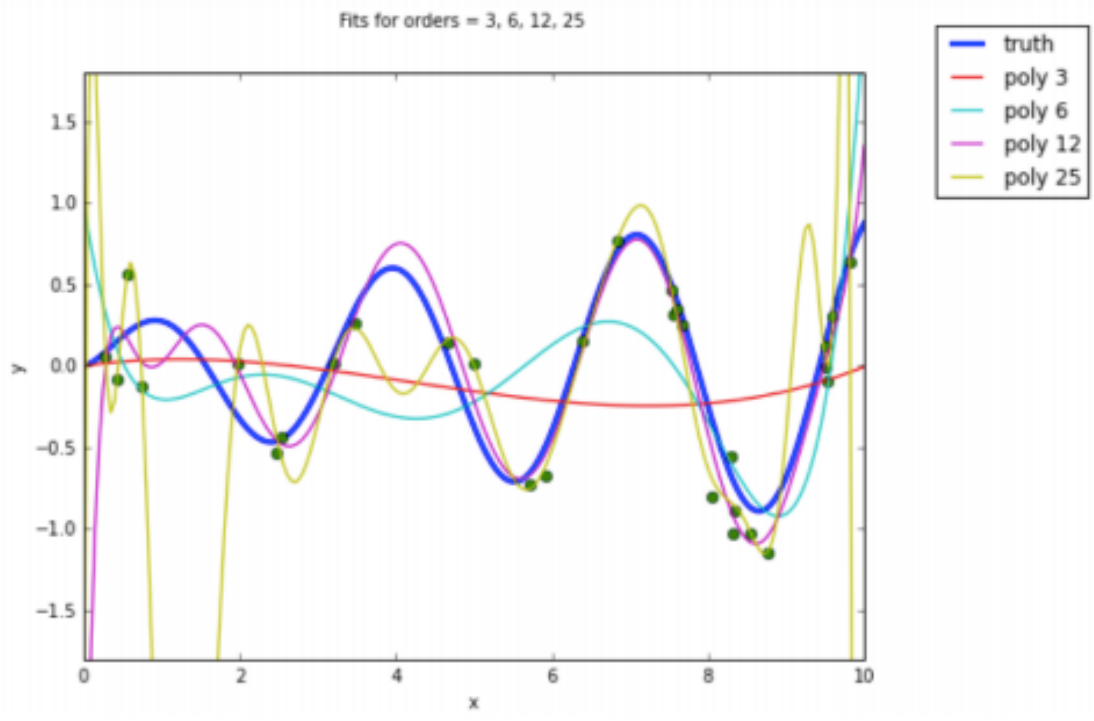
Underfitting and Overfitting



Over-Fitting



Overfitting



The more parameters a model has, the more complicated a curve it can fit.

- If we don't have very much data and we try to fit a complicated model to it, the model will make wild predictions.
- This phenomenon is referred to as overfitting

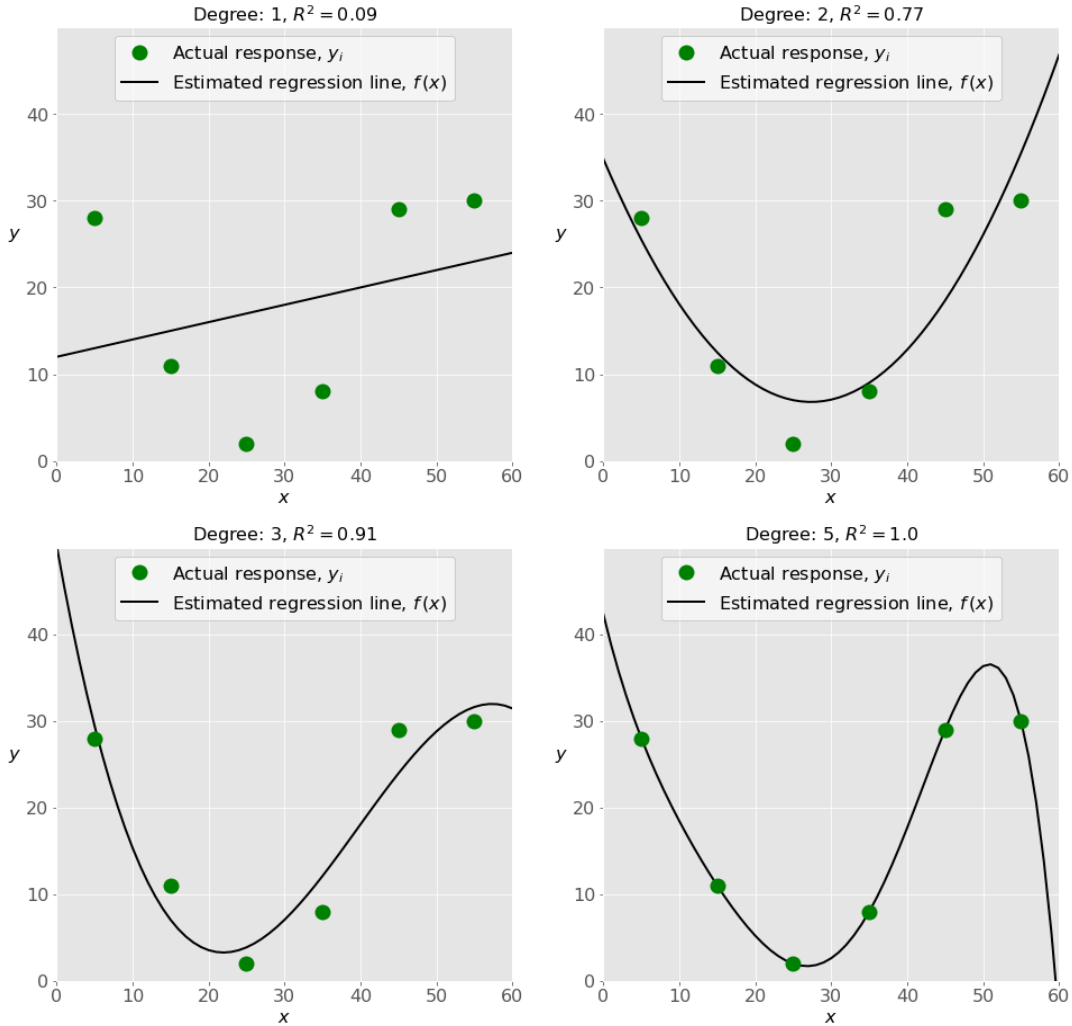
Overfitting, cont.

- Small polynomial; cannot fit the data well; said to have high bias
- Large polynomial; can fit the data well; fits the data too well; said to have small bias
- If there is known error in the data, then a close fit is wasted:
    - 25-th degree polynomial does all sorts of wild contortions!
- Poor fit due to high bias called underfitting
- Poor fit due to low bias called overfitting

FLUX Question

FLUX code: 5NKWED

Which model could be well-fitted?

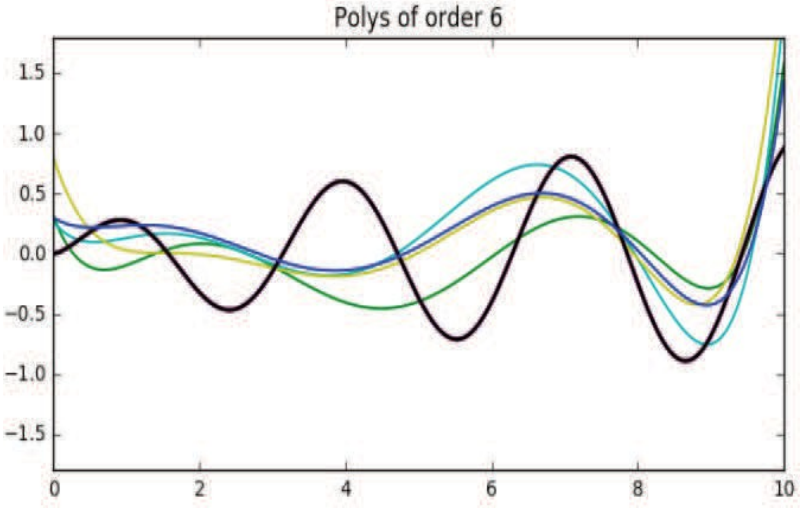


Theory of Data Analysis, cont.
Bias and Variance

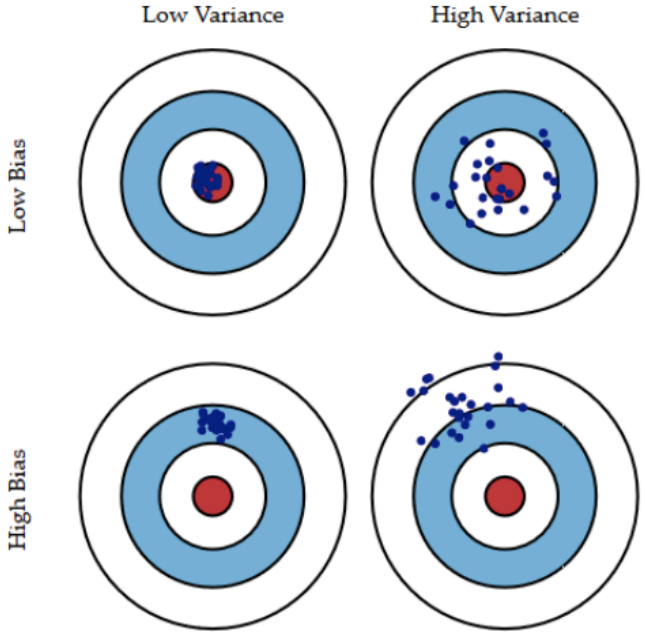
Training Set and Test Set

- Split up the data we have into two non-overlapping parts, a training set and a test set
- Do your learning, run your algorithm, build your model using the training set
- Run evaluation using the test set
- Don’t run evaluation on the training set
- How big to make the test set?

Bias and Variance



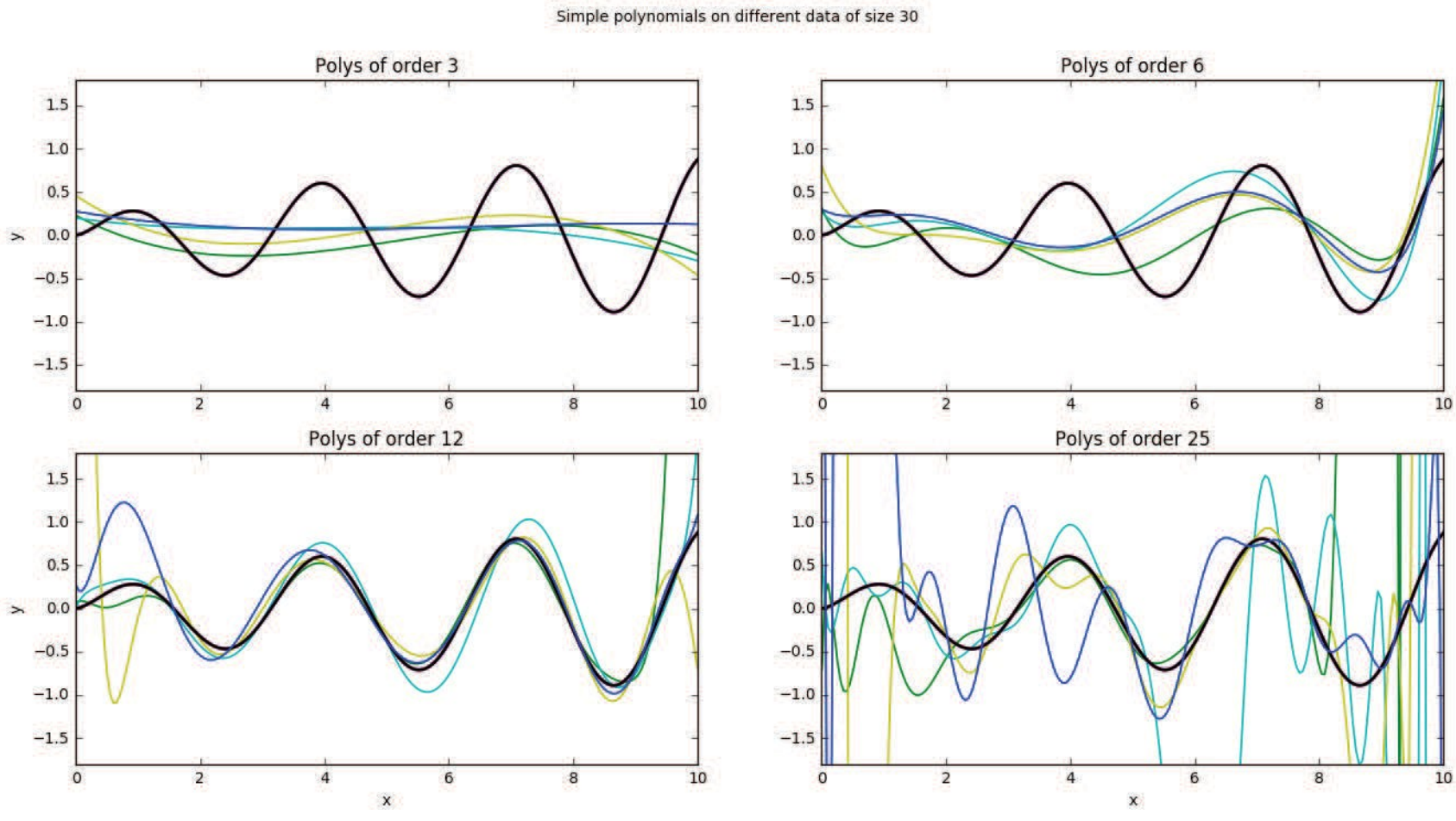
Different data sets of size 30.



Bias: measures how much the prediction differs from the desired regression function.

Variance: measures how much the predictions for individual data sets vary around their average.

Bias-Variance Examples



FLUX Question

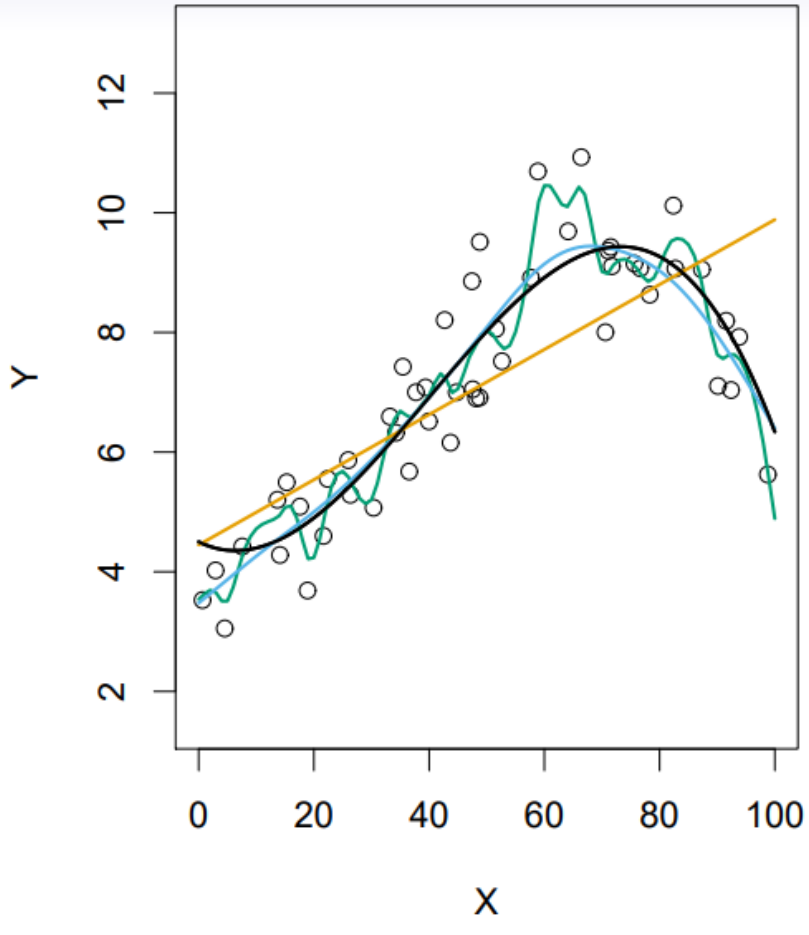
Which of the polynomials in the previous slide is a better model?

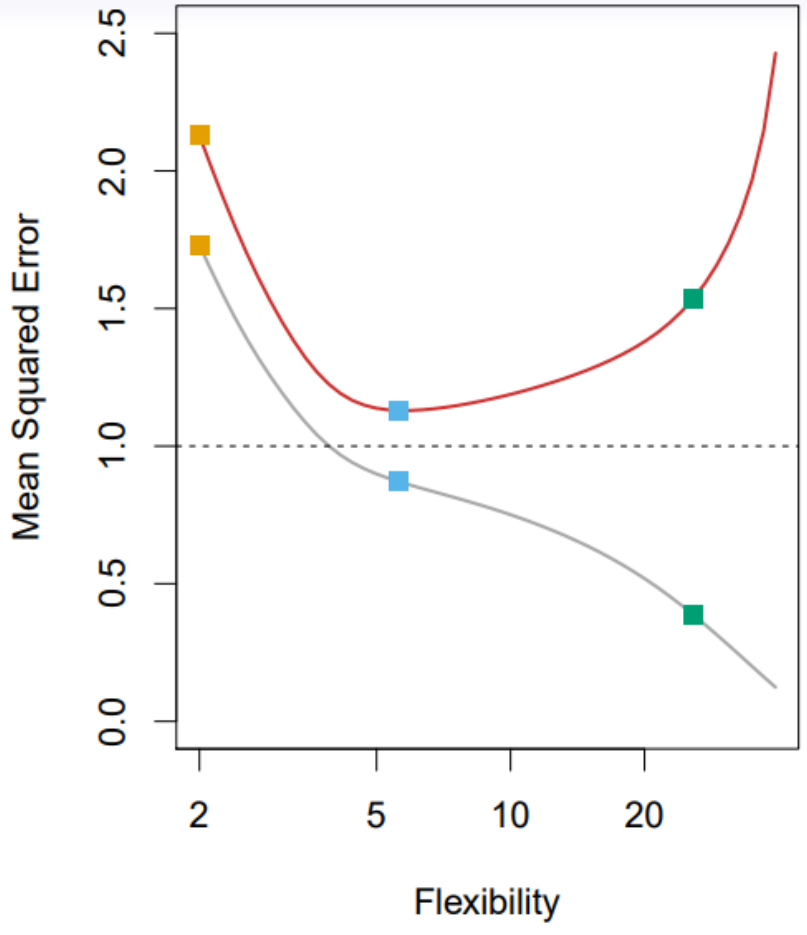
A. Order 3
B. Order 6
C. Order 12
D. Order 25

FLUX code: 5NKWED

Bias vs Variance Trade-off

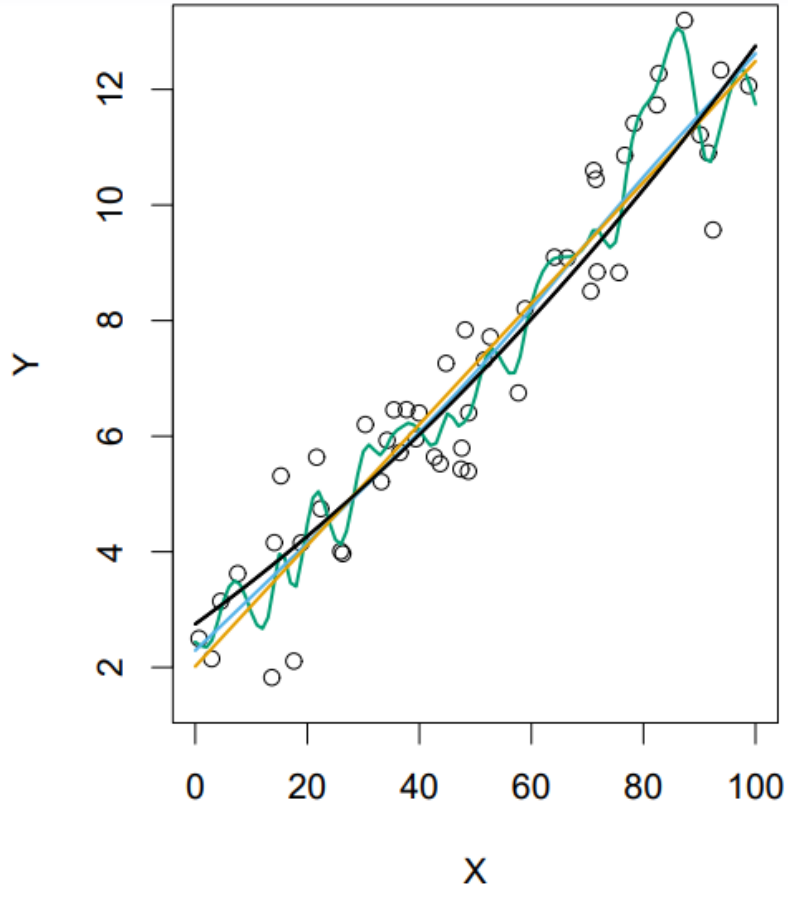
Scenario 1

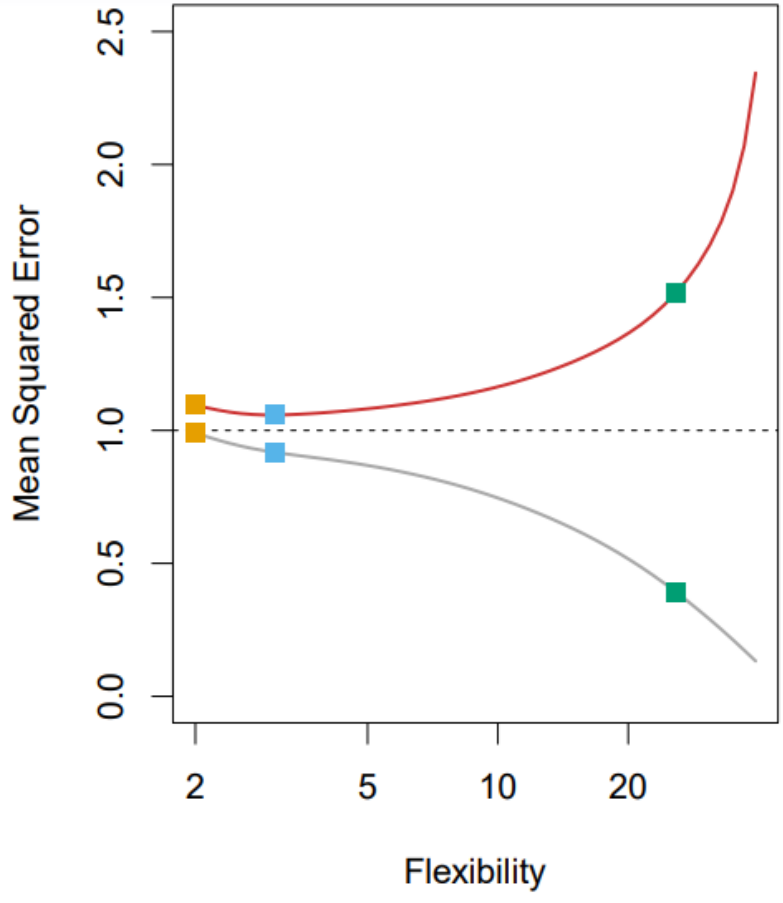




Bias vs Variance Trade-off

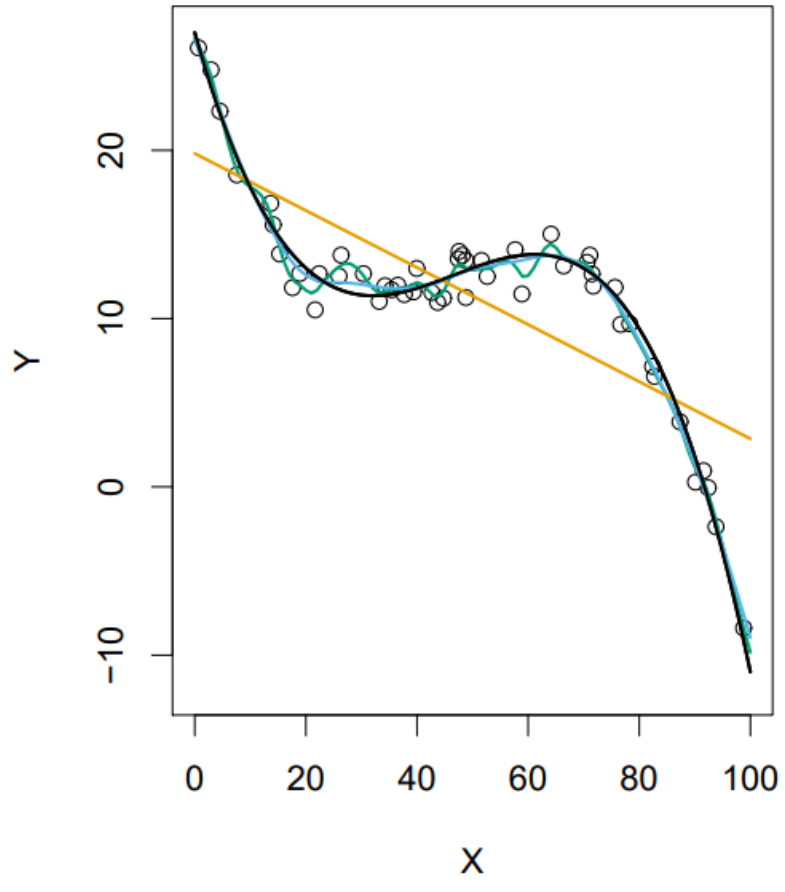
Scenario 2

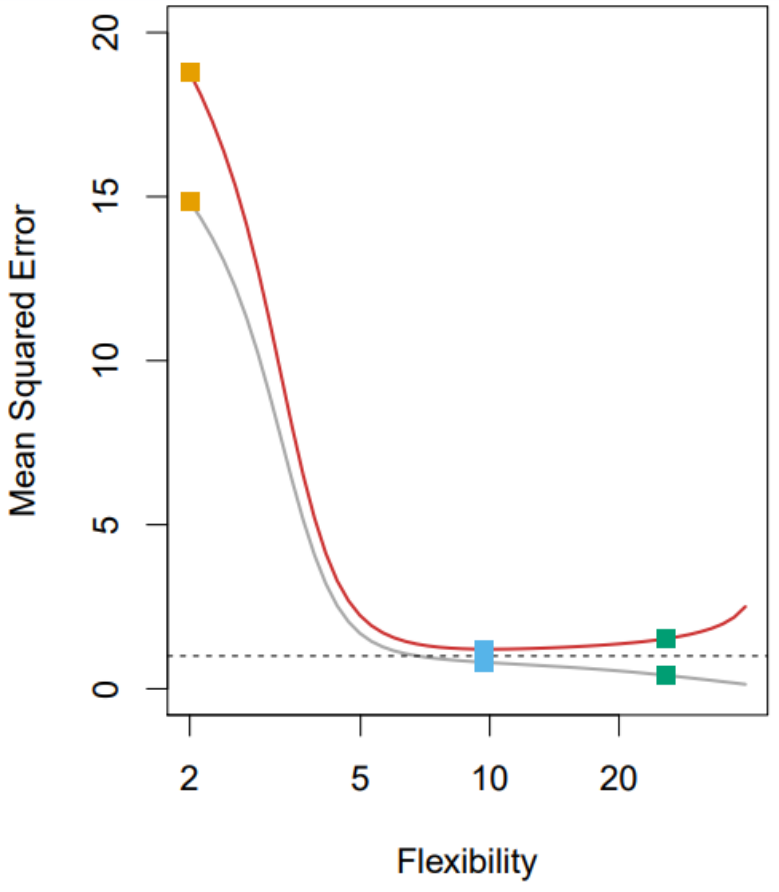




Bias vs Variance Trade-off

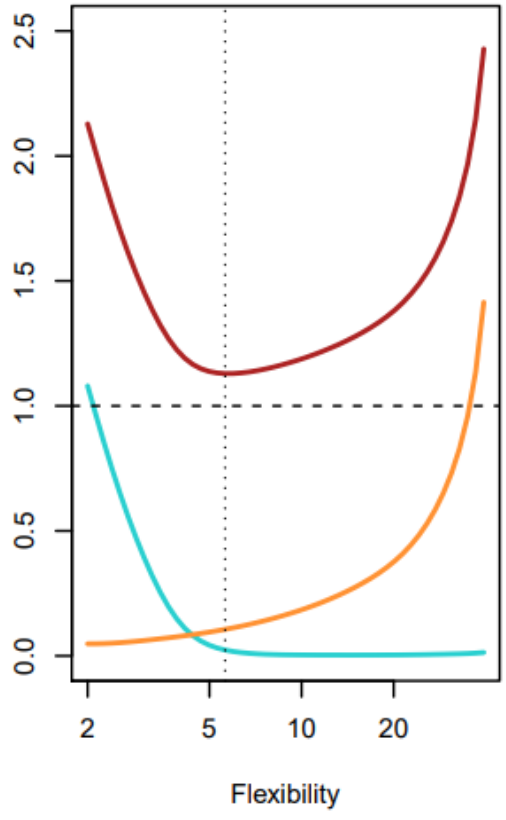
Scenario 3

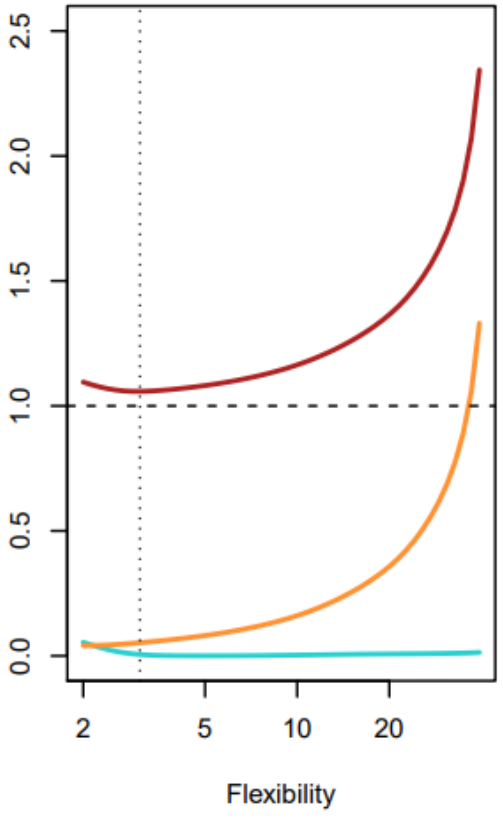


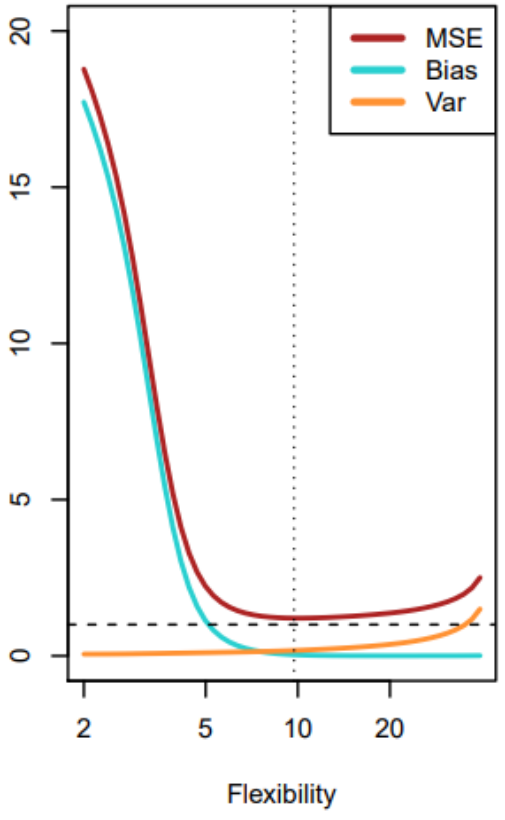


Bias vs Variance Trade-off

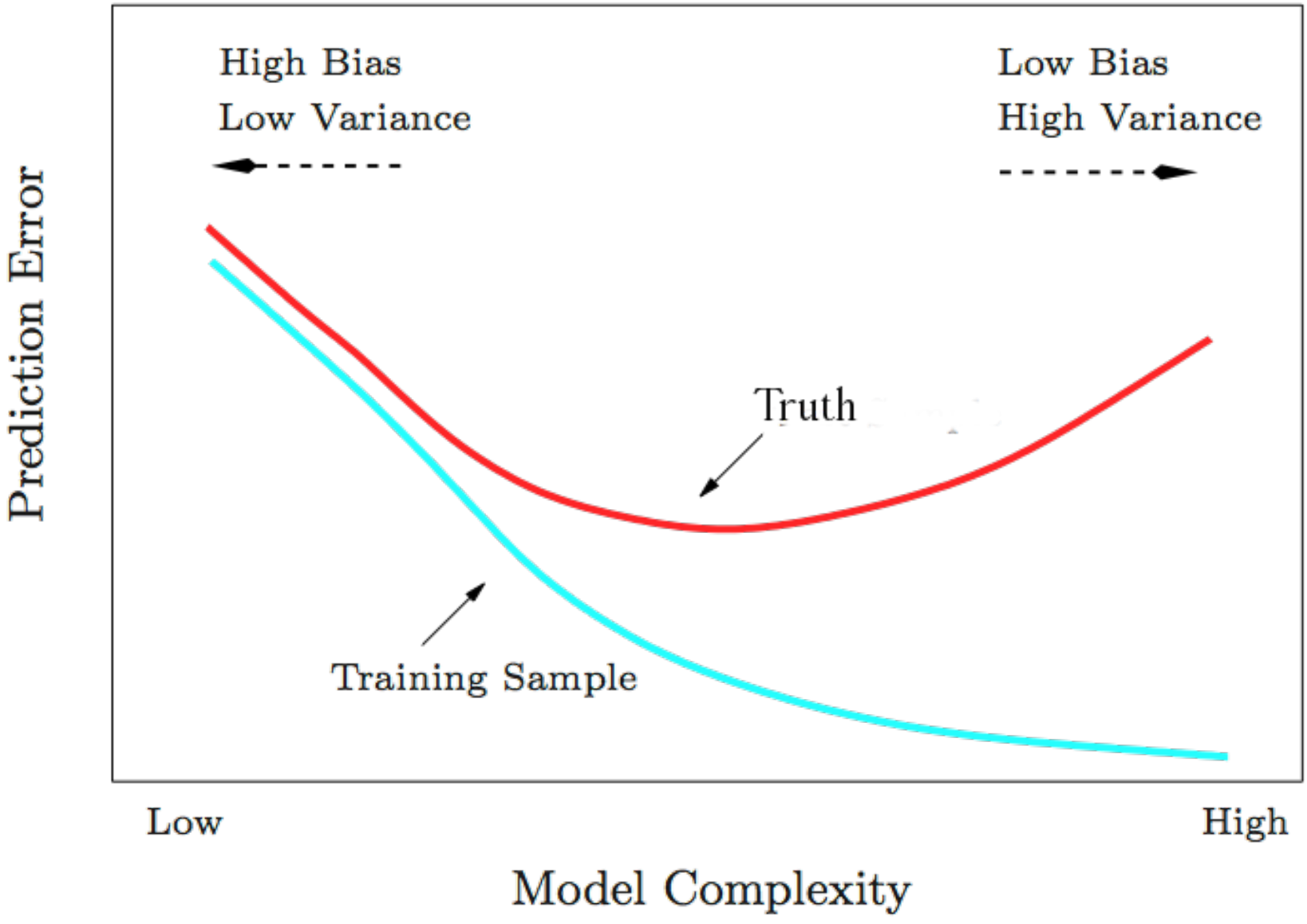
Optimum Degree







Bias-Variance Tradeoff



No Free Lunch Theorem

Wolpert and McCready proved:

> *if a [learning] algorithm performs well on a certain class of problems then it necessarily pays for that with degraded performance on the set of all remaining problems*

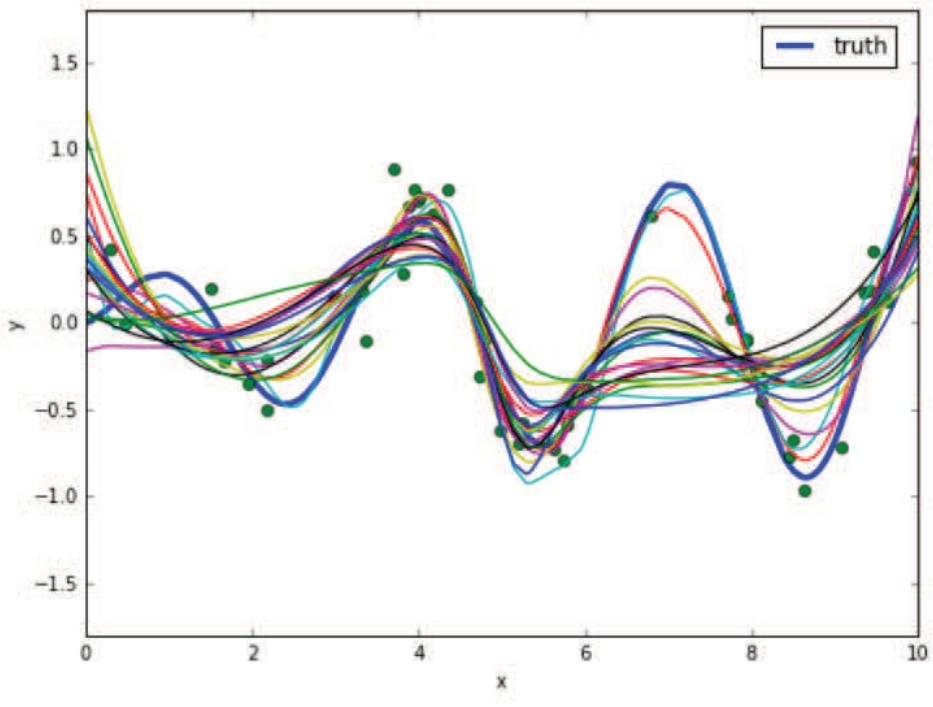
- there is no universally good machine learning algorithm (when one has finite data)

e.g. Naive Bayesian classification performs well for text classification with smaller data sets

e.g. linear Support Vector Machines perform well for text classification

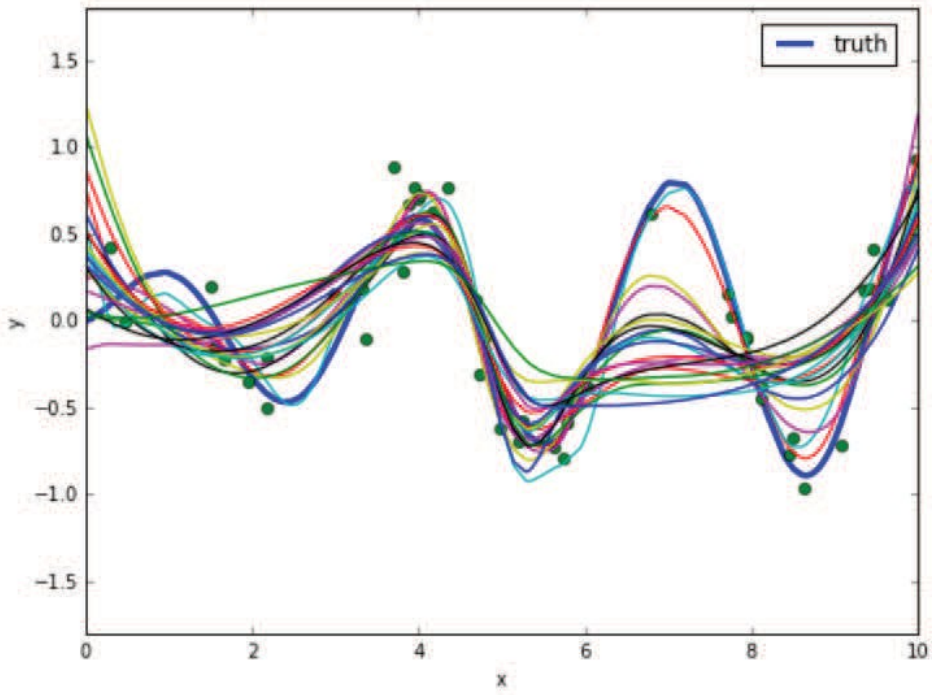
Ensembles

- given only data, we do not know the truth and can only estimate what may be the “truth”
- an ensemble is a collection of possible/reasonable models
- from this we can understand the variability and range of predictions that is realistic

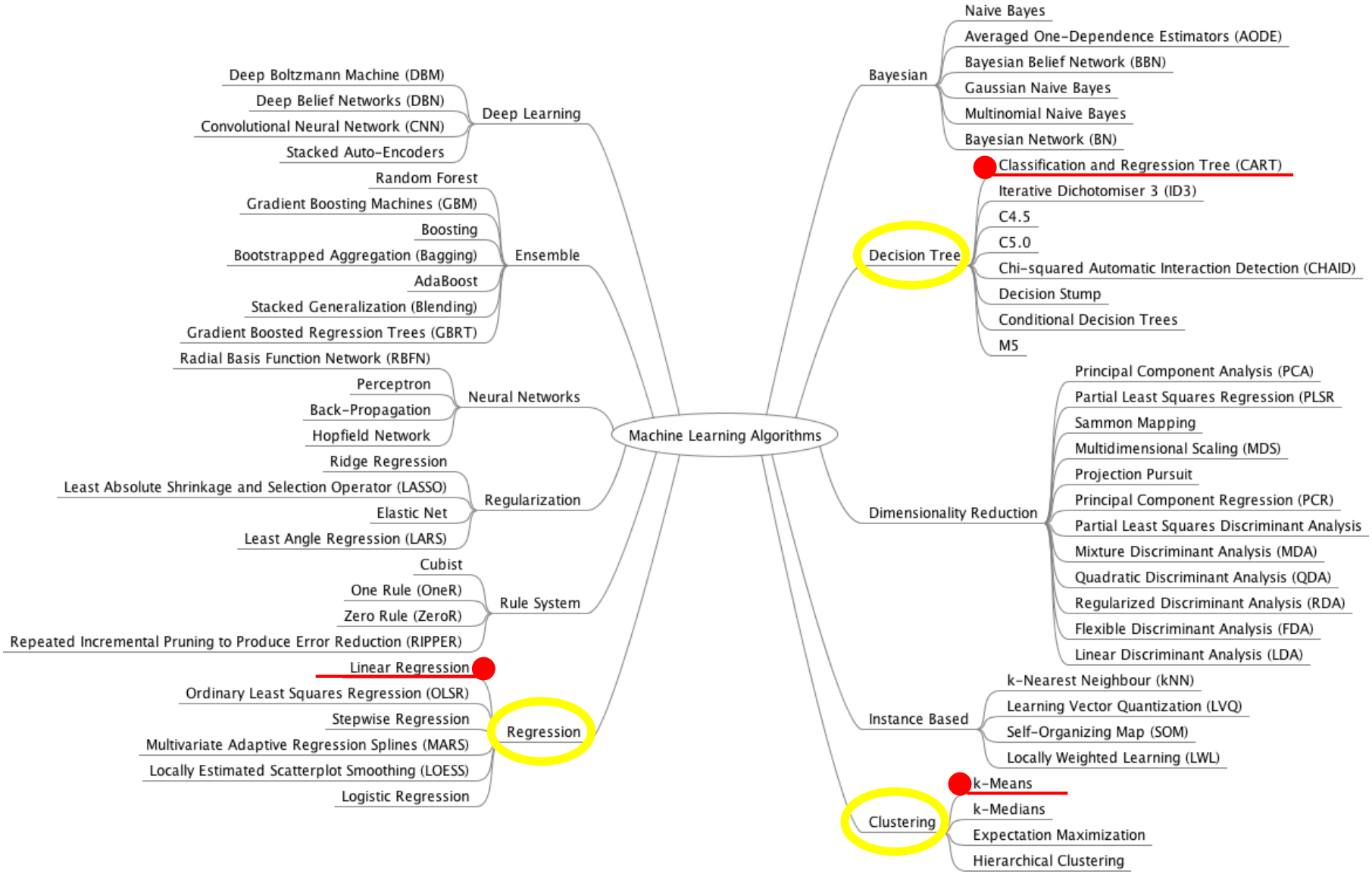


Ensembles (cont.)

- generating an ensemble is a whole statistical subject in itself
- often we average the predictions over the models in an ensemble to improve performance $\hat{y}(x) = \frac{1}{M} \sum_{i=1}^{M} \hat{y}^{(i)}(x)$



Machine Learning Algorithms



Tutorial/Lab week 6

- Regression in Python
    - Examine the bias of the linear regression model
    - Illustrates the fitting of a different type of model
    - Illustrates ensembles