Introduction to machine learning

Vlad Gladkikh

IBS CMCM

The dream that machines would be able to learn is older than computers themselves.

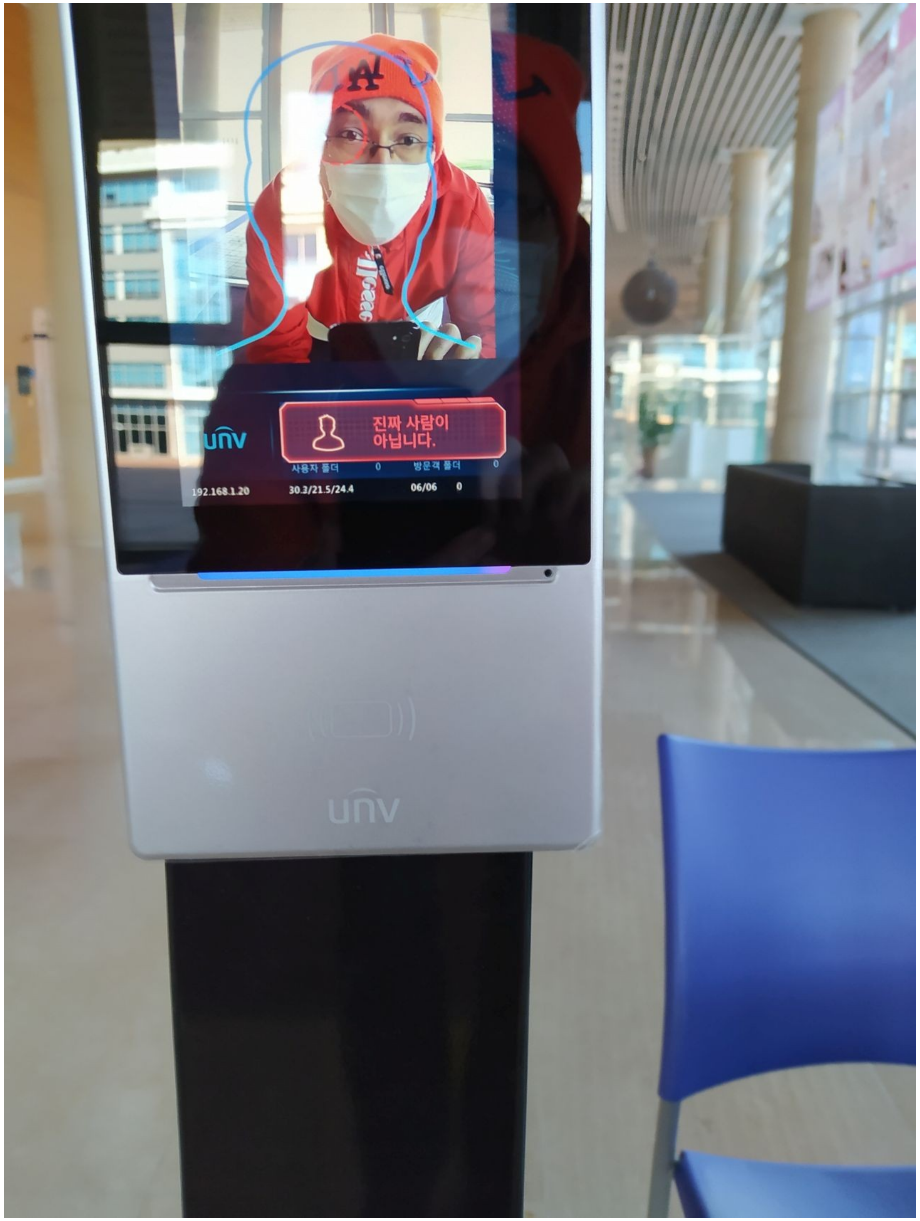
Impossible

Countess of Lovelace. Translator's notes to an article on Babbage's Analytical Engine.
Scientific Memoirs (ed. by R. Taylor), vol. 3 (1842), 691–731.

Possible

Douglas Hartree.
Calculating Instruments and Machines. (1949)

Machine learning is everywhere these days...



Many attempts to find how machines could learn:

Rule-based expert systems

Fuzzy expert systems

Frame-based expert systems

Artificial neural networks

Evolutionary computation

Hybrid intelligent systems

Michael Negnevitsky. Artificial Intelligence. A Guide to Intelligent Systems. (3rd Edition) (2011)

A picture (an example) is worth a thousand words

Try do describe in words how someone looks...

Better to show a few of photos

faster

easier

Our brain picks us the features necessary to recognize the person.

Machine learning:

We supply examples to the machine.

The machine's task is to convert the examples into knowledge.

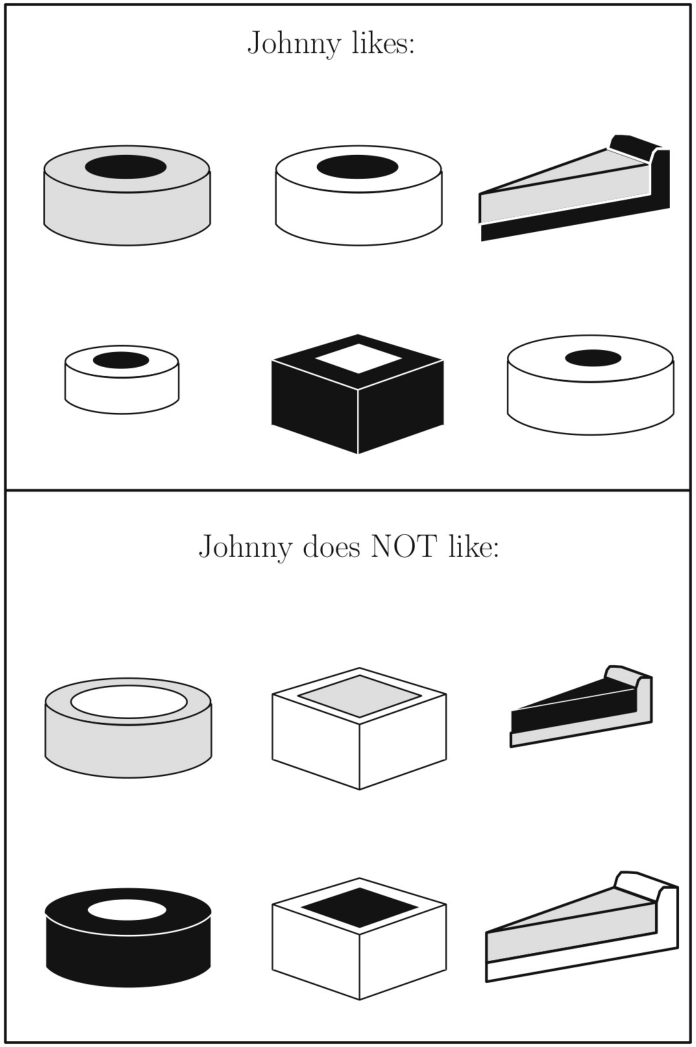
A simple machine-learning task:

Induce a classifier that predicts which pie will Johnny like

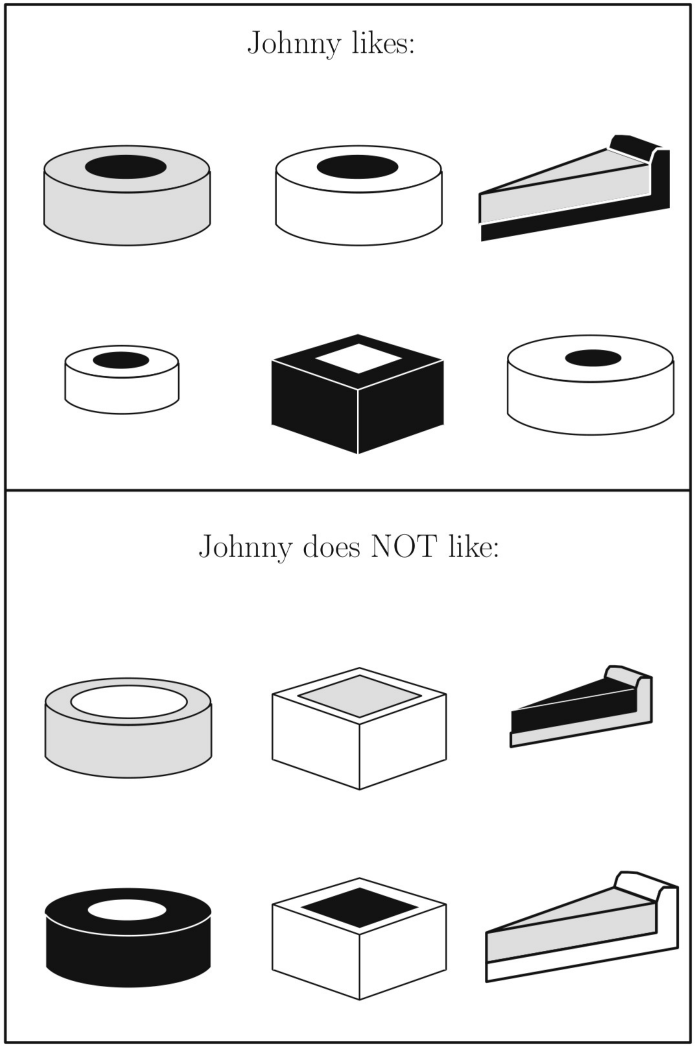
Training set
(pre-classified examples):

positive examples

negative examples



This is **supervised learning**.



Attributes, features, predictors,explanatory variables

Samples

Example	Shape	Crust		Filling		Class
		Size	Shade	Size	Shade	
ex1	Circle	Thick	Gray	Thick	Dark	pos
ex2	Circle	Thick	White	Thick	Dark	pos
ex3	Triangle	Thick	Dark	Thick	Gray	pos
ex4	Circle	Thin	White	Thin	Dark	pos
ex5	Square	Thick	Dark	Thin	White	pos
ex6	Circle	Thick	White	Thin	Dark	pos
ex7	Circle	Thick	Gray	Thick	White	neg
ex8	Square	Thick	White	Thick	Gray	neg
ex9	Triangle	Thin	Gray	Thin	Dark	neg
ex10	Circle	Thick	Dark	Thick	White	neg
ex11	Square	Thick	White	Thick	Dark	neg
ex12	Triangle	Thick	White	Thick	Gray	neg

Label, response variable

Selecting the right features usually takes way longer than all the other ML parts!

Design matrix, model matrix

Perfect classifier:

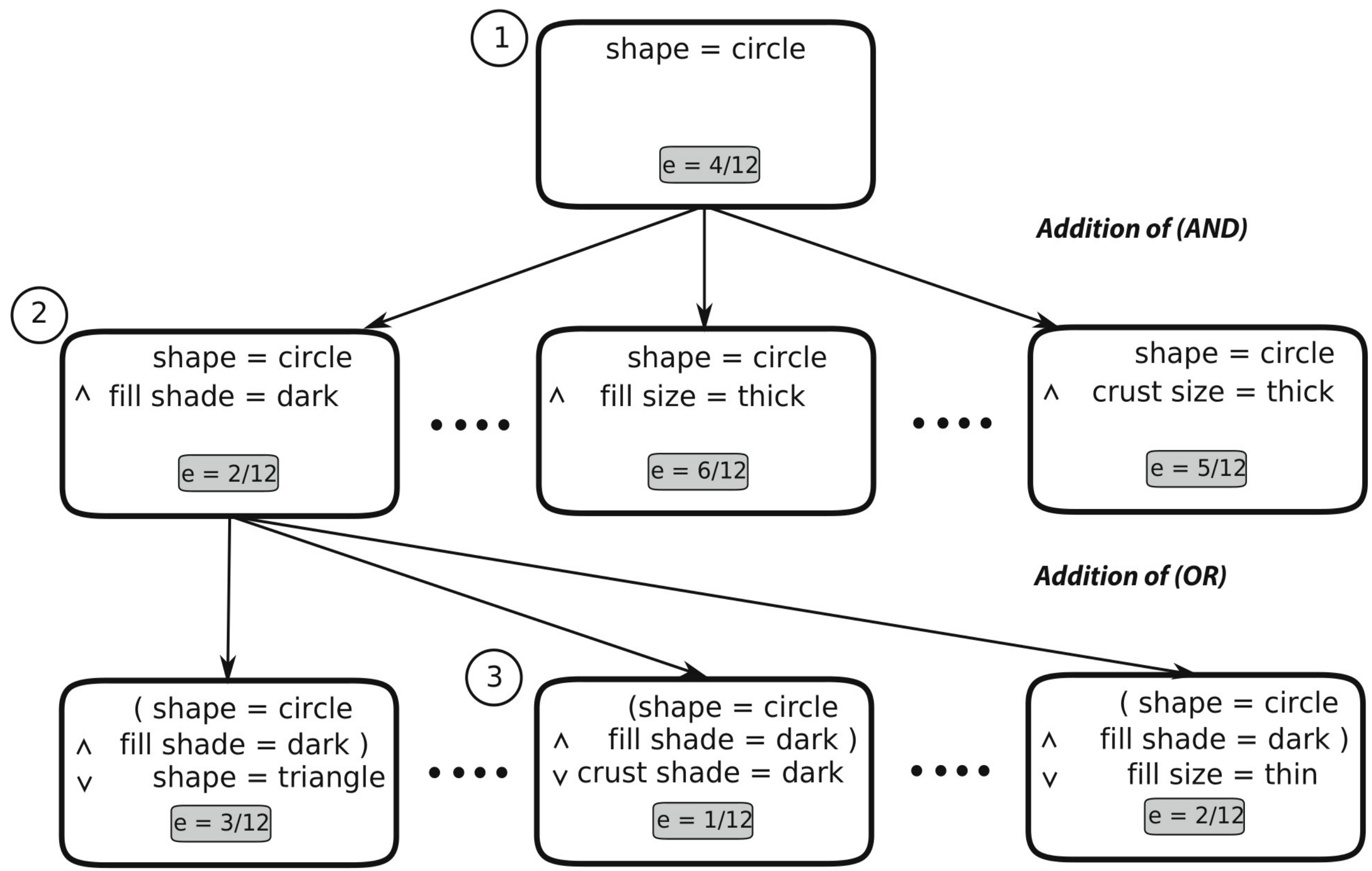
[(shape=circle) ∧ (filling-shade=dark)] ∨
[NOT(shape=circle) ∧ (crust-shade=dark)]

Search

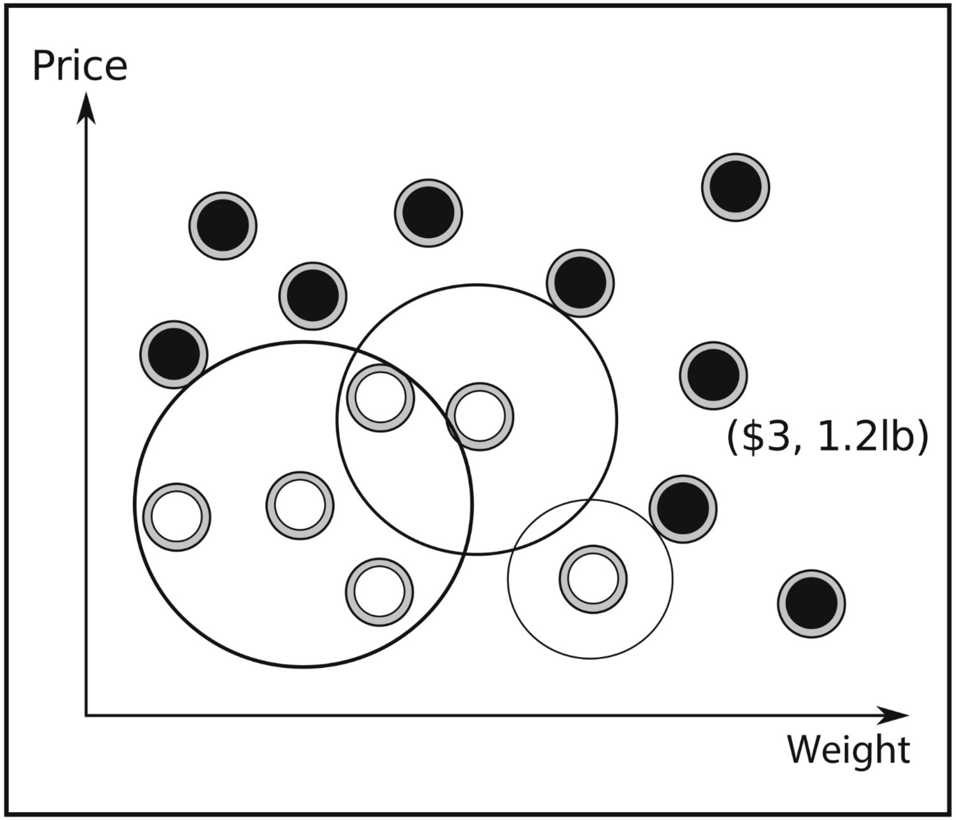
Input: a set of training examples, each described by the available attributes

Output: a boolean expression that is true for each positive example, and false for each negative example.

The evaluation function measures the given expression's error rate on the training set.



Numeric Attributes



Examples belonging to the same class tend to occupy a specific region.

Curves separating individual regions can be lines, circles, polynomials.

Search:

Identify the initial center with a randomly selected positive example, making the initial radius so small that the circle contains only this single example.

Two search operators:

– one increases the circle's radius

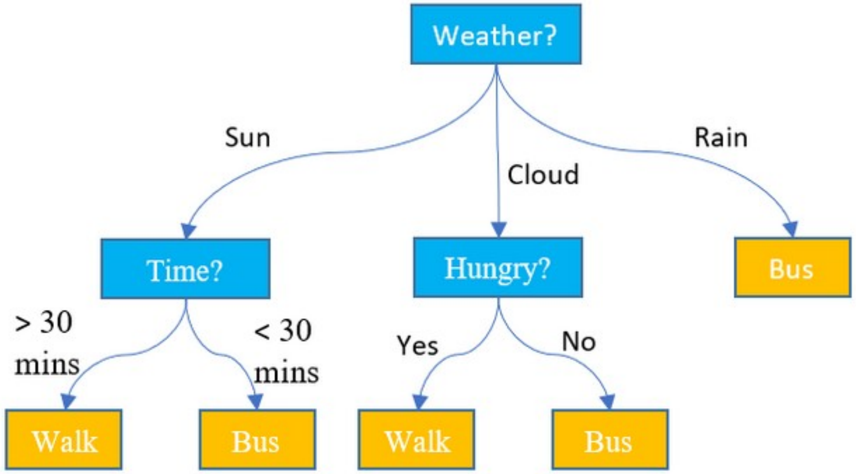
– the other shifts the center from one training example to another

Different categories of classifiers

Inductive/deductive

– deal with the creation and application of rules

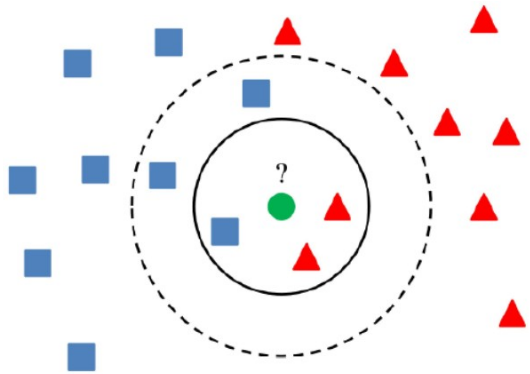
e.g. decision trees



Transductive

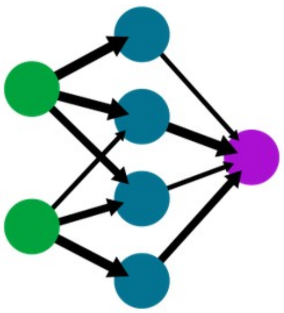
– based on the distances of the unknown data points to the known ones

e.g. *k* Nearest Neighbors



Heuristics-based

– use various heuristics for creating meta-features which are then used for the classification through some aggregation process

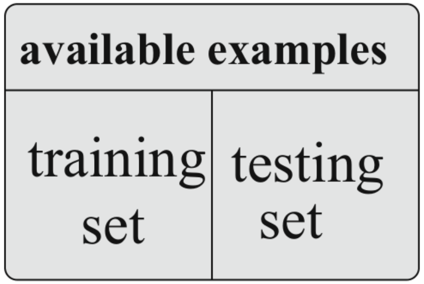


e.g. Artificial Neural Networks

Performance

The classifier's goal is to label future examples.

Divide the available pre-classified examples into two parts:



the classifier is induced

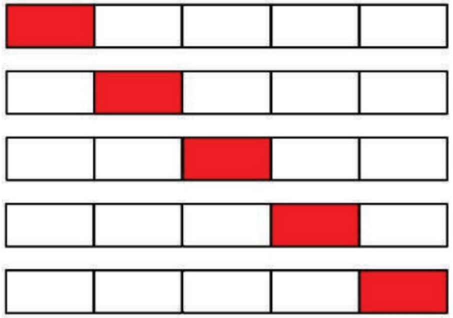
the classifier is evaluated

Drawback: A random choice of training examples may not be sufficiently representative of the underlying concept

A different training/testing set division gives rise to a different classifier.

One possible solution:

Repeat the random division into the training and testing sets several times.



https://datascience.stackexchange.com/questions/66930/how-to-keep-the-test-data-from-leaking-into-the-training-process-of-a-machine-le

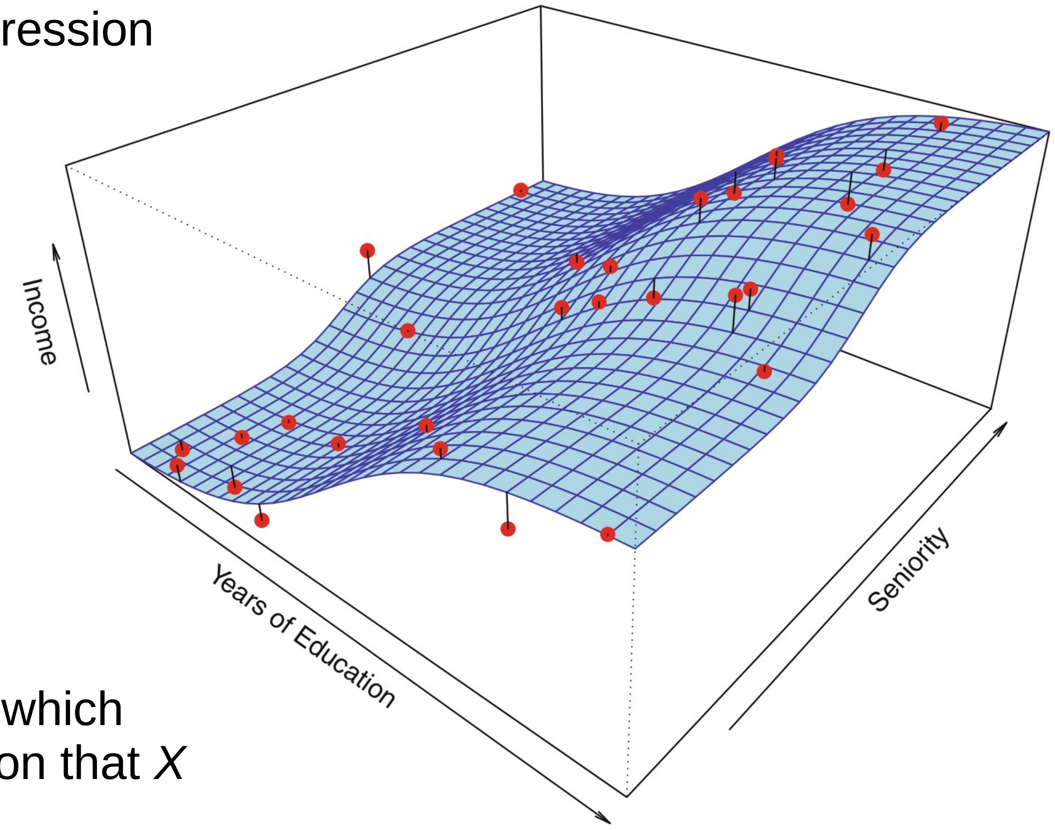
Regression

Assume that $y = f(X) + \epsilon$

$X = (X_1, X_2, \ldots, X_p)$

ϵ – random error, independent of X

$E[\epsilon] = 0$



$f(X)$ – some fixed but unknown function which represents the systematic information that X provides about y

Vertical lines: the error ϵ

The goal is to estimate *f* based on the observed points.

Prediction: $\hat{y} = \hat{f}(X)$

$X = (X_1, X_2, \ldots, X_p)$ – new data (not used for training)

$\hat{f}$ – estimate for f

Accuracy

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

Details:
https://stats.stackexchange.com/questions/110190/proof-derivation-of-residual-sum-of-squares-based-on-introduction-to-statistica

Reducible error

$\hat{f}$ is not a perfect estimate for f

Reducible error can be made smaller by using a better learning technique.

Irreducible error

– cannot be predicted using X

– may contain unmeasured variables that are useful in predicting *y*

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

Detais: https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff

Bias error ← erroneous assumptions in the learning algorithm

Relevant relations between features and target outputs are missed – **underfitting**

Variance error ← sensitivity to small fluctuations in the training set

The algorithm models random variations in the training data, rather than the intended outputs – **overfitting**

Difficulties with data

Irrelevant predictors

Each predictor increases the dimensionality of the problem.

Irrelevant predictors add to computational costs.

They can even mislead the learner.

Missing predictors

E.g. Johnny may be prejudiced against expensive pies but
the predictor price is missing.

Two examples (one positive, and another negative) can be identical
in terms of the available predictors but differ in the values of the vital
missing predictor.

Redundant predictors

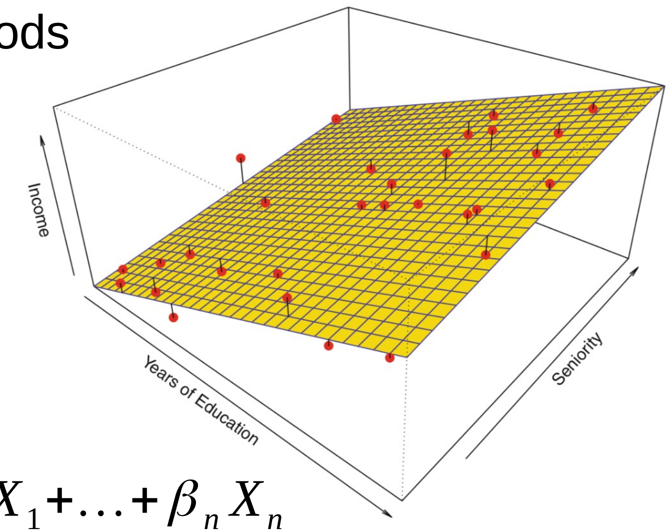
Their values can be obtained from other predictors

Missing predictor values

Predictor value noise

Class-label noise

Parametric and non-parametric methods



Parametric: A certain functional form of f is assumed

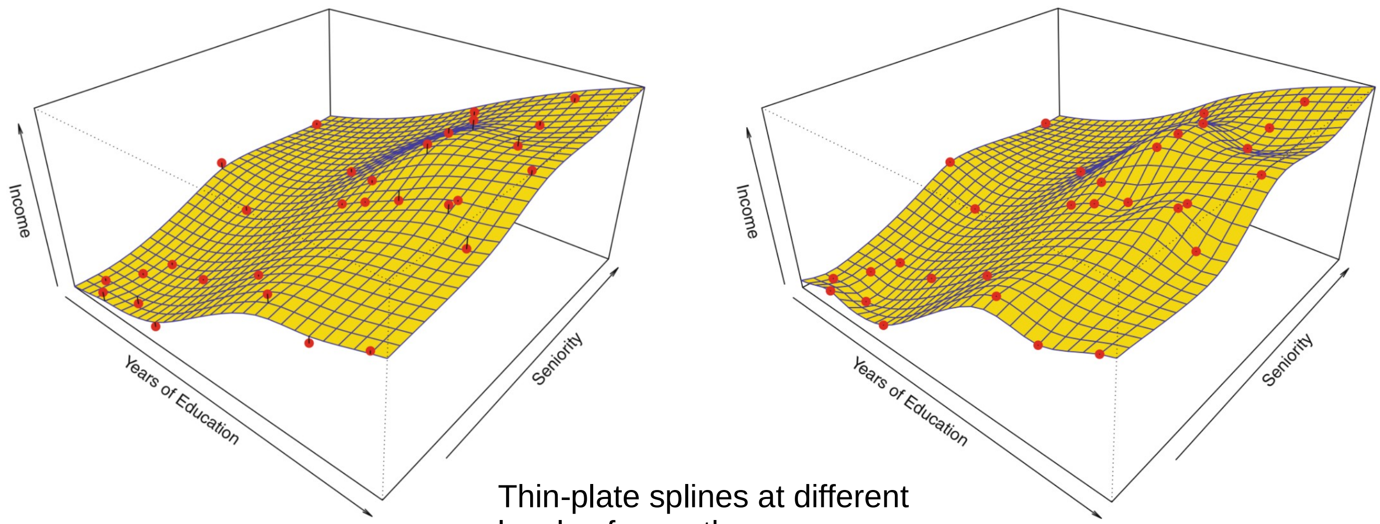
Disadvantage: The model may not match the true unknown form of f

E.g. $f(X)=\beta_0+\beta_1 X_1+\ldots+\beta_n X_n$

Advantage: One only needs to estimate the coefficients β_i

We can choose a more **flexible** model that can fit many different possible functional forms.

A more flexible model has more parameters → overfitting the data



Thin-plate splines at different levels of smoothness.

Non-parametric:

Seek an estimate of f that gets as close to the data as possible without being too rough or wiggly.

Lower smoothness → overfitting the data

Semi-parametric methods

$$f(X) = f_{known}\left(\phi_1(X_{k_1}, \ldots X_{n_1}), \ldots, \phi_p(X_{k_p}, \ldots X_{n_p})\right)$$

Assumed functional form, unknown parameters

Unknown functions learned from data

E.g. $f(X) = \dfrac{\beta_0 + \beta_1 \phi_1(X_1)}{\beta_2 \phi_2(X_2) + \beta_3 \phi_3(X_3)}$

E.g. from domain knowledge

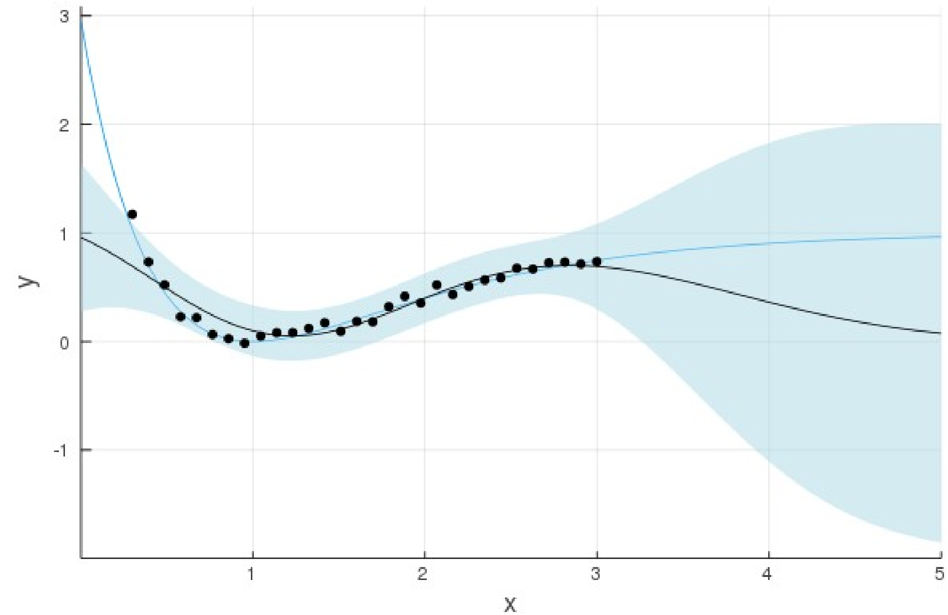
Additive models: $f(X) = \beta_0 + \beta_1 \phi_1(X_1) + \ldots + \beta_n \phi_n(X_n)$

An attempt to learn the Morse potential with a Gaussian process regression ...

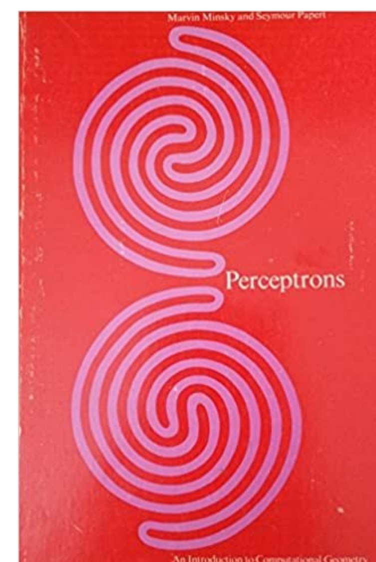
$$y=\left(1-e^{-(r-r_e)}\right)^2$$

julia 1.5.3

```
using GaussianProcesses, Random, Plots
gr(fmt=:png);

morse(x) = (1.0-exp(-(x-1.0)))^2;

# Training data
n_train = 30; #number of training points
x_train = LinRange(0.3, 3, n_train);
y_train = morse.(x_train) .+ 0.05*randn(n_train);

# Test data
n_test = 100; #number of test points
x_test = LinRange(0.001, 5, n_test);
y_test = morse.(x_test);

mZero = MeanZero();
kern = SE(0.0,0.0);
logObsNoise = -1.0;
gp = GP(x_train, y_train, mZero,kern,logObsNoise);

μ, σ² = predict_y(gp, x_test);

plot(x_test, y_test)
plot!(gp; xlabel="x", ylabel="y", legend=false, xlims=(x_test[1],x_test[end]))
```



1969



Consequences of learning without context

https://twitter.com/interesting_jpg

Lake et al. Building Machines That Learn and Think Like People.
https://arxiv.org/abs/1604.00289

A man is holding a woman in a hat.

Two people sitting on a bench talking to each other.

A man that is about to kiss.

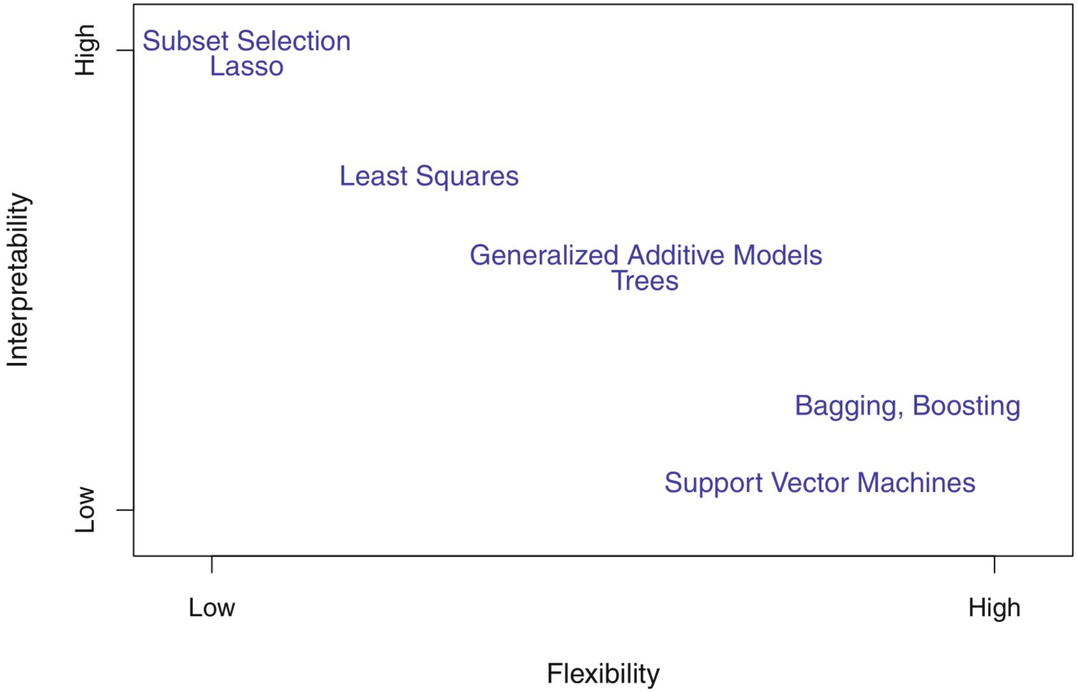
A car with a bunch of stuff on it.

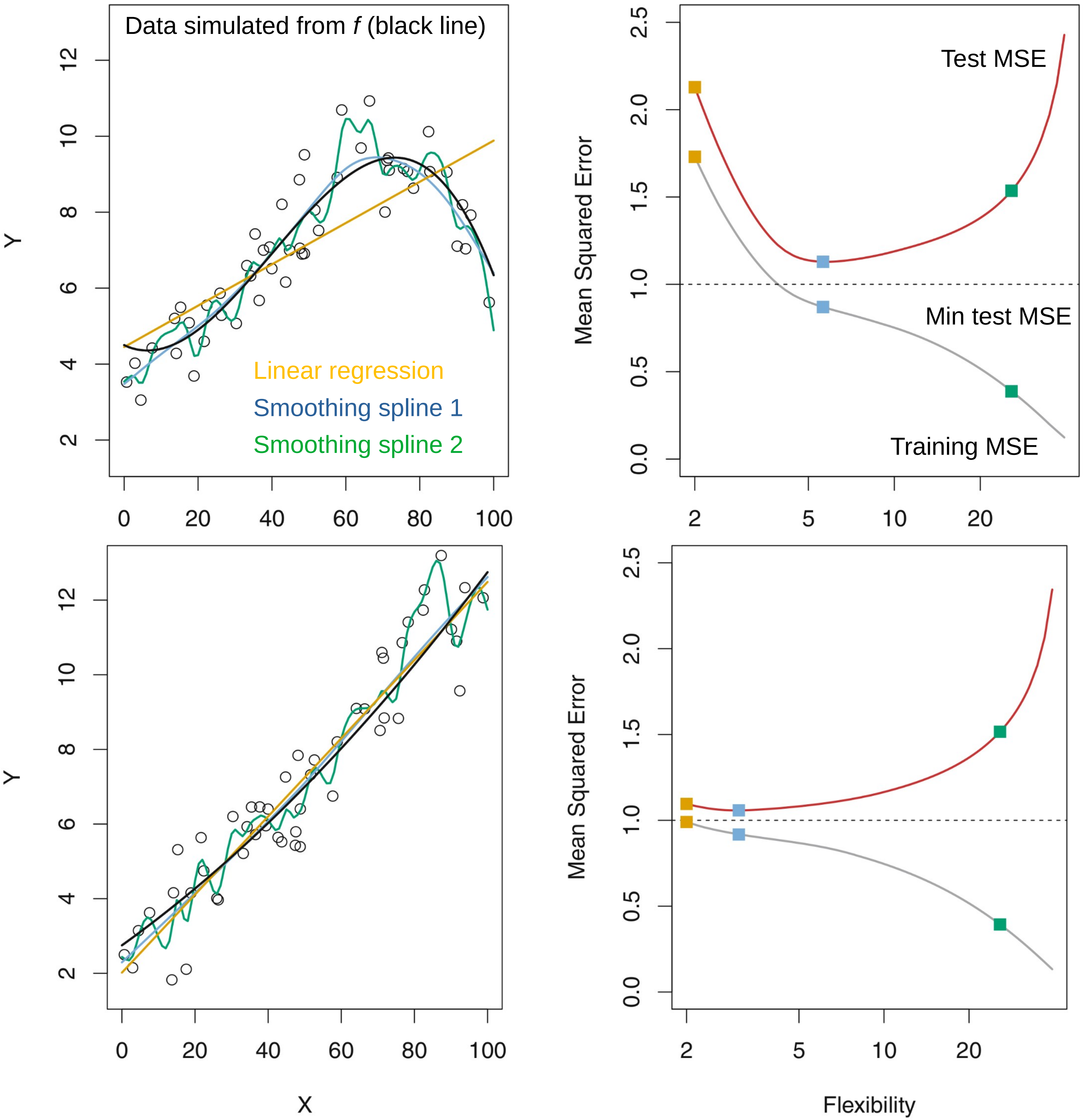
Inference

We want to understand the relationship between y and $X_1, X_2, \ldots, X_p$

Which predictors are associated with the response?

What is the relationship between
the response and each predictor?





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

Common cost functions:

https://stats.stackexchange.com/questions/154879/a-list-of-cost-functions-used-in-neural-networks-alongside-applications

https://stats.stackexchange.com/questions/488434/can-overfitting-and-underfitting-occur-simultaneously

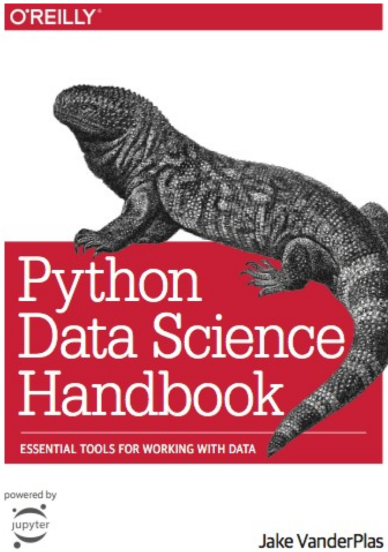
A simple f also captures some of the random patterns due to ϵ.

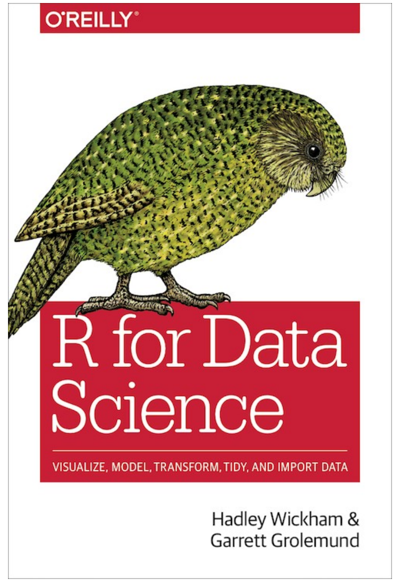
The no free lunch theorem (Wolpert 1996):

There is no model that works well for all data.

Hands-On Start

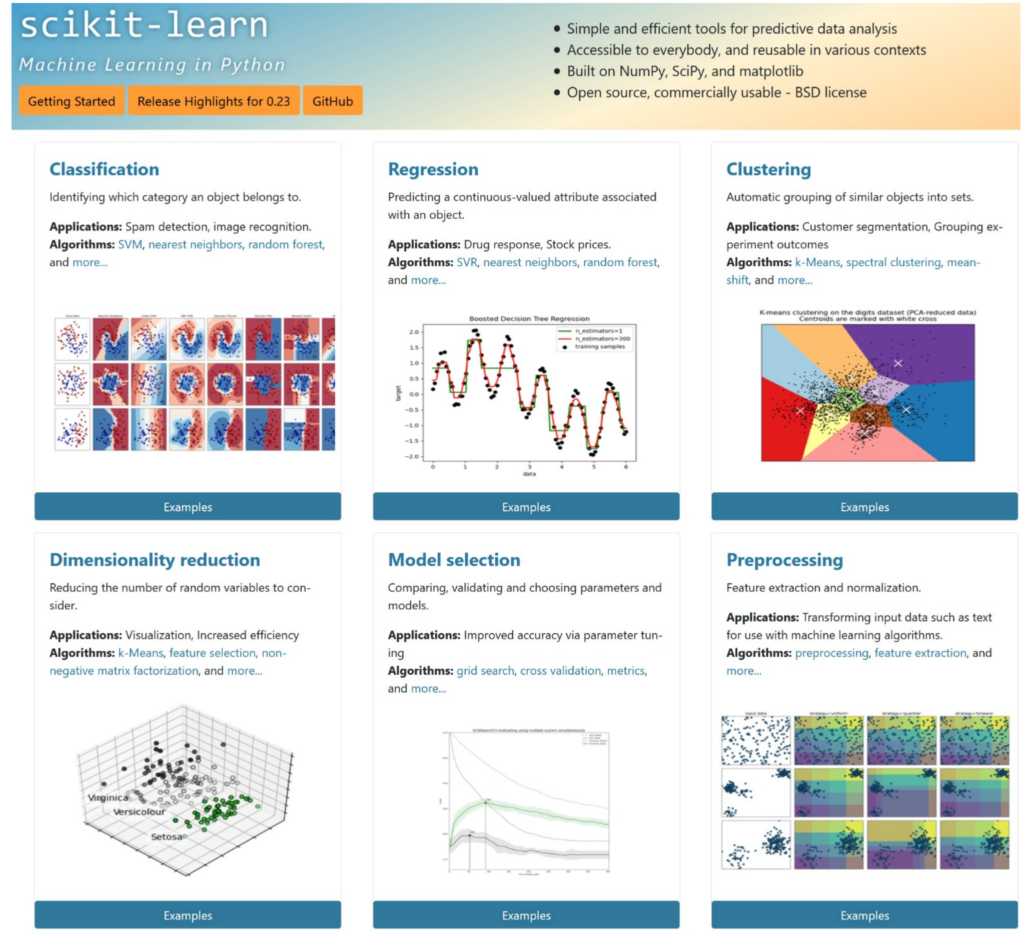
Scikit-learn https://scikit-learn.org/





https://r4ds.had.co.nz/

https://jakevdp.github.io/PythonDataScienceHandbook/



Kaggle https://www.kaggle.com/

Keras

PyTorch

TensorFlow

H2O.ai

mxnet

mlpack

theano

Lasagne https://lasagne.readthedocs.io/en/latest/

Blocks https://blocks.readthedocs.io/en/latest/index.html

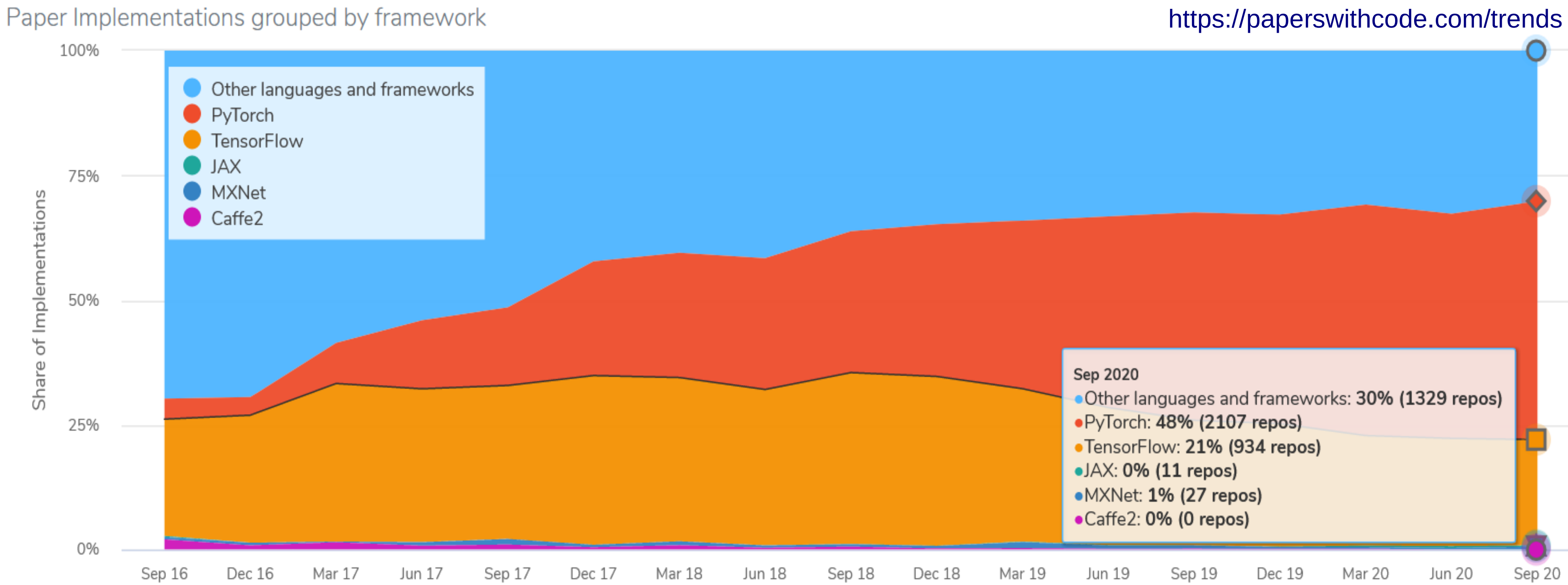
Fortran https://github.com/modern-fortran/neural-fortran

JAX https://github.com/google/jax

Haskell http://www.datahaskell.org/

Rust http://www.arewelearningyet.com/

Machine learning libraries in different languages: https://github.com/josephmisiti/awesome-machine-learning



Flux.jl Clustering.jl Regression.jl

ForneyLab.jl LIBSVM.jl MLBase.jl

GaussianProcesses.jl MLDatasets.jl

DecisionTree.jl RDatasets.jl

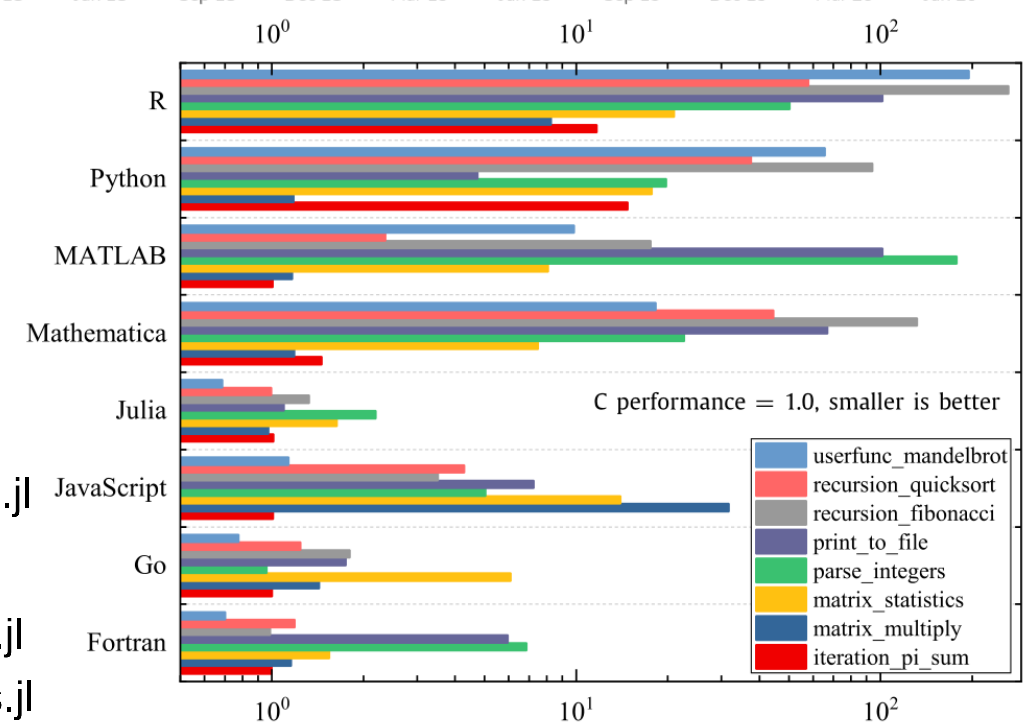
MLDataUtils.jl XGBoost.jl

Knet.jl Diffiqflux.jl KernelFunctions.jl

CombineML.jl BayesNets.jl Lasso.jl

Reinforce.jl NearestNeighbors.jl GLM.jl

GaussianMixtures.jl MultivariateStats.jl



K. Gao, G. Mei, F. Piccialli et al.
Computer Science Review 37 (2020) 100254

Machine Learning for Everyone https://vas3k.com/blog/machine_learning/

Stack Abuse https://stackabuse.com/

Deep Learning Prerequisites: The Numpy Stack in Python https://www.udemy.com/course/numpy-python/

Statistics with Julia https://statisticswithjulia.org/

Julia language in machine learning https://doi.org/10.1016/j.cosrev.2020.100254

R-bloggers https://www.r-bloggers.com/

Very ML https://infomate.club/ml/

Python Data Science Handbook https://jakevdp.github.io/PythonDataScienceHandbook/

A summary of tools for data science for Python http://www.davekuhlman.org/py-datasci-survey.html

Real Python Tutorials https://realpython.com/

Practical Business Python https://pbpython.com/

Python Programming Guides and Tutorials https://www.pythoncentral.io/

Machine learning mastery https://machinelearningmastery.com

Papers with code https://paperswithcode.com/sota

Python implementations of some of the ML models https://github.com/eriklindernoren/ML-From-Scratch

Deep learning roadmap https://github.com/instillai/deep-learning-roadmap

Open Machine Learning Course https://mlcourse.ai/

Machine Learning Crash Course https://developers.google.com/machine-learning/crash-course/

References

Miroslav Kubat. An Introduction to Machine Learning. 2017. 2nd Edition.
DOI 10.1007/978-3-319-63913-0

Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. An Introduction to Statistical Learning with Applications in R. 2017. DOI 10.1007/978-1-4614-7138-7

Kevin P. Murphy. Machine learning: a probabilistic perspective. 2012.

Shai Shalev-Shwartz, Shai Ben-David. Understanding Machine Learning: From Theory to Algorithms. 2014.
https://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/index.html

Richard J. Roiger. Just enough R! : an interactive approach to machine learning and analytics. 2020.

Hefin I. Rhys. Machine Learning with R, the tidyverse, and mlr. 2020.

Brad Boehmke, Brandon Greenwell. Hands-On Machine Learning with R. 2020.

Jeremy Watt, Reza Borhani, Aggelos K. Katsaggelos. Machine Learning Refined. Foundations, Algorithms, and Applications. 2nd Edition. 2020.

Machine Learning Meets Quantum Physics https://link.springer.com/book/10.1007%2F978-3-030-40245-7

Matthias Rupp, Machine learning for quantum mechanics in a nutshell https://doi.org/10.1002/qua.24954