EE522: SELECTED TOPICS IN QUANTITATIVE ECONOMICS 2

Wasin Siwasarit
Faculty of Economics,Thammasat University

August 8, 2022

THAMMASAT

Battle Plan of Today's Lecture

Course Syllabus

What's Python?

Setting up Your Python Environment

Problems Machine Learning Can Solve

Getting Started with Python Machine Learning

Outline

THAMMASAT

Technical requirement: Students should bring their own laptops.

Course Syllabus

THAMMASAT

Outline

THAMMASAT

Python is a general-purpose programming language conceived in 1989 by Dutch programmer Guido van Rossum.

About Python

Outline

Anaconda

Setting up Your Python Environment

THAMMASAT

Outline

THAMMASAT

Supervised Learning

The user provides the algorithm with pairs of inputs and desired outputs, and the algorithm finds a way to produce the desired output given an input.

Examples: Supervised Learning

Identifying the zip code from handwritten digits on an envelope
Here the input is a scan of the handwriting, and the desired output is the actual digits in the zip code. To create a dataset for building a machine learning model, you need to collect many envelopes. Then you can read the zip codes yourself and store the digits as your desired outcomes.

Examples: Supervised Learning

Detecting fraudulent activity in credit card transactions
Here the input is a record of the credit card transaction, and the output is whether it is likely to be fraudulent or not. Assuming that you are the entity distributing the credit cards, collecting a dataset means storing all transactions and recording if a user reports any transaction as fraudulent.

Unsupervised Learning

They are the other type of algorithm that we will cover in this class. In unsupervised learning, only the input data is known, and no known output data is given to the algorithm. While there are many successful applications of these methods, they are usually harder to understand and evaluate.

Example : Unsupervised Learning

Segmenting customers into groups with similar preferences
Given a set of customer records, you might want to identify which customers are similar, and whether there are groups of customers with similar preferences. For a shopping site, these might be "parents," "bookworms," or "gamers." Because you don't know in advance what these groups might be, or even how many there are, you have no known outputs.

Outline

THAMMASAT

Getting Started with Python Machine Learning

- The purpose of this lecture is to introduce the basic idea of machine learning with a very simple example. Despite its simplicity, it will challenge us with the risk of overfitting.

Main Objectives

- Reading in the data and cleaning it .

Main Objectives

- Reading in the data and cleaning it .
- Exploring and understanding the input data.

Main Objectives

- Reading in the data and cleaning it .
- Exploring and understanding the input data.
- Analyzing how best to present the data to the learning algorithm.

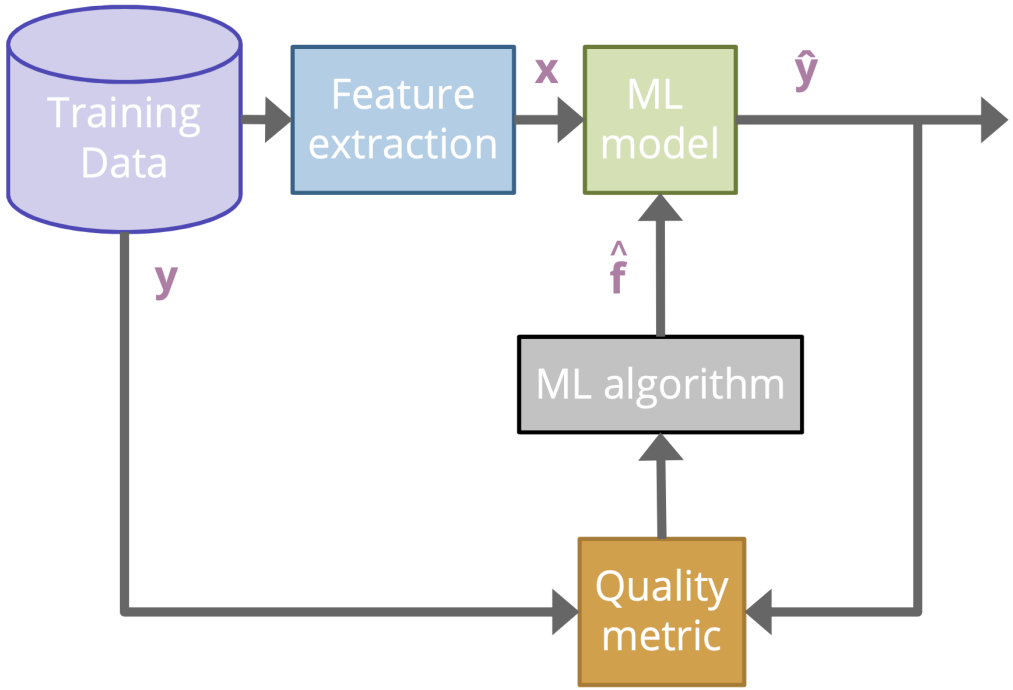
Main Objectives

- Reading in the data and cleaning it .
- Exploring and understanding the input data.
- Analyzing how best to present the data to the learning algorithm.
- Choosing the right model and learning algorithm.

Main Objectives

- Reading in the data and cleaning it .
- Exploring and understanding the input data.
- Analyzing how best to present the data to the learning algorithm.
- Choosing the right model and learning algorithm.
- Measuring the performance correctly.

Main Objectives



THAMMASAT

Trade-off

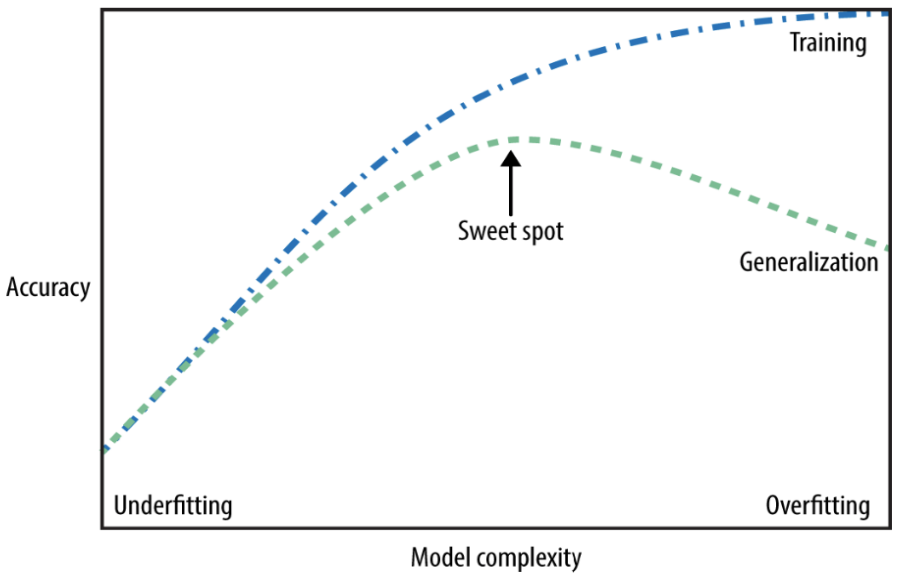


Figure 2-1. Trade-off of model complexity against training and test accuracy

THAMMASAT

Trade-off

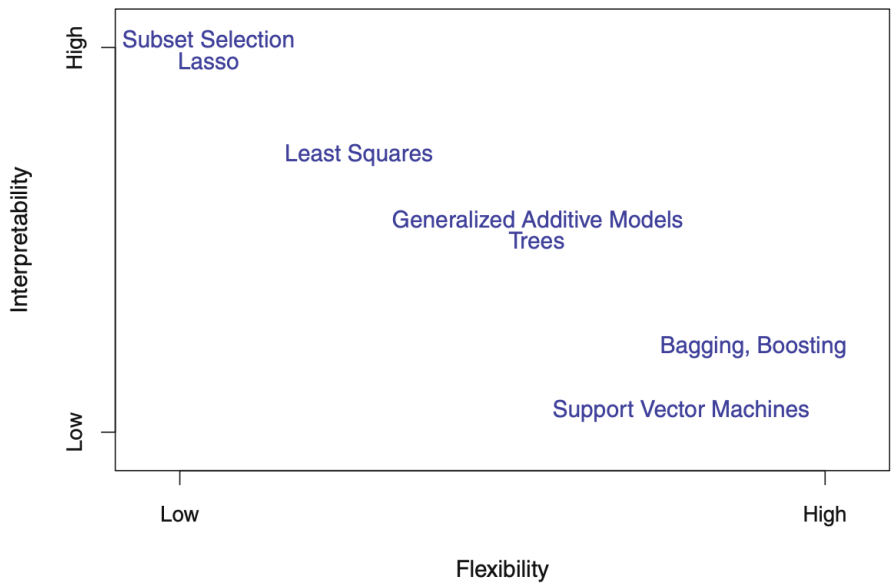


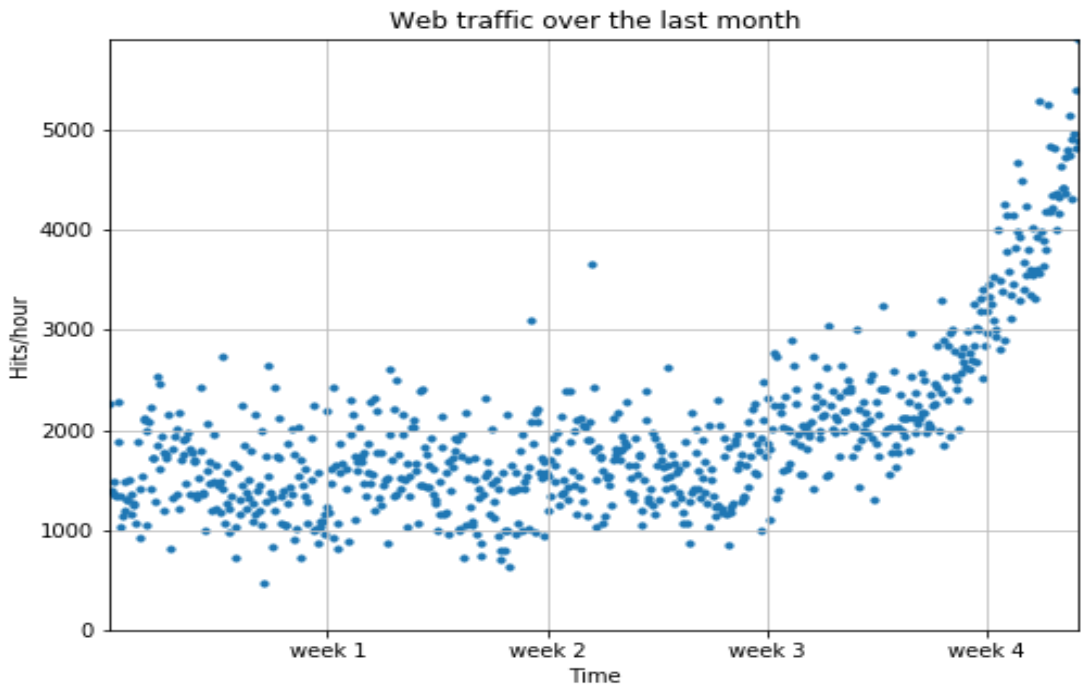
FIGURE 2.7. *A representation of the tradeoff between flexibility and interpretability, using different statistical learning methods. In general, as the flexibility of a method increases, its interpretability decreases.*

THAMMASAT

Python Codes: Example1-1

```
import os
from utils import DATA_DIR, CHART_DIR
import scipy as sp
import matplotlib.pyplot as plt

sp.random.seed(3)  # to reproduce the data later on

data = sp.genfromtxt(os.path.join(DATA_DIR, "web_traffic.tsv"), delimiter="\t")
print(data[:10])
print(data.shape)
```

As seen,we can see that while in the first weeks the traffic stayed more or less the same, the last week shows a steep increase.



Choosing the right model and learning algorithm

- Find the real model behind the noisy data points .

Choosing the right model and learning algorithm

- Find the real model behind the noisy data points .
- Following this, use the model to extrapolate into the future to find the point in time where our infrastructure has to be extended.

Python Codes: Example1-2

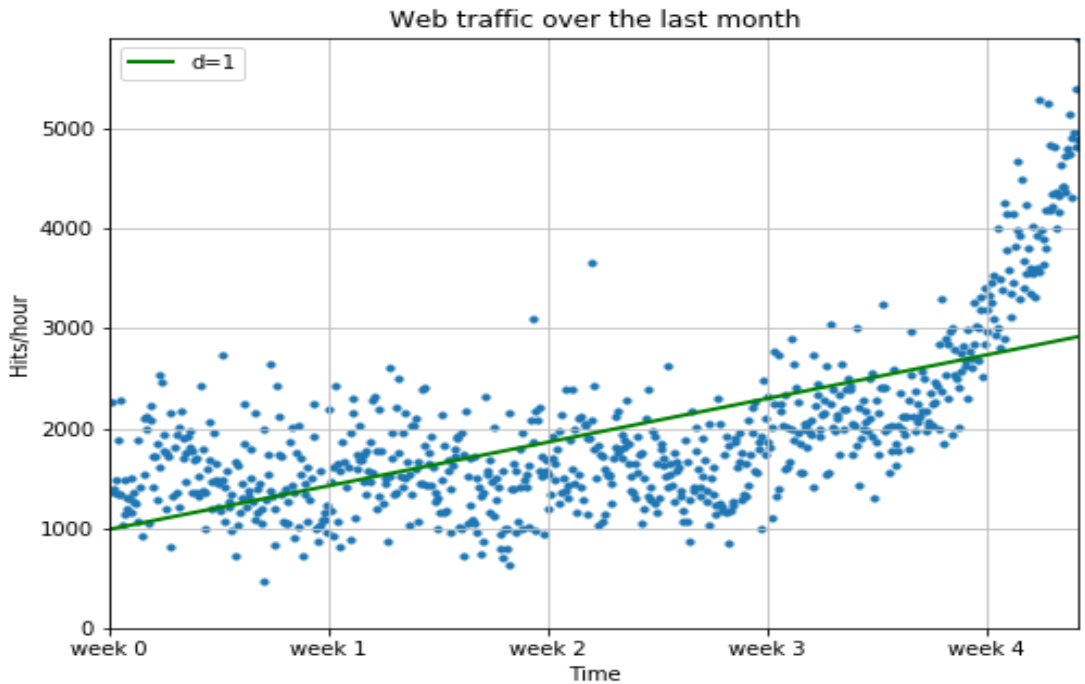
```
# all examples will have three classes in this file
colors = ['g', 'k', 'b', 'm', 'r']
linestyles = ['-', '-.', '--', ':', '-']

x = data[:, 0]
y = data[:, 1]
print("Number of invalid entries:", sp.sum(sp.isnan(y)))
x = x[~sp.isnan(y)]
y = y[~sp.isnan(y)]


def plot_models(x, y, models, fname, mx=None, ymax=None, xmin=None):
    ''' plot input data '''

    plt.figure(num=None, figsize=(8, 6))
    plt.clf()
    plt.scatter(x, y, s=10)
    plt.title("Web traffic over the last month")
    plt.xlabel("Time")
    plt.ylabel("Hits/hour")
    plt.xticks(
        [w * 7 * 24 for w in range(10)], ['week %i' % w for w in range(10)])
```

Python Codes: Example1-3

```
    if models:
        if mx is None:
            mx = sp.linspace(0, x[-1], 1000)
        for model, style, color in zip(models, linestyles, colors):
            # print "Model:",model
            # print "Coeffs:",model.coeffs
            plt.plot(mx, model(mx), linestyle=style, linewidth=2, c=color)

        plt.legend(["d=%i" % m.order for m in models], loc="upper left")

    plt.autoscale(tight=True)
    plt.ylim(ymin=0)
    if ymax:
        plt.ylim(ymax=ymax)
    if xmin:
        plt.xlim(xmin=xmin)
    plt.grid(True, linestyle='-', color='0.75')
    plt.savefig(fname)

# first look at the data
plot_models(x, y, None, os.path.join(CHART_DIR, "1.png"))
```

THAMMASAT

Starting with a simple straight line, we clearly see that there is something wrong with our initial assumption that the underlying model is a straight line.

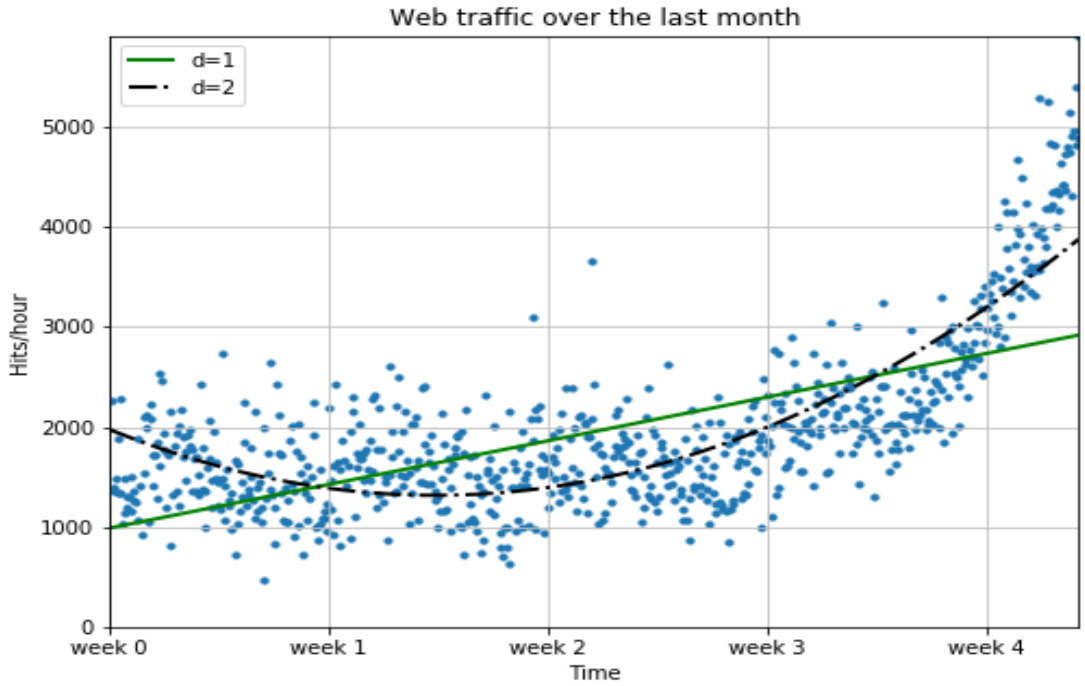


THAMMASAT

Python Codes: Example1-4

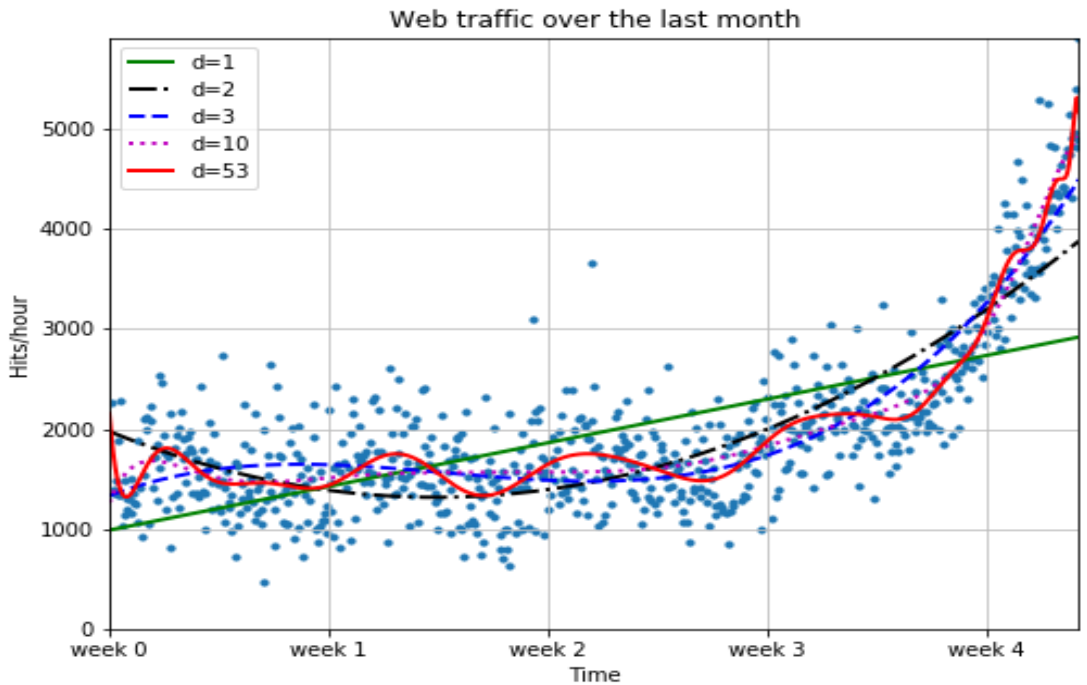
```
# create and plot models
fp1, res1, rank1, sv1, rcond1 = sp.polyfit(x, y, 1, full=True)
print("Model parameters of fp1: %s" % fp1)
print("Error of the model of fp1:", res1)
f1 = sp.poly1d(fp1)

fp2, res2, rank2, sv2, rcond2 = sp.polyfit(x, y, 2, full=True)
print("Model parameters of fp2: %s" % fp2)
print("Error of the model of fp2:", res2)
f2 = sp.poly1d(fp2)
f3 = sp.poly1d(sp.polyfit(x, y, 3))
f10 = sp.poly1d(sp.polyfit(x, y, 10))
f100 = sp.poly1d(sp.polyfit(x, y, 100))

plot_models(x, y, [f1], os.path.join(CHART_DIR, "2.png"))
plot_models(x, y, [f1, f2], os.path.join(CHART_DIR, "3.png"))
plot_models(x, y, [f1, f2, f3, f10, f100], os.path.join(CHART_DIR, "4.png"))
```

The error is almost half the error of the straight line model. This is good but unfortunately this comes with a price.



THAMMASAT

Looking at the polynomial of degree 10 and 53, we see wildly oscillating behavior. It seems that the models are fitted too much to the data. So much that it is now capturing not only the underlying process but also the noise. This is called overfitting.

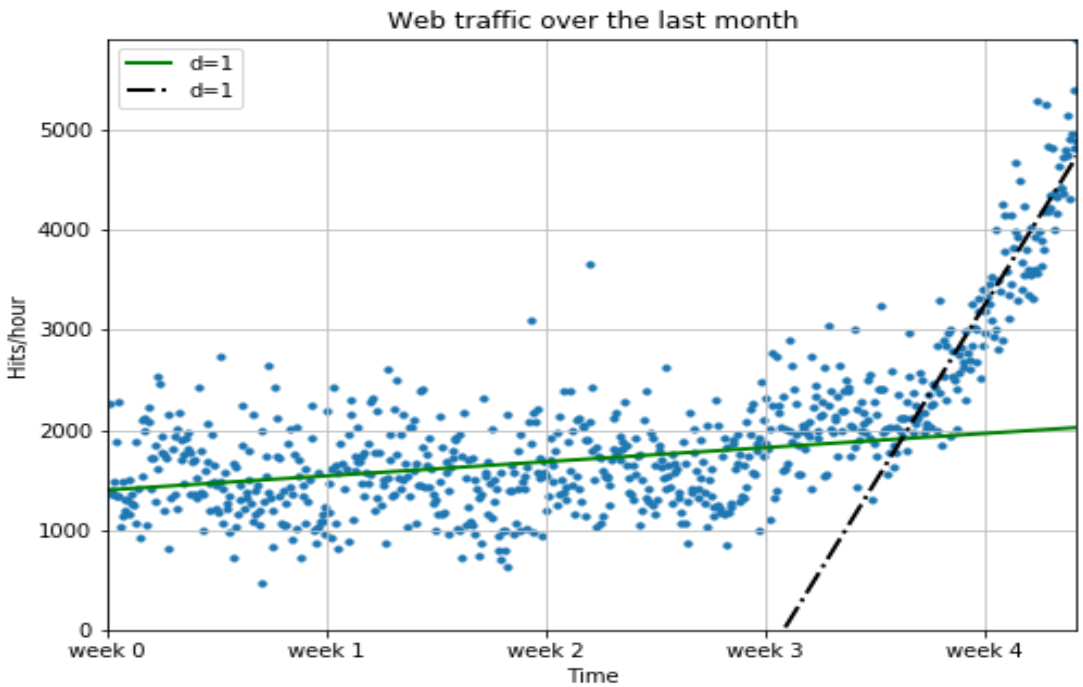


THAMMASAT

Python Codes: Example1-5

```
# fit and plot a model using the knowledge about inflection point
inflection = 3.5 * 7 * 24
xa = x[:inflection]
ya = y[:inflection]
xb = x[inflection:]
yb = y[inflection:]

fa = sp.poly1d(sp.polyfit(xa, ya, 1))
fb = sp.poly1d(sp.polyfit(xb, yb, 1))

plot_models(x, y, [fa, fb], os.path.join(CHART_DIR, "5.png"))

def error(f, x, y):
    return sp.sum((f(x) - y) ** 2)

print("Errors for the complete data set:")
for f in [f1, f2, f3, f10, f100]:
    print("Error d=%i: %f" % (f.order, error(f, x, y)))

print("Errors for only the time after inflection point")
for f in [f1, f2, f3, f10, f100]:
    print("Error d=%i: %f" % (f.order, error(f, xb, yb)))

print("Error inflection=%f" % (error(fa, xa, ya) + error(fb, xb, yb)))
```

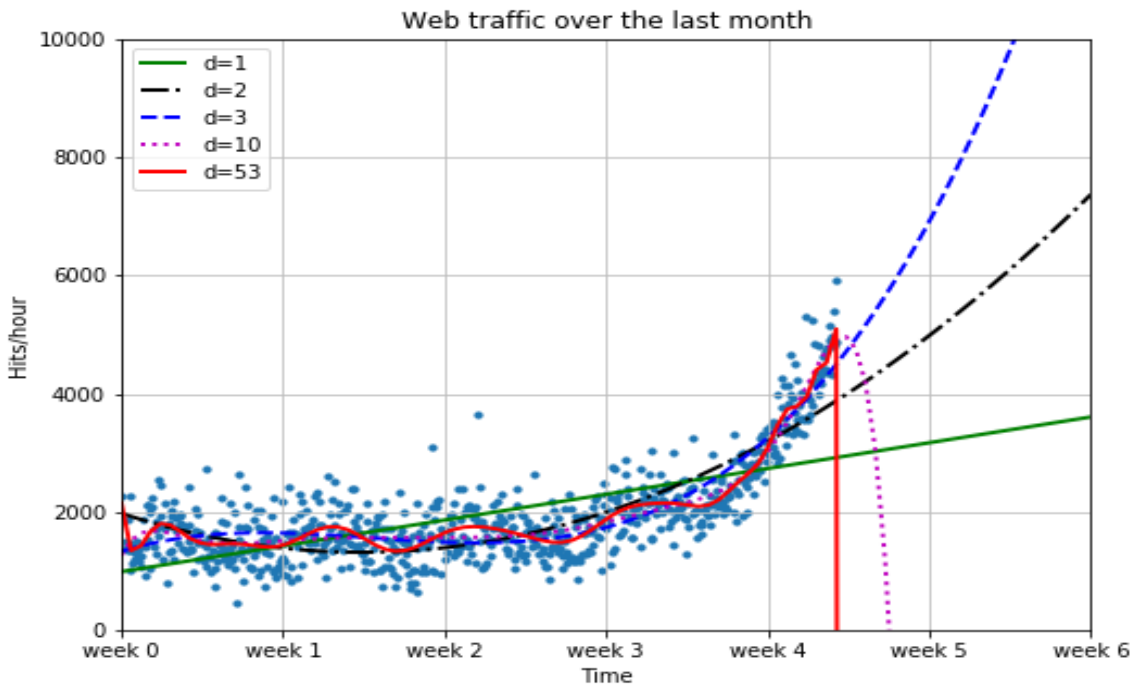
The combination of these two lines seems to be a much better fit to the data than anything we have modeled before. But still, the combined error is higher than the higher order polynomials.



Python Codes: Example1-6

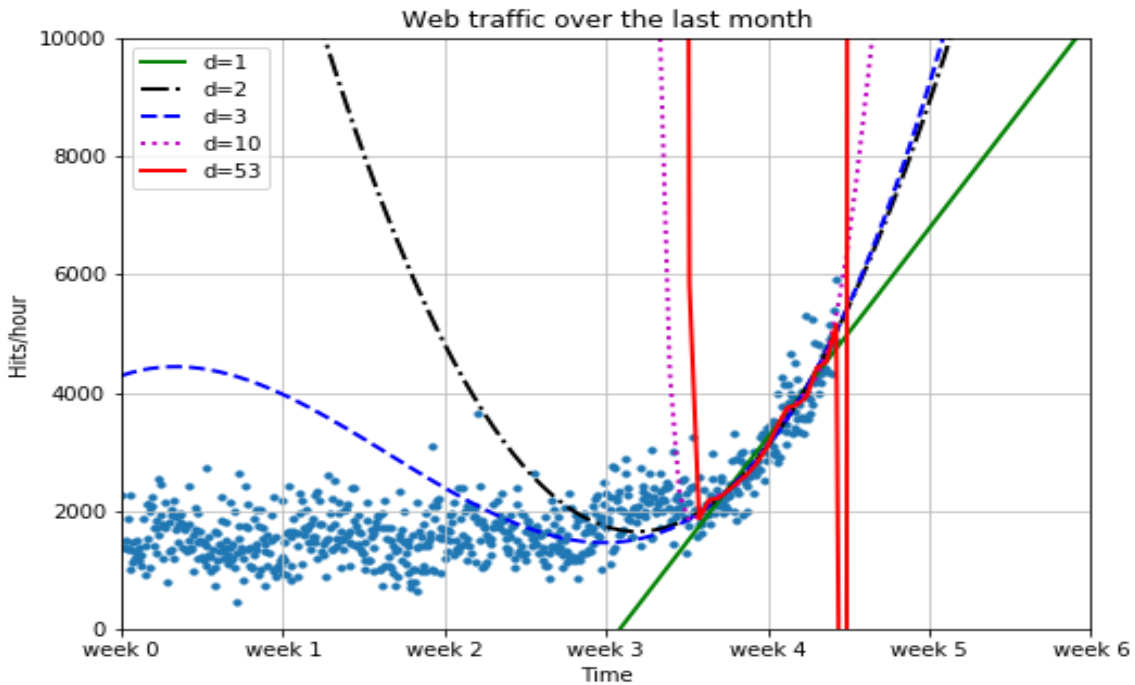
```
# extrapolating into the future
plot_models(
    x, y, [f1, f2, f3, f10, f100],
    os.path.join(CHART_DIR, "6.png"),
    mx=sp.linspace(0 * 7 * 24, 6 * 7 * 24, 100),
    ymax=10000, xmin=0 * 7 * 24)

print("Trained only on data after inflection point")
fb1 = fb
fb2 = sp.poly1d(sp.polyfit(xb, yb, 2))
fb3 = sp.poly1d(sp.polyfit(xb, yb, 3))
fb10 = sp.poly1d(sp.polyfit(xb, yb, 10))
fb100 = sp.poly1d(sp.polyfit(xb, yb, 100))

print("Errors for only the time after inflection point")
for f in [fb1, fb2, fb3, fb10, fb100]:
    print("Error d=%i: %f" % (f.order, error(f, xb, yb)))

plot_models(
    x, y, [fb1, fb2, fb3, fb10, fb100],
    os.path.join(CHART_DIR, "7.png"),
    mx=sp.linspace(0 * 7 * 24, 6 * 7 * 24, 100),
    ymax=10000, xmin=0 * 7 * 24)
```

The models of degree 10 and 53 don't seem to expect a bright future of our start-up. They tried so hard to model the given data correctly that they are clearly useless to extrapolate beyond.



THAMMASAT

Only to the data of the last week,we believe that the last week says more about the future than the data prior to it. The result can be seen in the following psychedelic chart, which further shows how badly the problem of overfitting is.

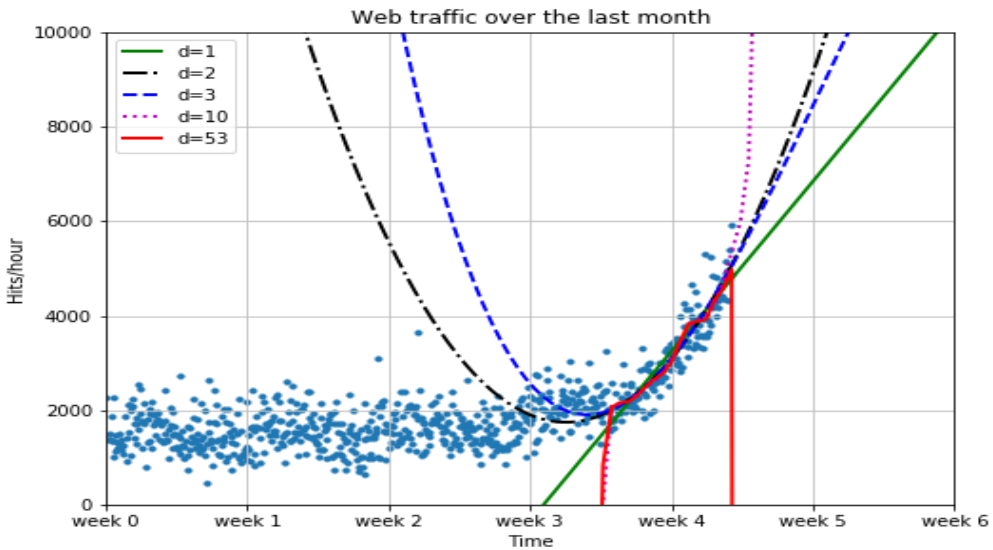


THAMMASAT

Python Codes: Example1-7

```
# separating training from testing data
frac = 0.3
split_idx = int(frac * len(xb))
shuffled = sp.random.permutation(list(range(len(xb))))
test = sorted(shuffled[:split_idx])
train = sorted(shuffled[split_idx:])
fbt1 = sp.poly1d(sp.polyfit(xb[train], yb[train], 1))
fbt2 = sp.poly1d(sp.polyfit(xb[train], yb[train], 2))
print("fbt2(x)= \n%s" % fbt2)
print("fbt2(x)-100,000= \n%s" % (fbt2-100000))
fbt3 = sp.poly1d(sp.polyfit(xb[train], yb[train], 3))
fbt10 = sp.poly1d(sp.polyfit(xb[train], yb[train], 10))
fbt100 = sp.poly1d(sp.polyfit(xb[train], yb[train], 100))

print("Test errors for only the time after inflection point")
for f in [fbt1, fbt2, fbt3, fbt10, fbt100]:
    print("Error d=%i: %f" % (f.order, error(f, xb[test], yb[test])))

plot_models(
    x, y, [fbt1, fbt2, fbt3, fbt10, fbt100],
    os.path.join(CHART_DIR, "8.png"),
    mx=sp.linspace(0 * 7 * 24, 6 * 7 * 24, 100),
    ymax=10000, xmin=0 * 7 * 24)

from scipy.optimize import fsolve
print(fbt2)
print(fbt2 - 100000)
reached_max = fsolve(fbt2 - 100000, x0=800) / (7 * 24)
print("100,000 hits/hour expected at week %f" % reached_max[0])
```

Training and testing: Let's remove a certain percentage of the data and train on the remaining one. Then we used the held-out data to calculate the error. As the model has been trained not knowing the held-out data, we should get a more realistic picture of how the model will behave in the future.



THAMMASAT

In this lecture, we have learned:

- A typical machine learning operator, you will spend most of your time in understanding and refining the data.

In this lecture, we have learned:

- A typical machine learning operator, you will spend most of your time in understanding and refining the data.
- We also learned how important it is to have the correct experiment setup and that it is vital to not mix up training and testing .