Machine Learning with Python

Sebastian Raschka, Ph.D.
MSU Data Science workshop
East Lansing, Michigan State University • Feb 21, 2018

Today's focus:

scikit learn

And if we have time, a quick overview ...

TensorFlow

PYTORCH

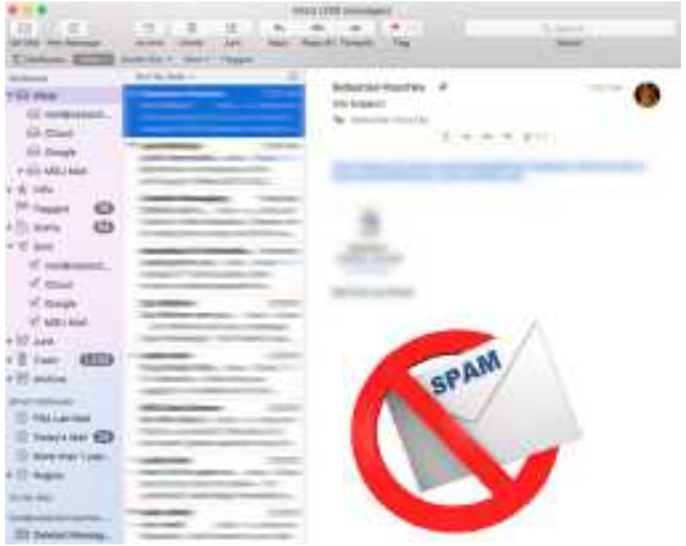
Tutorial Material on GitHub:

https://github.com/rasbt/msu-datascience-ml-tutorial-2018

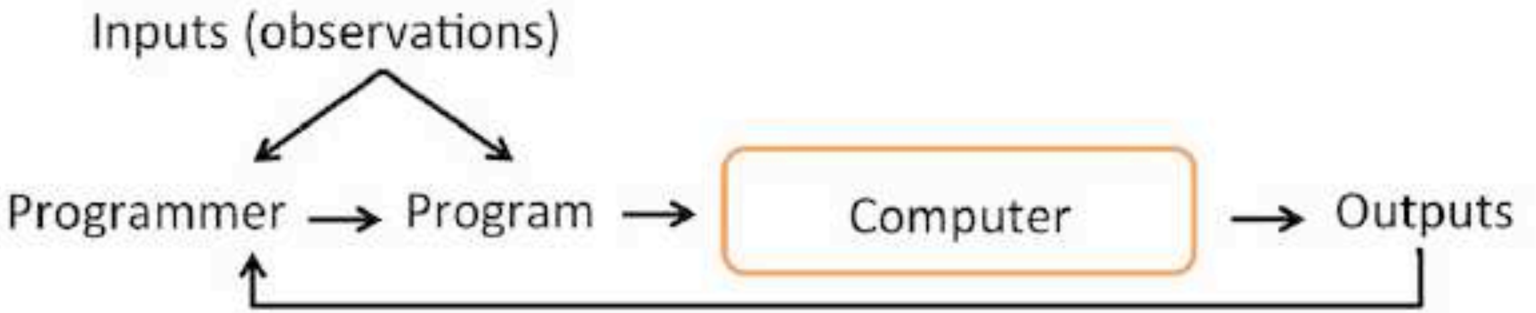
Contact:

- E-mail: mail@sebastianraschka.com
- Website: http://sebastianraschka.com
- Twitter: @rasbt
- GitHub: rasbt

Machine learning is used & useful (almost) anywhere



The Traditional Programming Paradigm



Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed
– Arthur Samuel (1959)

Machine Learning

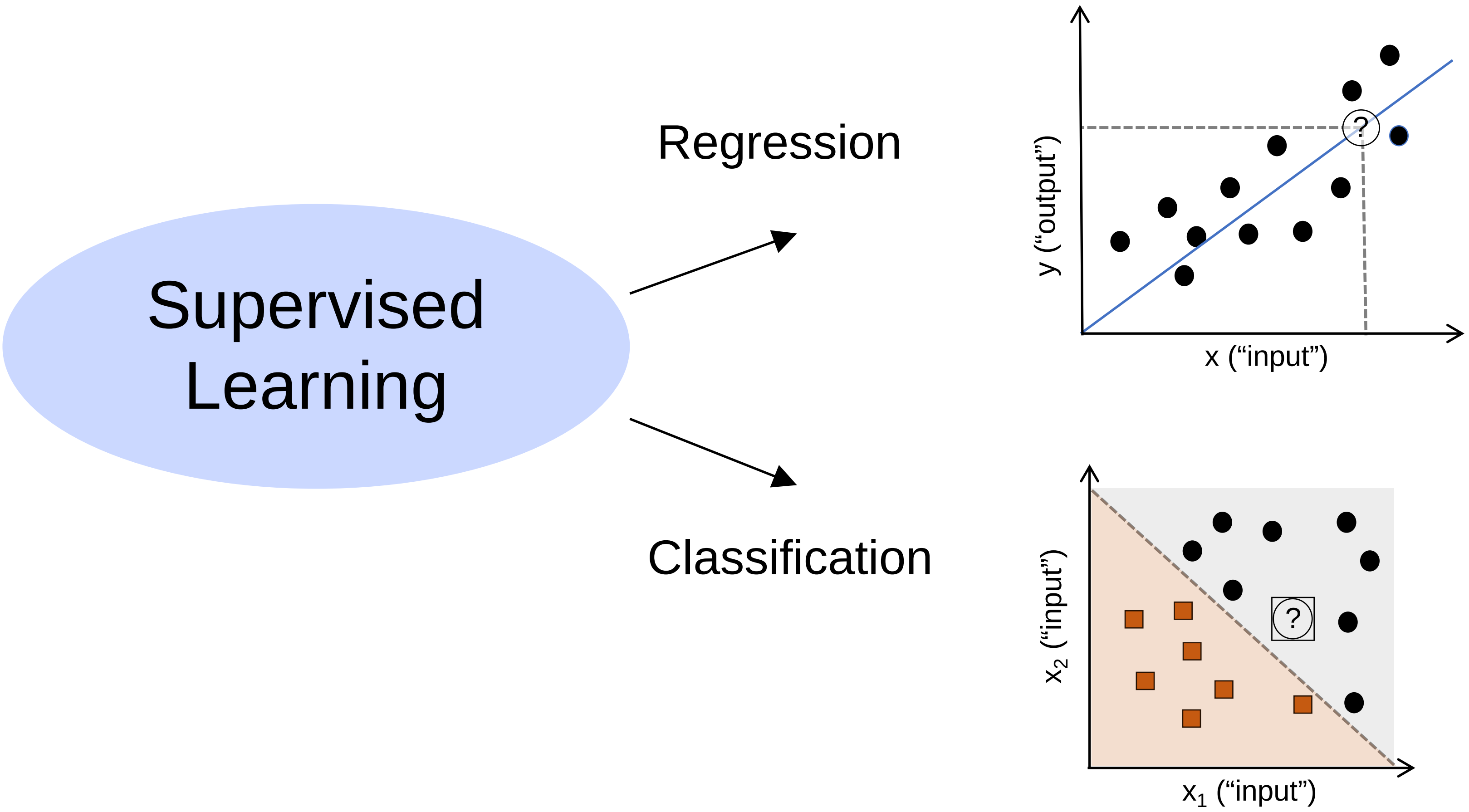
3 Types of Learning

Supervised

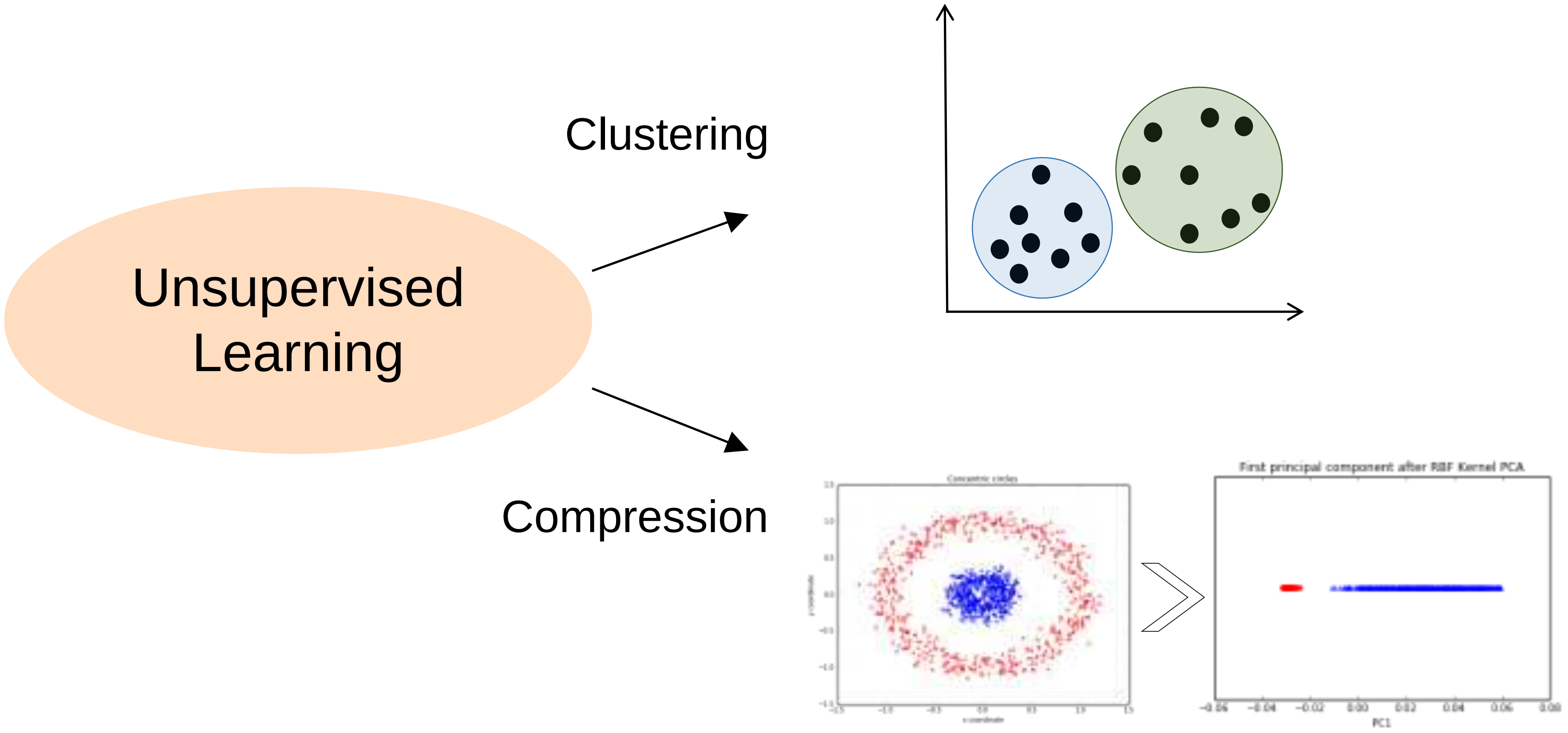
Unsupervised

Reinforcement

Working with Labeled Data



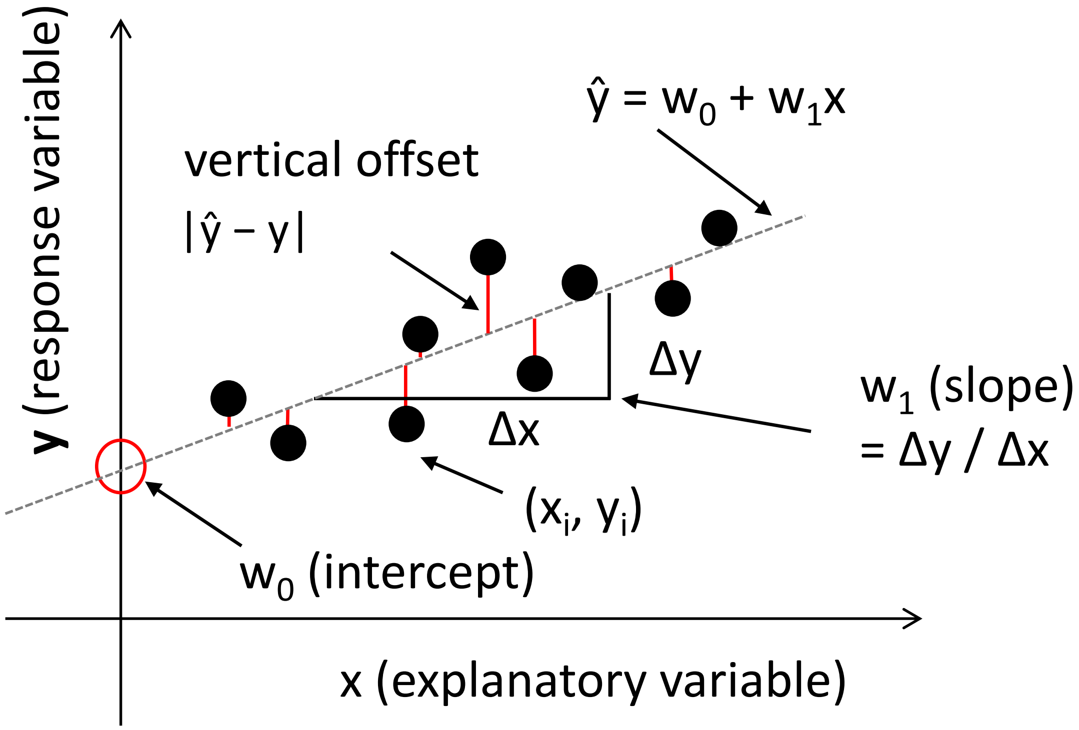
Working with Unlabeled Data



Topics

1. Introduction to Machine Learning
2. **Linear Regression**
3. Introduction to Classification
4. Feature Preprocessing & scikit-learn Pipelines
5. Dimensionality Reduction: Feature Selection & Extraction
6. Model Evaluation & Hyperparameter Tuning

Simple Linear Regression



Data Representation

Columns: features (explanatory variables, independent variables, covariates, predictors, variables, inputs, attributes)

X=

x_0	x_1	…	x_m
$x_{0,0}$	$x_{0,1}$		
$x_{1,0}$	$x_{1,1}$		
$x_{2,0}$	$x_{2,1}$		
$x_{3,0}$	$x_{3,1}$		
.			
.			
.			
$x_{n,0}$	$x_{n,1}$	…	$x_{n,m}$

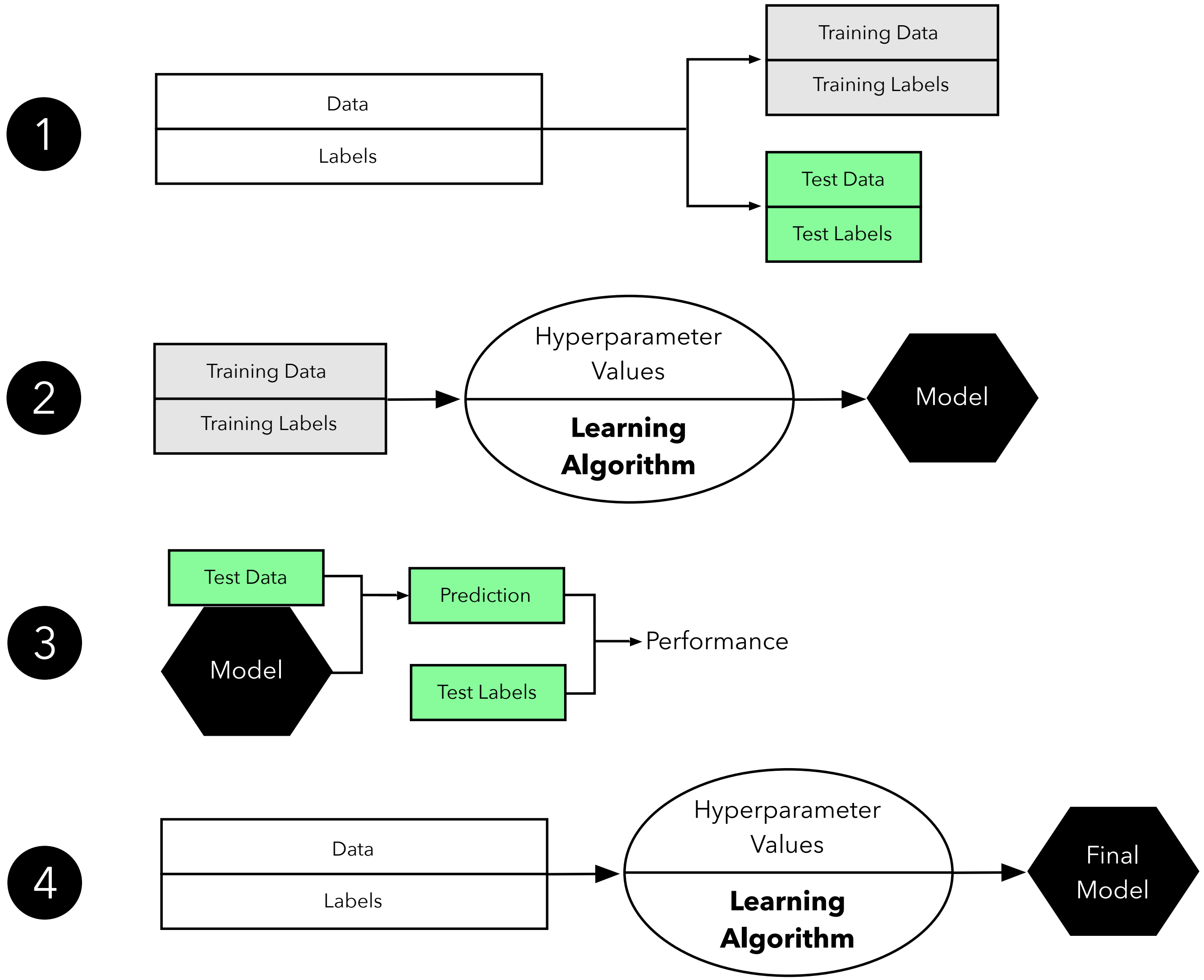
Rows: training examples (observations, records, instances, samples)

y=

y_0
y_1
y_2
y_3
.
.
.
y_n

Targets (target variable,response variable, dependent variable, labels, ground truth)

“Basic” Supervised Learning Workflow



⇨ Jupyter Notebook

Topics

1. Introduction to Machine Learning
2. Linear Regression
3. **Introduction to Classification**
4. Feature Preprocessing & scikit-learn Pipelines
5. Dimensionality Reduction: Feature Selection & Extraction
6. Model Evaluation & Hyperparameter Tuning

Scikit-learn API

```
class SupervisedEstimator(...):
    def __init__(self, hyperparam, ...):
        ...
    def fit(self, X, y):
        ...
        return self
    def predict(self, X):
        ...
        return y_pred
    def score(self, X, y):
        ...
        return score
    ...
```

Iris Dataset

Iris-Setosa

Iris-Versicolor

Iris-Virginica

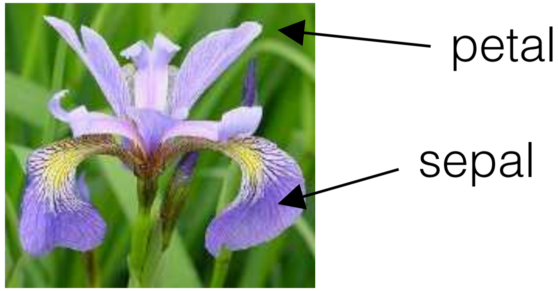
Iris Dataset

features (columns)

X=

	sepal length [cm]	sepal width [cm]	petal lengt h [cm]	petal width [cm]
1	5.1	3.5	1.4	0.2
2	4.9	3.0	1.4	0.2
50	6.4	3.5	4.5	1.2
	.			
	.			
	.			
150	5.9	3.0	5.0	1.8

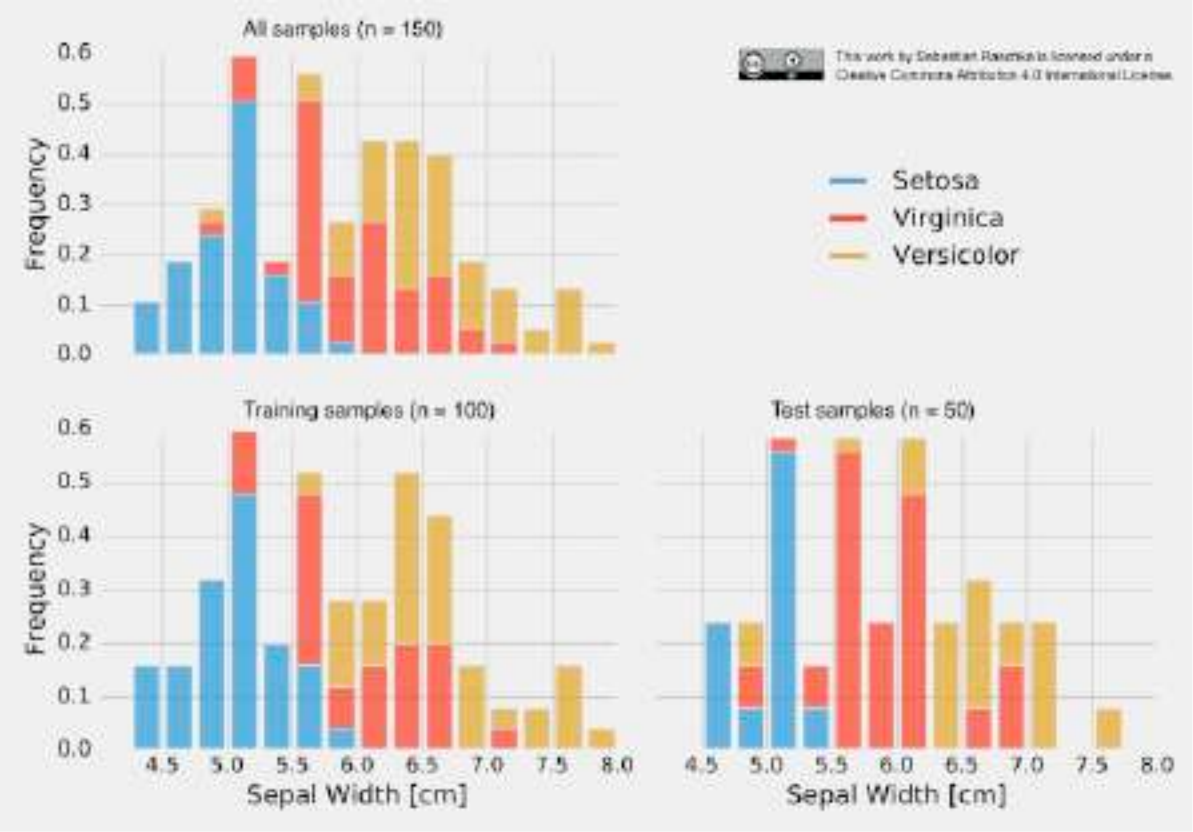
samples (rows)



y=

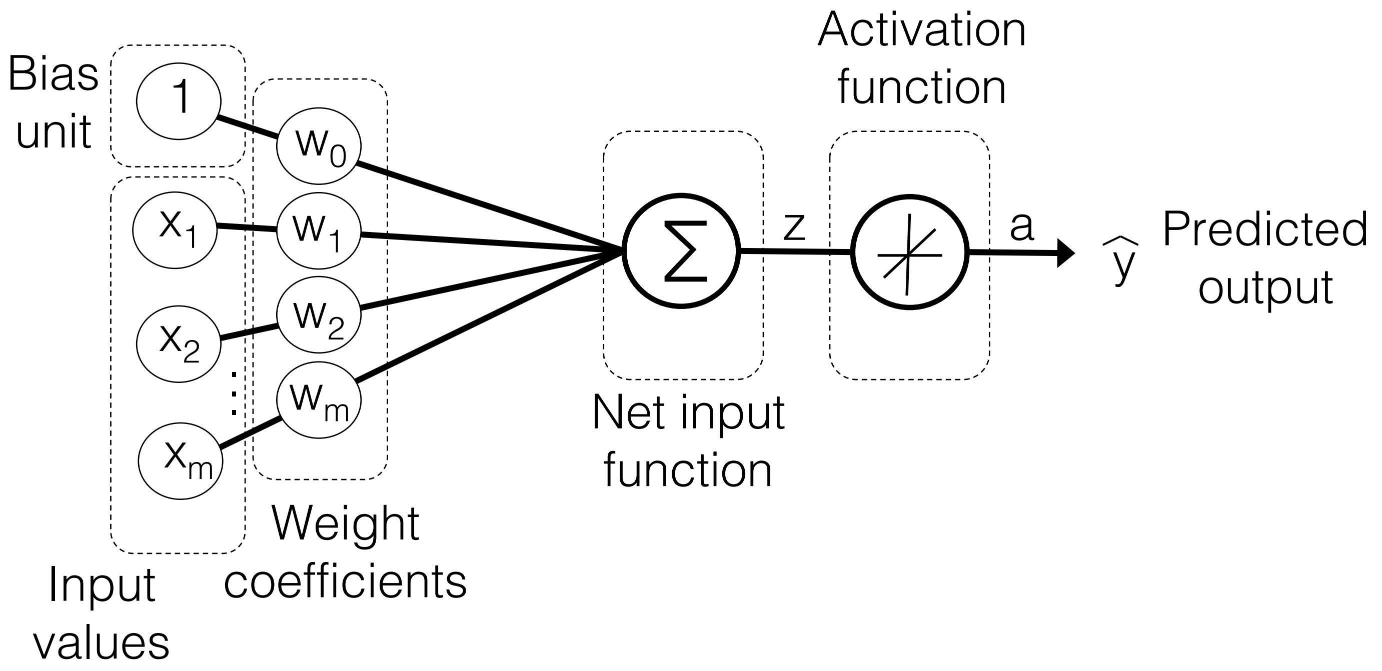
setosa
setosa
versicolor
.
.
.
virginica

Note about Non-Stratified Splits

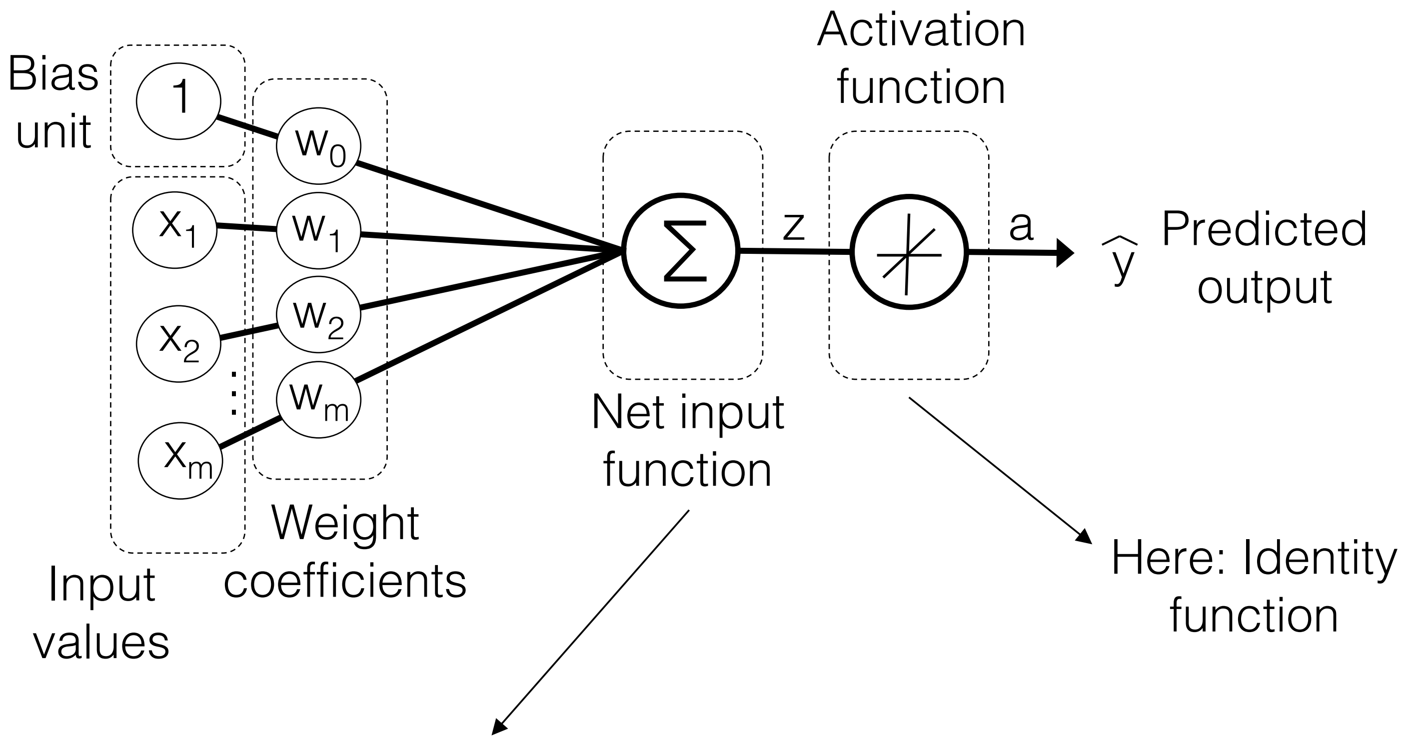


- training set → 38 x Setosa, 28 x Versicolor, 34 x Virginica
- test set → 12 x Setosa, 22 x Versicolor, 16 x Virginica

Linear Regression Recap

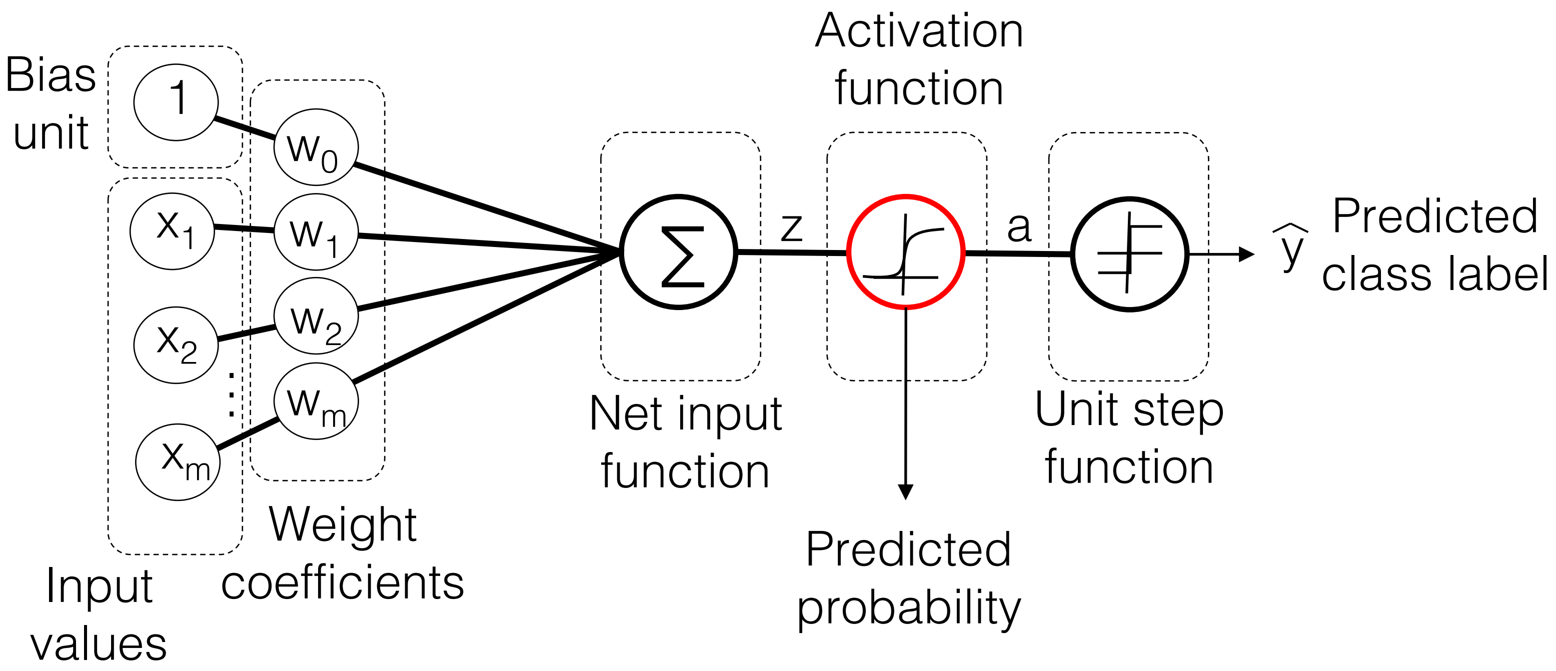


Linear Regression Recap

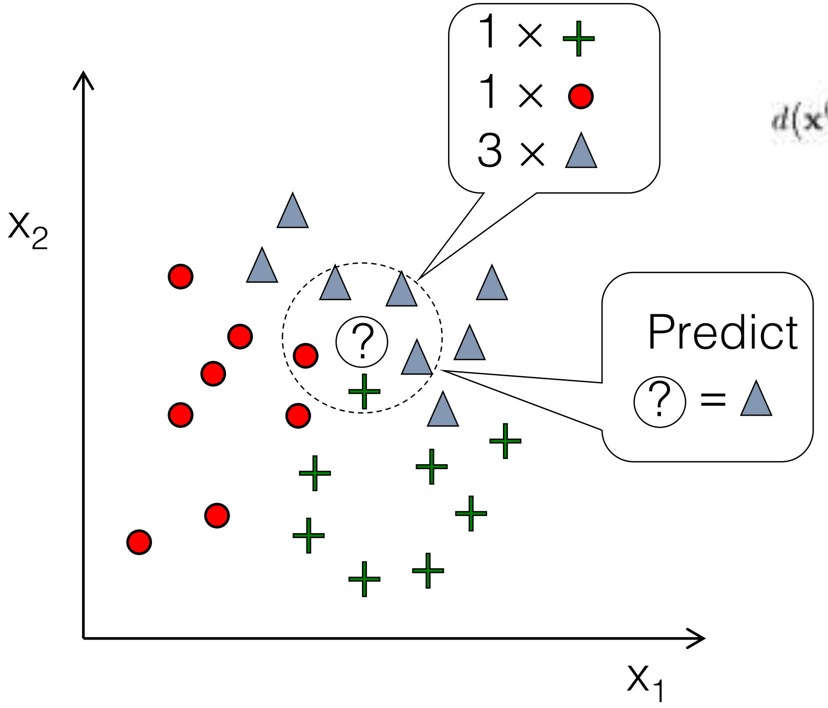


$$z = w_0 x_0 + w_1 x_1 + \cdots + w_m x_m = \mathbf{w}^\mathbf{T}\mathbf{x}$$

Logistic Regression, a Generalized Linear Model (a Classifier)



A “Lazy Learner:” K-Nearest Neighbors Classifier



$$d(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \sqrt[p]{\sum_k |x_k^{(i)} - x_k^{(j)}|^p}$$

⇨ Jupyter Notebook

There are many, many more classification and regression algorithms ...

http://scikit-learn.org/stable/supervised_learning.html

Topics

1. Introduction to Machine Learning
2. Linear Regression
3. Introduction to Classification
4. **Feature Preprocessing & scikit-learn Pipelines**
5. Dimensionality Reduction: Feature Selection & Extraction
6. Model Evaluation & Hyperparameter Tuning

Categorical Variables

color	size	price	class label
red	M	$10.49	0
blue	XL	$15.00	1
green	L	$12.99	1

Encoding Categorical Variables (Ordinal vs Nominal)

color	size	price	class label
red	M	$10.49	0
blue	XL	$15.00	1
green	L	$12.99	1

red	blue	green
1	0	0
0	1	0
0	0	1

size
0
2
1

Feature Normalization

Min-max scaling

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Z-score standardization

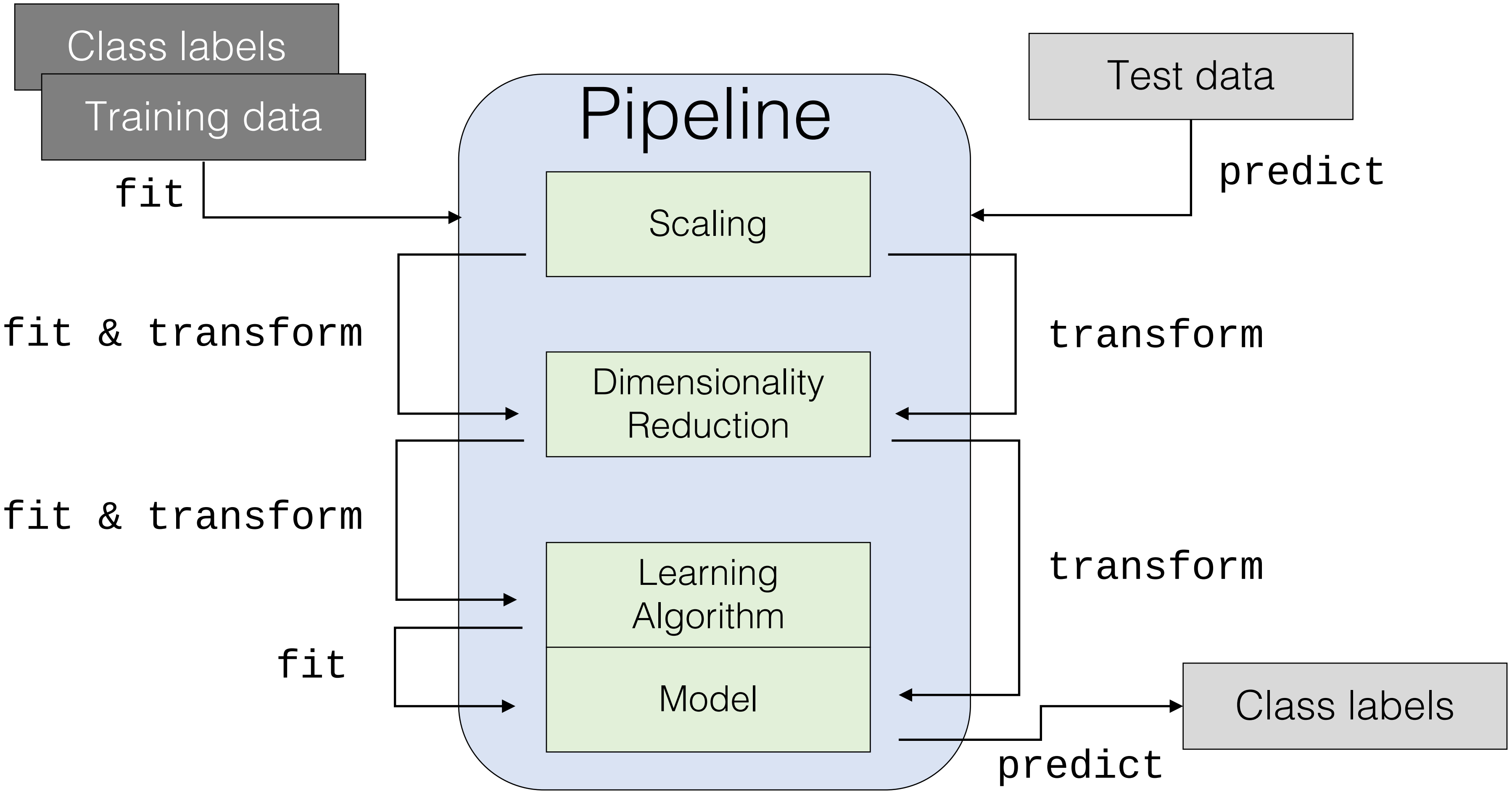
$$z = \frac{x - \mu}{\sigma}$$

feature	minmax	z-score
1.0	0.0	-1.46385
2.0	0.2	-0.87831
3.0	0.4	-0.29277
4.0	0.6	0.29277
5.0	0.8	0.87831
6.0	1.0	1.46385

Scikit-learn API

```
class UnsupervisedEstimator(...):
    def __init__(self, ...):
        ...
    def fit(self, X):
        ...
        return self
    def transform(self, X):
        ...
        return X_transf
  def predict(self, X):
        ...
        return pred
```

Scikit-learn Pipelines

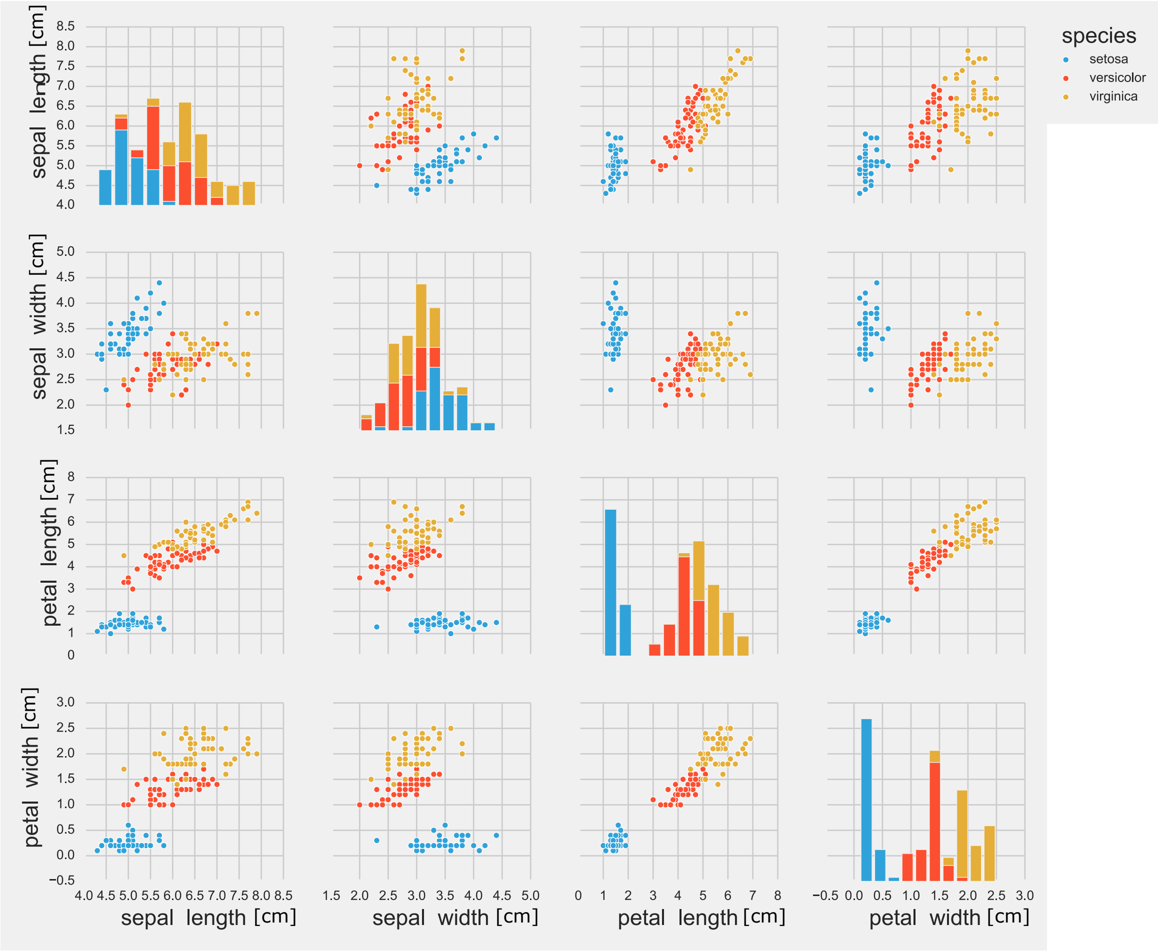


⇨ Jupyter Notebook

Topics

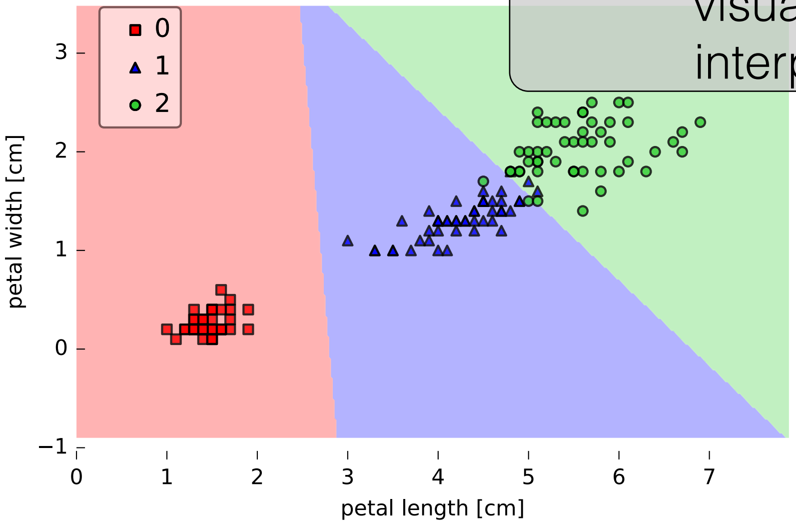
1. Introduction to Machine Learning
2. Linear Regression
3. Introduction to Classification
4. Feature Preprocessing & scikit-learn Pipelines
5. **Dimensionality Reduction: Feature Selection & Extraction**
6. Model Evaluation & Hyperparameter Tuning

Dimensionality Reduction – why?



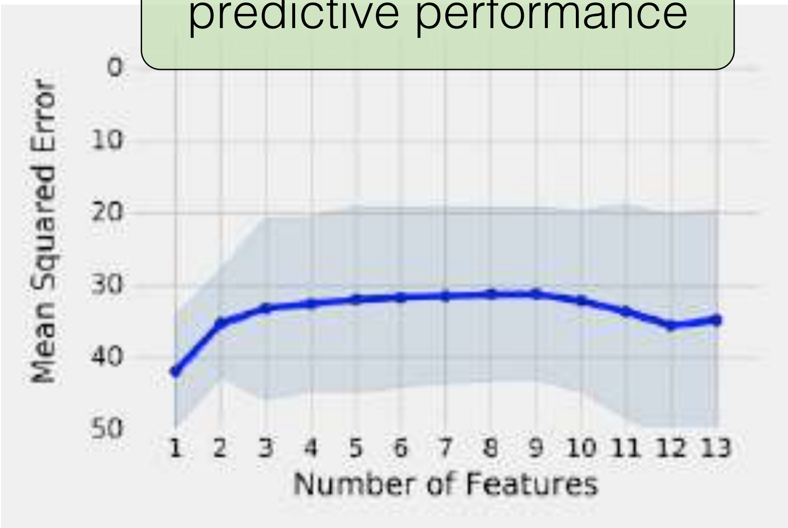
Dimensionality Reduction – why?

visualization &
interpretability



storage & speed

predictive performance



Recursive Feature Elimination

available features: [f1 f2 f3 f4]

[w1 w2 w3 w4]

fit model, remove lowest weight, repeat

[w1 w2 w4]

fit model, remove lowest weight, repeat

[w1 w4]

fit model, remove lowest weight, repeat

[w4]

Sequential Feature Selection

available features: [f1 f2 f3 f4]

[f1] [f2] [f3] [f4]

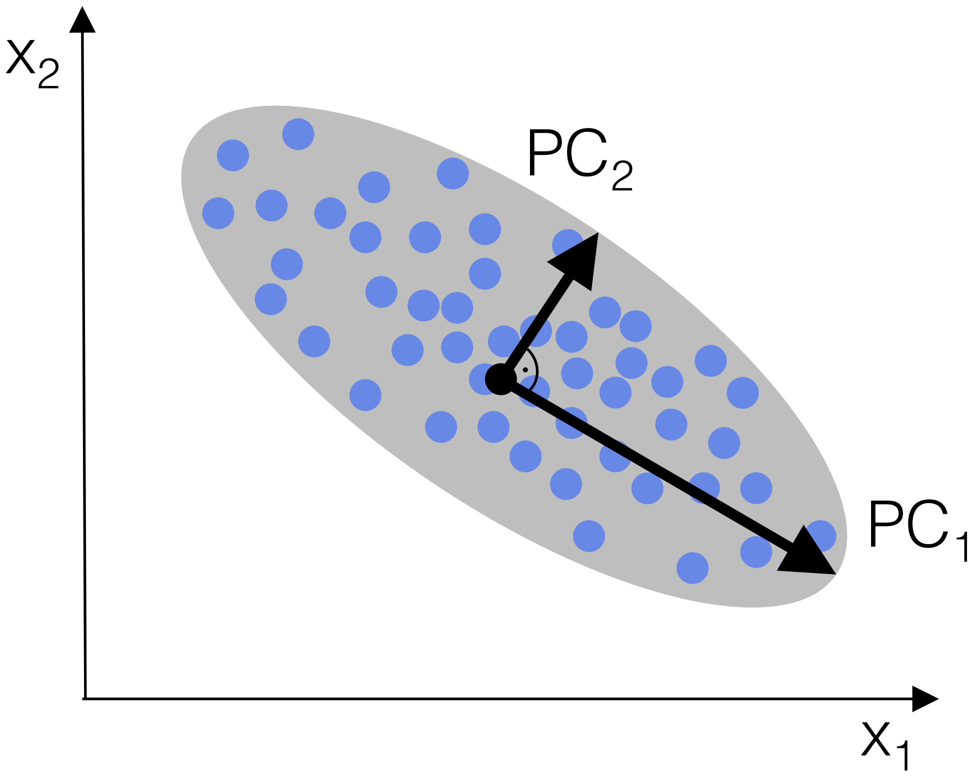
fit model, pick best, repeat

[f1 f3] [f1 f2] [f1 f4]

fit model, pick best, repeat

[f1 f3 f4] [f1 f3 f2]

Principal Component Analysis

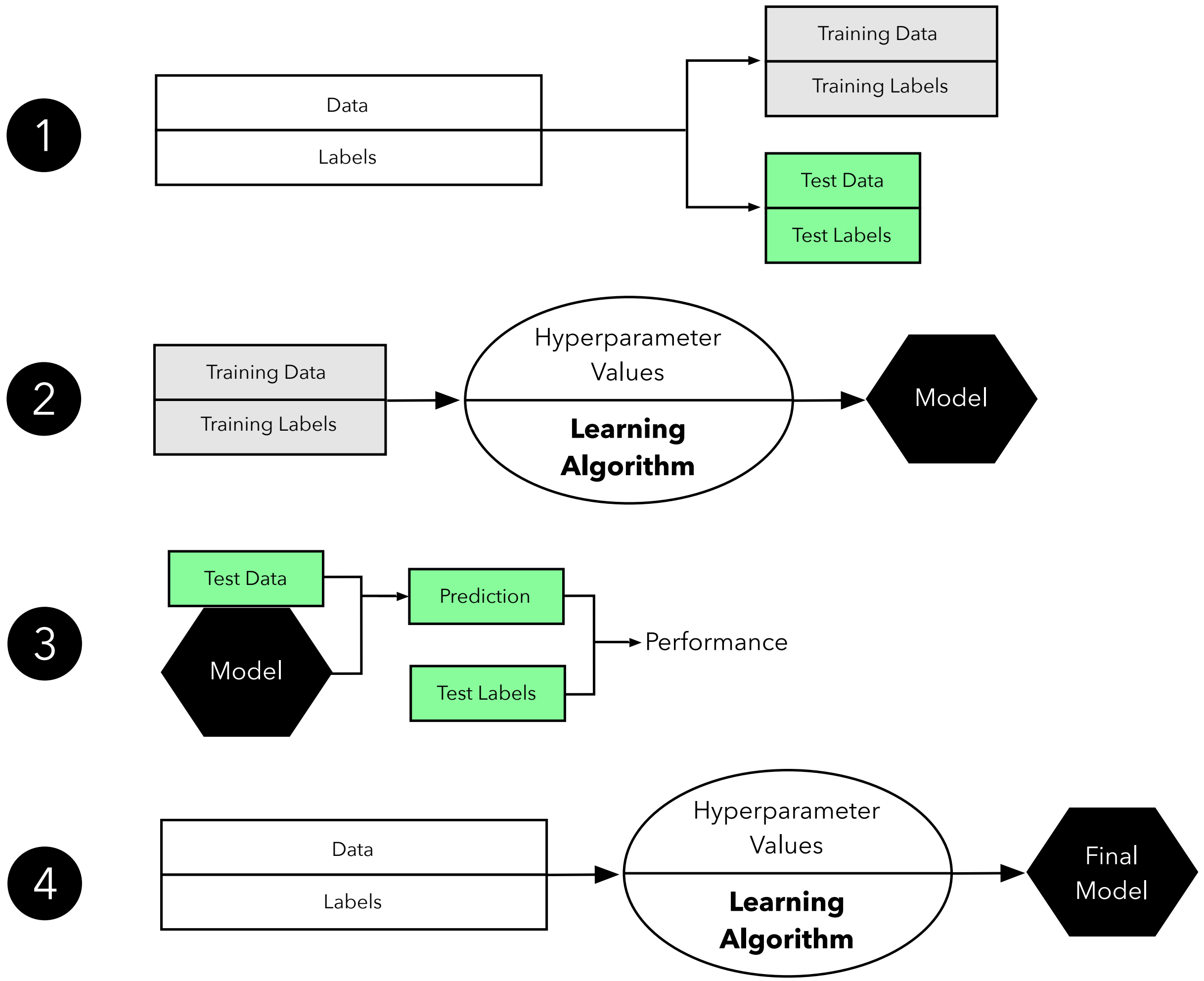


Jupyter Notebook

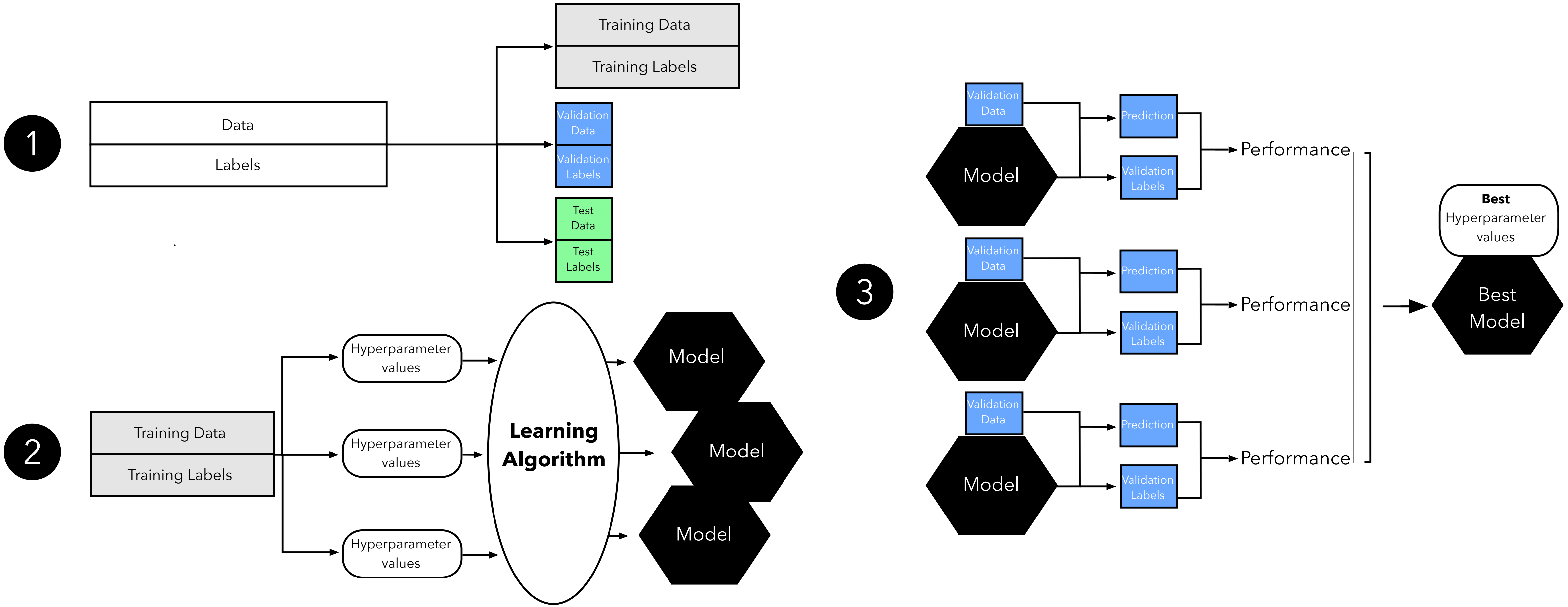
Topics

1. Introduction to Machine Learning
2. Linear Regression
3. Introduction to Classification
4. Feature Preprocessing & scikit-learn Pipelines
5. Dimensionality Reduction: Feature Selection & Extraction
6. **Model Evaluation & Hyperparameter Tuning**

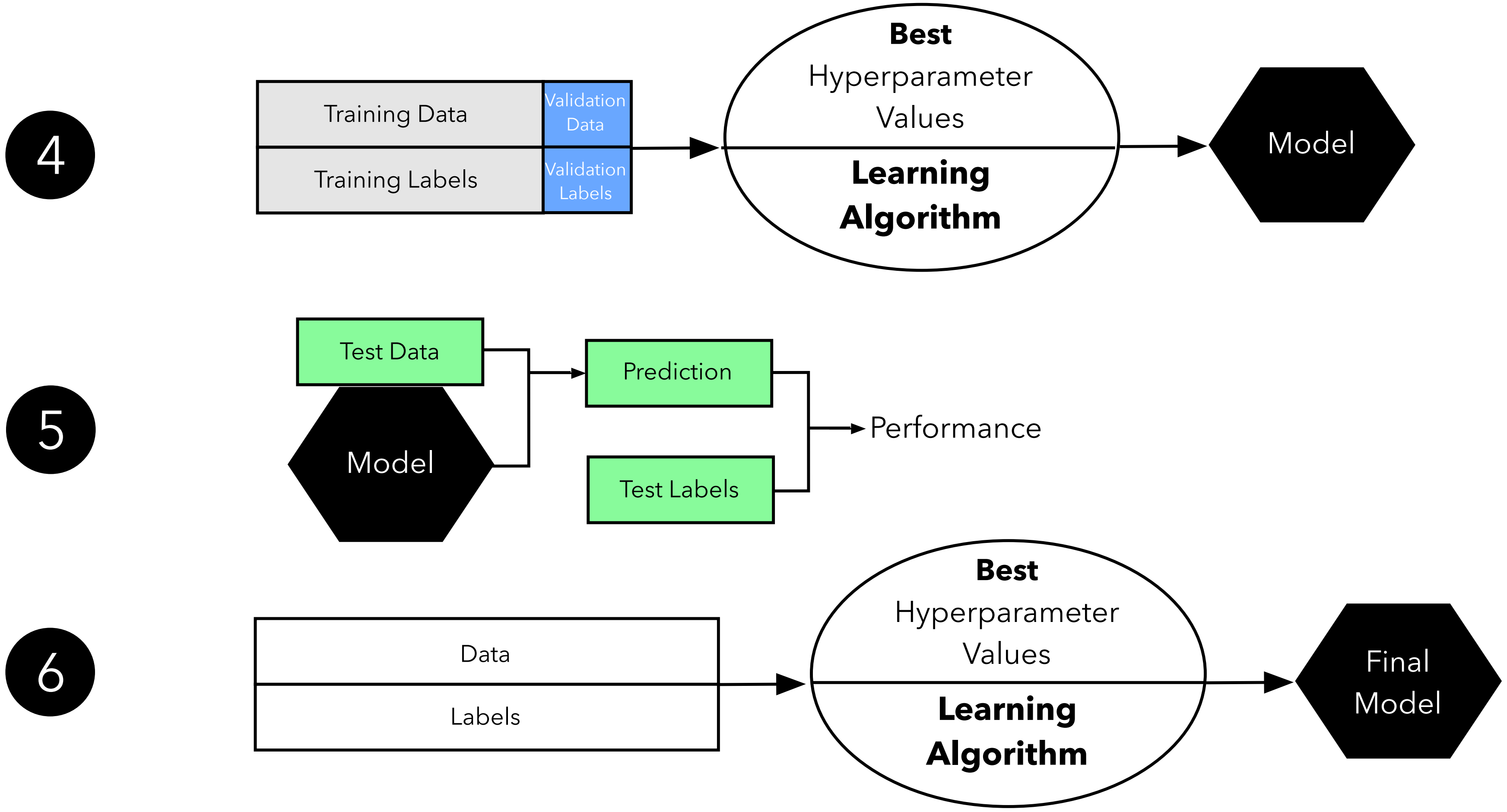
“Basic” Supervised Learning Workflow



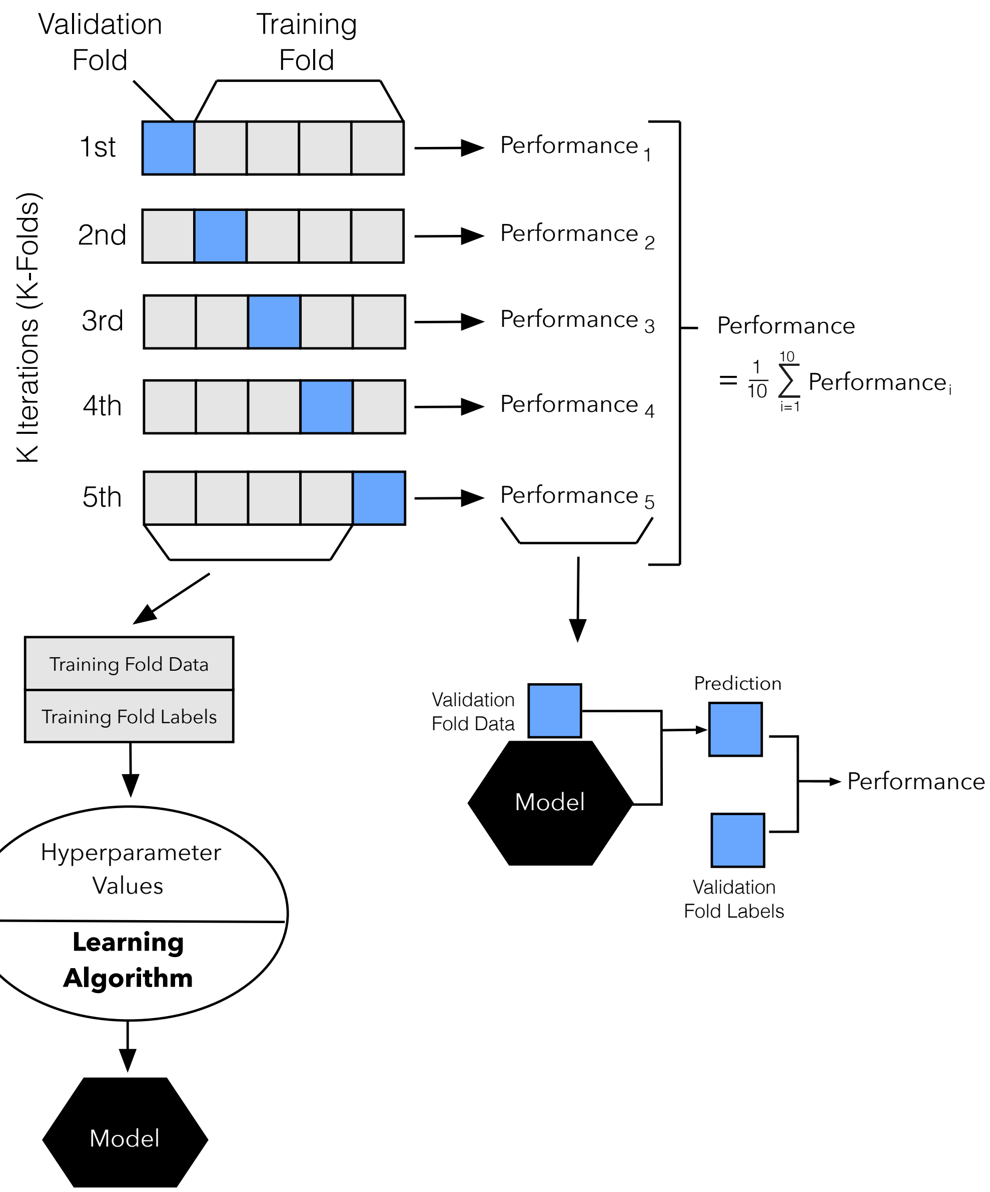
Holdout Method and Hyperparameter Tuning 1-3



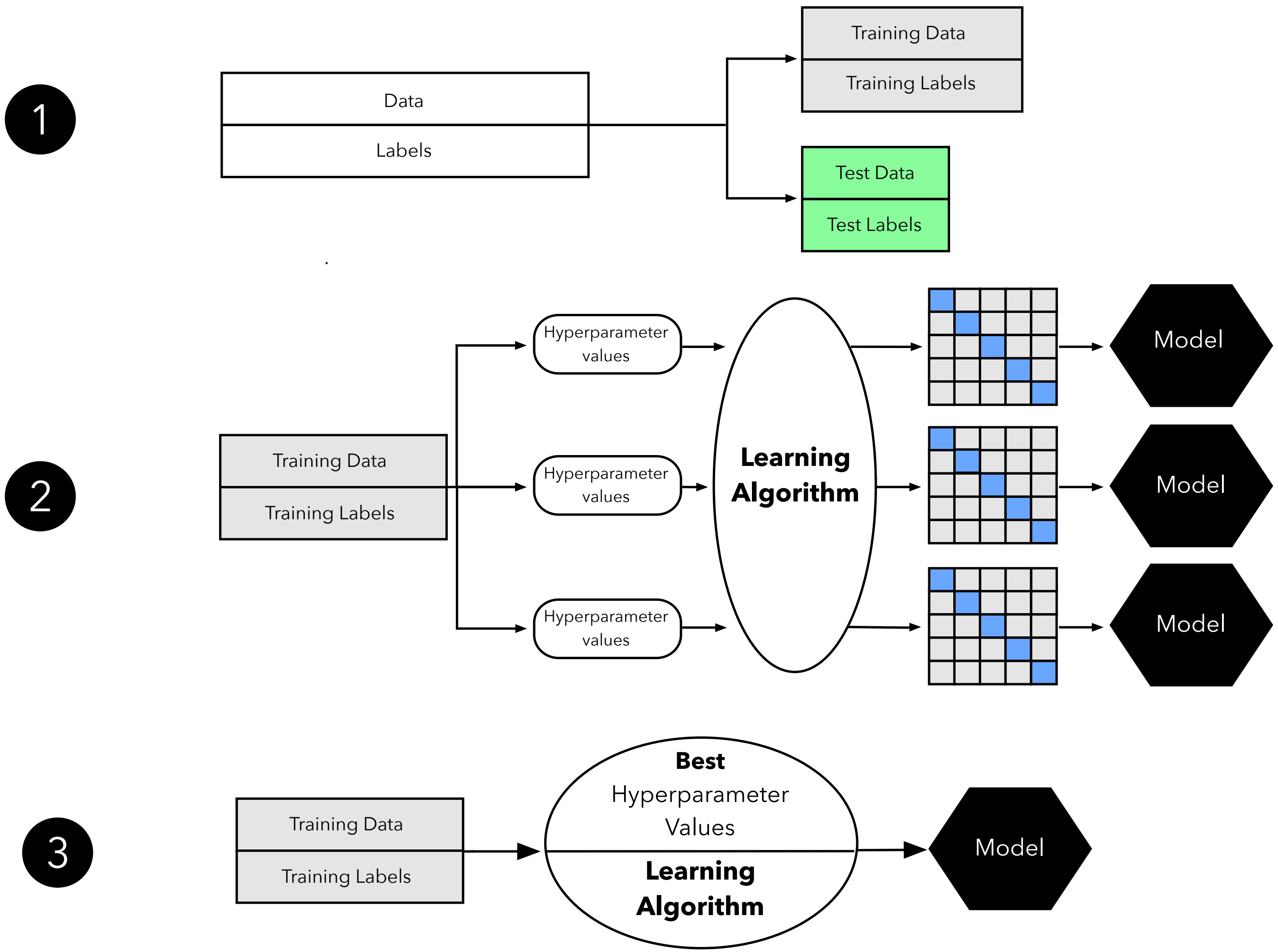
Holdout Method and Hyperparameter Tuning 4-6



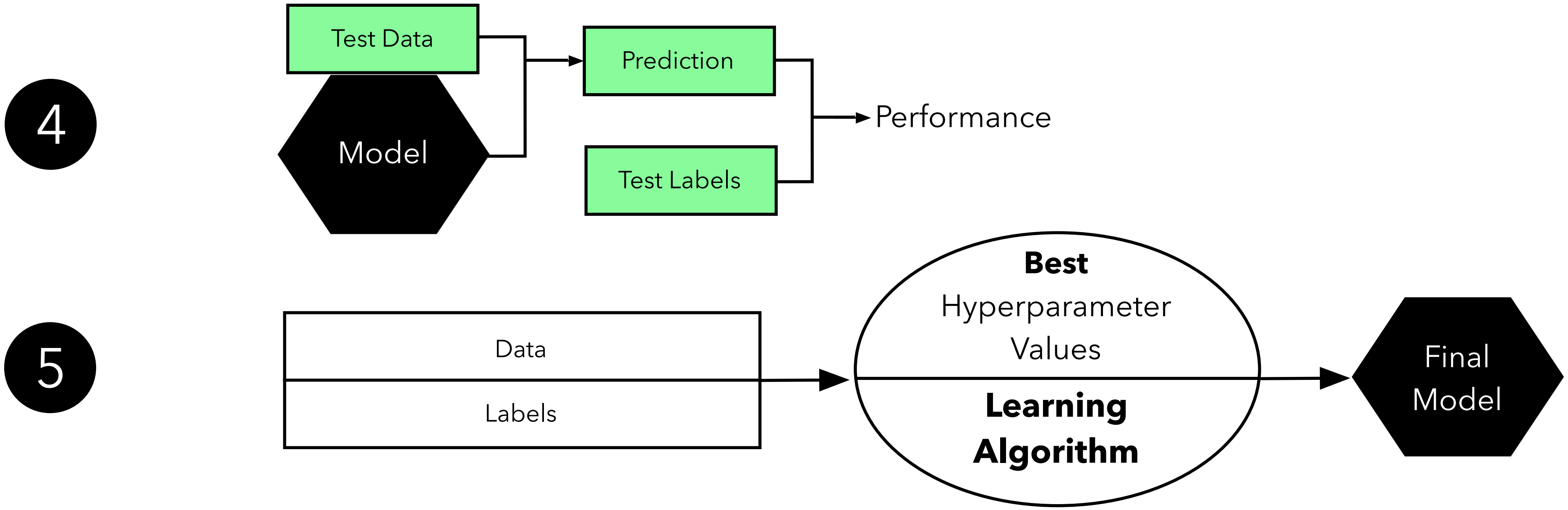
K-fold Cross-Validation



K-fold Cross-Validation Workflow 1-3



K-fold Cross-Validation Workflow 4-5



More info about model evaluation (one of the most important topics in ML):

https://sebastianraschka.com/blog/index.html

- Model evaluation, model selection, and algorithm selection in machine learning Part I - The basics
- Model evaluation, model selection, and algorithm selection in machine learning Part II - Bootstrapping and uncertainties
- Model evaluation, model selection, and algorithm selection in machine learning Part III - Cross-validation and hyperparameter tuning

Jupyter Notebook

BONUS SLIDES

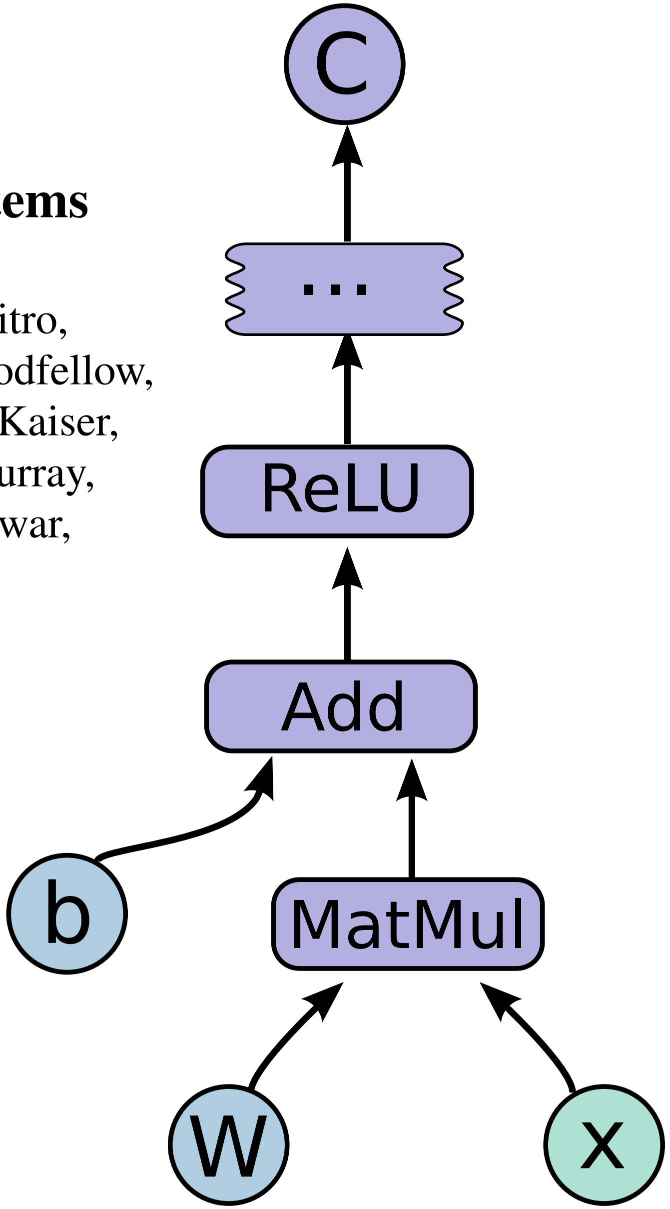
TensorFlow

https://www.tensorflow.org

TensorFlow:
Large-Scale Machine Learning on Heterogeneous Distributed Systems
(Preliminary White Paper, November 9, 2015)

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng
Google Research*

https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45166.pdf



Tensors?

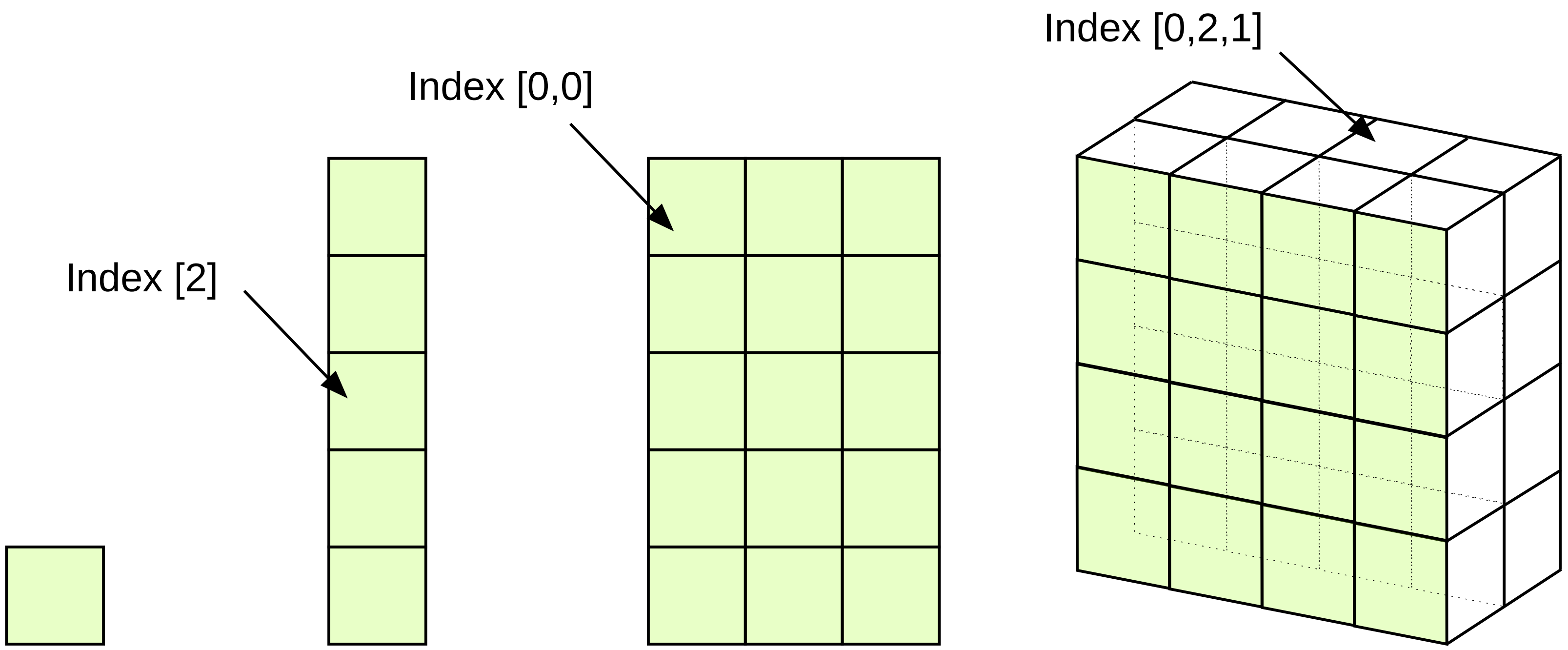
Scalar: $\mathbb{R}$

Vector: $\mathbb{R}^n$

Matrix: $\mathbb{R}^n \times \mathbb{R}^m$

3-Tensor: $\mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p$

...



rank 0 tensor
dimensions [ ]
scalar

rank 1 tensor
dimensions [5]
vector

rank 2 tensor
dimensions [5, 3]
matrix

rank 3 tensor
dimensions [4, 4, 2]

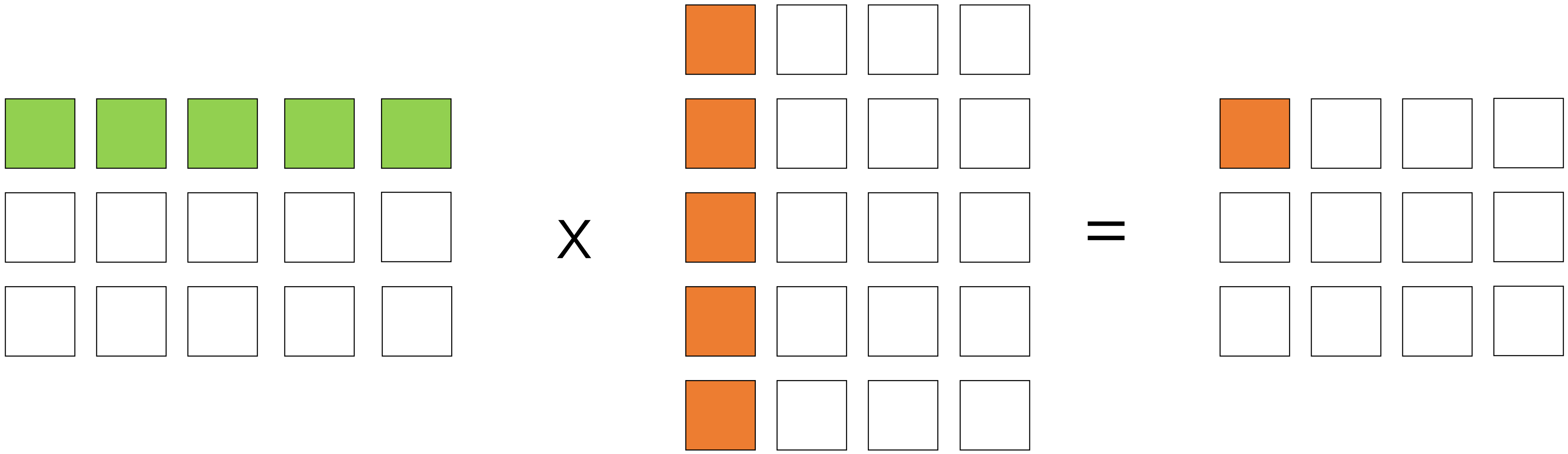
https://sebastianraschka.com/pdf/books/dlb/appendix_g_tensorflow.pdf

GPUs

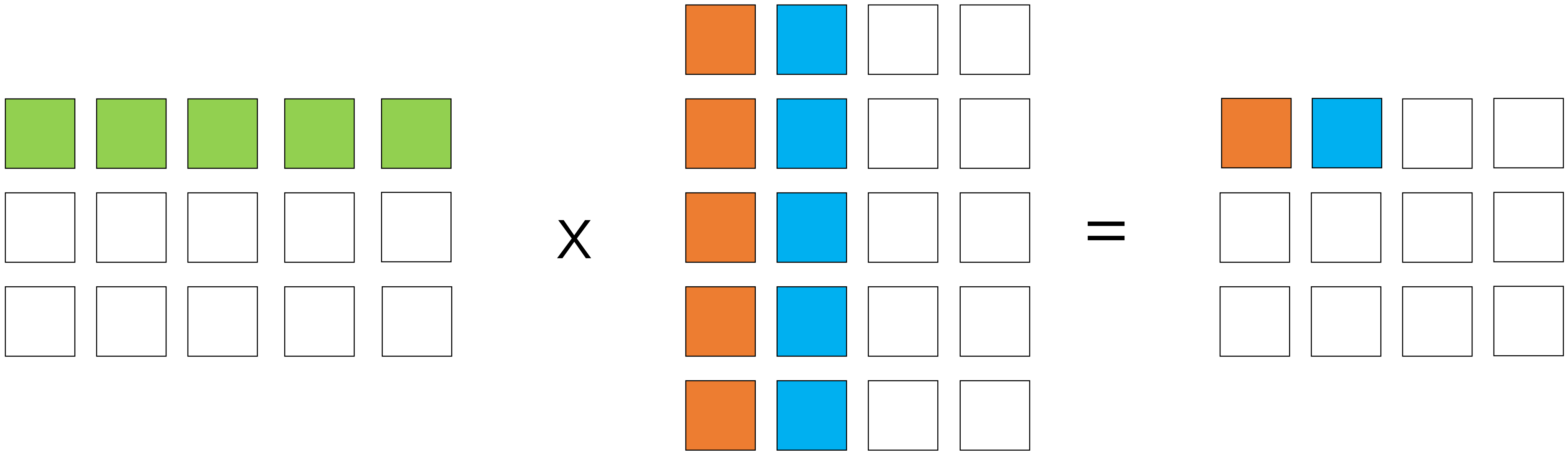
Specifications	Intel® Core™ i7-6900K Processor Extreme Ed.	NVIDIA GeForce® GTX™ 1080 Ti
Base Clock Frequency	3.2 GHz	< 1.5 GHz
Cores	8	3584
Memory Bandwidth	64 GB/s	484 GB/s
Floating-Point Calculations	409 GFLOPS	11300 GFLOPS
Cost	~ $1000.00	~ $700.00

Vectorization

```
X = np.random.random((num_train_examples, num_features))
W = np.random.random((num_features, num_hidden))
```

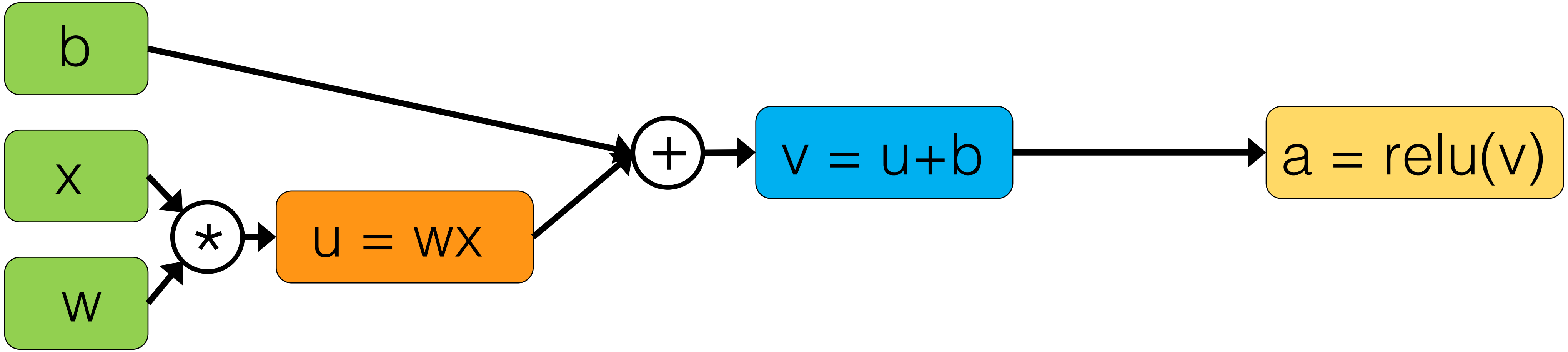


Vectorization



Computation Graphs

$$a(x, w, b) = \mathrm{relu}(\underbrace{\underbrace{w^*x}_{u} + b}_{v})$$

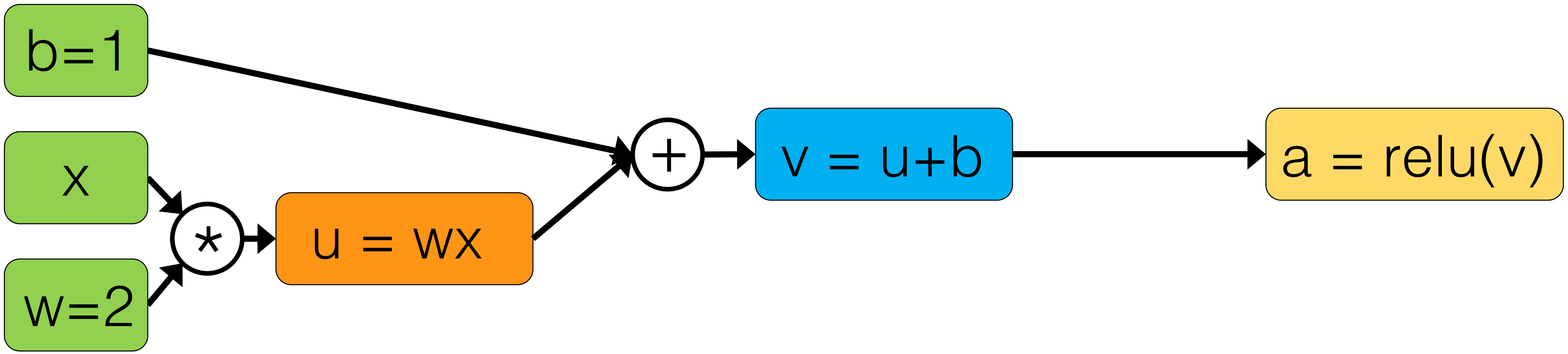


Computation Graphs

```
import tensorflow as tf

g = tf.Graph()
with g.as_default() as g:

    x = tf.placeholder(dtype=tf.float32, shape=None, name='x')
    w = tf.Variable(initial_value=2, dtype=tf.float32, name='w')
    b = tf.Variable(initial_value=1, dtype=tf.float32, name='b')

    u = x * w
    v = u + b
    a = tf.nn.relu(v)

print(x, w, b, u, v, a)
```

```
Tensor("x:0", dtype=float32) <tf.Variable 'w:0' shape=() dtype=float32_ref> <tf.Variable
'b:0' shape=() dtype=float32_ref> Tensor("mul:0", dtype=float32) Tensor("add:0",
dtype=float32) Tensor("Relu:0", dtype=float32)
```

Computation Graphs



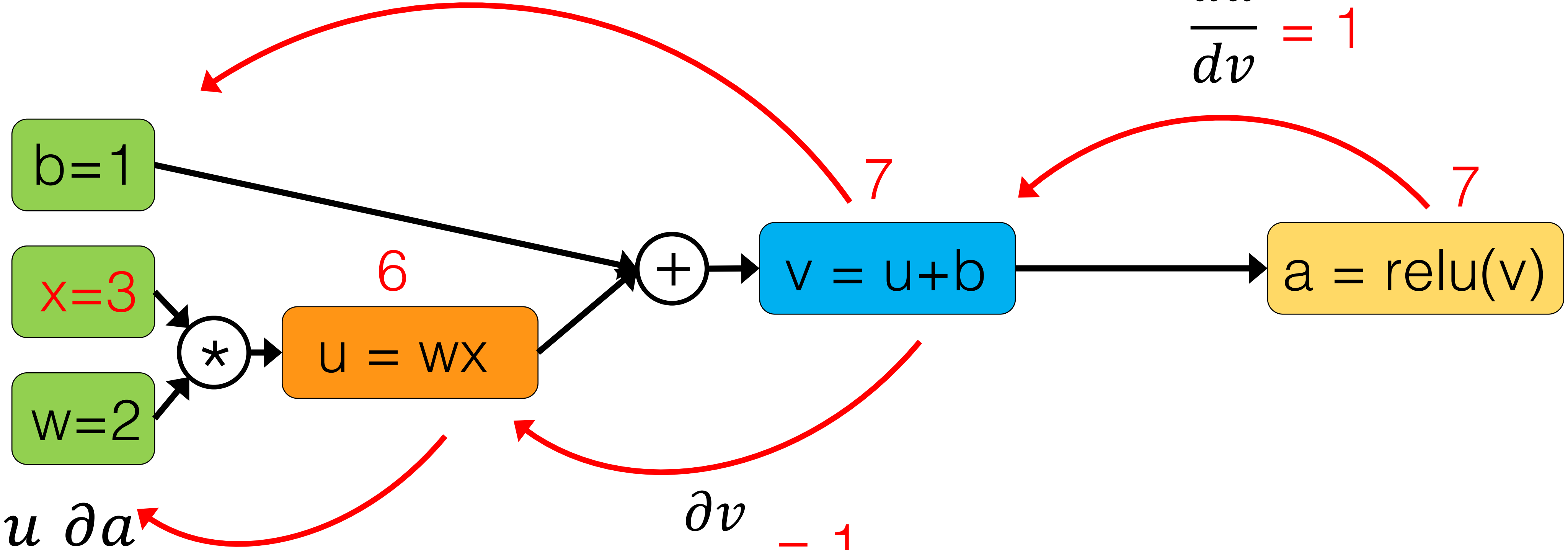

```
with tf.Session(graph=g) as sess:
    sess.run(init_op)
    b_res = sess.run('b:0')

print(b_res)
```

```
1.0
```

$$\frac{\partial a}{\partial b} = \frac{\partial v}{\partial b}\frac{\partial a}{\partial v} = 1$$

$$\frac{\partial v}{\partial b} = 1$$

$$\frac{da}{dv} = 1$$



$$\frac{\partial v}{\partial u} = 1$$

$$\frac{\partial a}{\partial w} = \frac{\partial u}{\partial w}\frac{\partial a}{\partial u}$$

$$\frac{\partial u}{\partial w} = 3$$

$$= \frac{\partial u}{\partial w}\frac{\partial v}{\partial u}\frac{\partial a}{\partial v} = 3*1*1 = 3$$

https://github.com/rasbt/pydata-annarbor2017-dl-tutorial

```python
g = tf.Graph()
with g.as_default() as g:

    x = tf.placeholder(dtype=tf.float32, shape=None, name='x')
    w = tf.Variable(initial_value=2, dtype=tf.float32, name='w')
    b = tf.Variable(initial_value=1, dtype=tf.float32, name='b')

    u = x * w
    v = u + b
    a = tf.nn.relu(v)

    d_a_w = tf.gradients(a, w)
    d_b_w = tf.gradients(a, b)

with tf.Session(graph=g) as sess:
    sess.run(tf.global_variables_initializer())
    res = sess.run([d_a_w, d_b_w], feed_dict={'x:0': 3})
```

[3.0] [1.0]

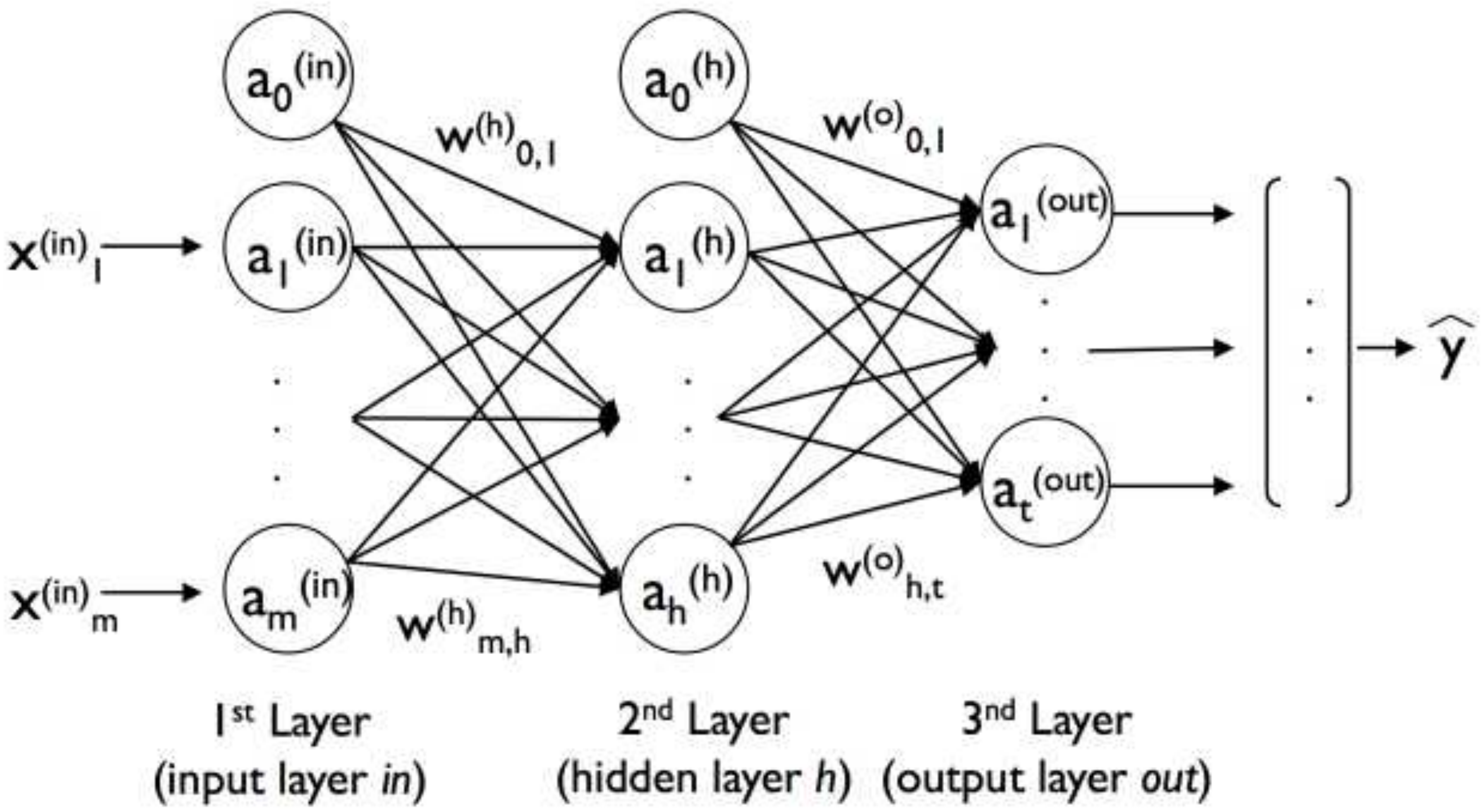
PYTORCH

http://pytorch.org

```python
import torch
import torch.nn.functional as F
from torch.autograd import Variable
from torch.autograd import grad


x = Variable(torch.Tensor([3]))
w = Variable(torch.Tensor([2]), requires_grad=True)
b = Variable(torch.Tensor([1]), requires_grad=True)

u = x * w
v = u + b
a = F.relu(v)

partial_derivatives = grad(a, (w, b))

for name, grad in zip("wb", (partial_derivatives)):
    print('d_a_%s:' % name, grad)
```

```
d_a_w: Variable containing:
 3
[torch.FloatTensor of size 1]

d_a_b: Variable containing:
 1
[torch.FloatTensor of size 1]
```

Multilayer Perceptron



https://github.com/rasbt/python-machine-learning-book-2nd-edition/blob/master/code/ch12/images/12_02.png

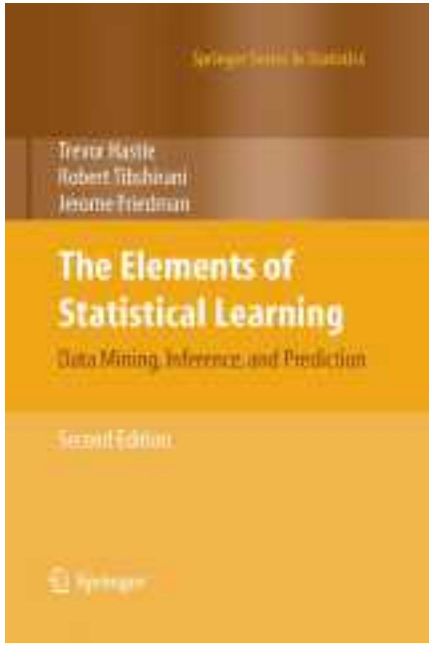
TensorFlow

```python
g = tf.Graph()
with g.as_default():

    # Input data
    tf_x = tf.placeholder(tf.float32, [None, n_input], name='features')
    tf_y = tf.placeholder(tf.float32, [None, n_classes], name='targets')

    # Model parameters
    weights = {
        'h1': tf.Variable(tf.truncated_normal([n_input, n_hidden_1], stddev=0.1)),
        'out': tf.Variable(tf.truncated_normal([n_hidden_2, n_classes], stddev=0.1))
    }
    biases = {
        'b1': tf.Variable(tf.zeros([n_hidden_1])),
        'out': tf.Variable(tf.zeros([n_classes]))
    }

    # Multilayer perceptron
    layer_1 = tf.add(tf.matmul(tf_x, weights['h1']), biases['b1'])
    layer_1 = tf.nn.relu(layer_1)
    out_layer = tf.matmul(layer_1, weights['out']) + biases['out']

    # Loss and optimizer
    loss = tf.nn.softmax_cross_entropy_with_logits(logits=out_layer, labels=tf_y)
    cost = tf.reduce_mean(loss, name='cost')
    optimizer = tf.train.GradientDescentOptimizer(learning_rate=learning_rate)
    train = optimizer.minimize(cost, name='train')

    # Prediction
    correct_prediction = tf.equal(tf.argmax(tf_y, 1), tf.argmax(out_layer, 1))
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32), name='accuracy')

with tf.Session(graph=g) as sess:
    sess.run(tf.global_variables_initializer())

    for epoch in range(training_epochs):
        avg_cost = 0.
        total_batch = mnist.train.num_examples // batch_size

        for i in range(total_batch):
            batch_x, batch_y = mnist.train.next_batch(batch_size)
            _, c = sess.run(['train', 'cost:0'], feed_dict={'features:0': batch_x,
                                                            'targets:0': batch_y})
```

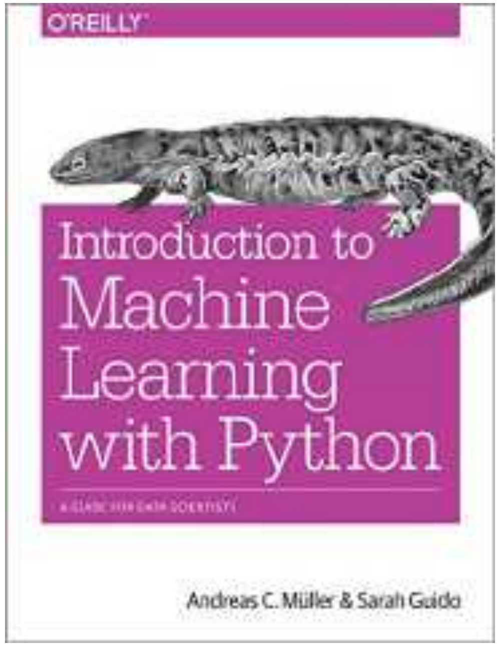
PYTORCH

```python
class MultilayerPerceptron(torch.nn.Module):

    def __init__(self, num_features, num_classes):
        super(MultilayerPerceptron, self).__init__()

        ### 1st hidden layer
        self.linear_1 = torch.nn.Linear(num_features, num_hidden_1)

        ### Output layer
        self.linear_out = torch.nn.Linear(num_hidden_2, num_classes)

    def forward(self, x):
        out = self.linear_1(x)
        out = F.relu(out)
        logits = self.linear_out(out)
        probas = F.softmax(logits, dim=1)
        return logits, probas

model = MultilayerPerceptron(num_features=num_features,
                             num_classes=num_classes)

if torch.cuda.is_available():
    model.cuda()

for epoch in range(num_epochs):
    for batch_idx, (features, targets) in enumerate(train_loader):

        features = Variable(features.view(-1, 28*28))
        targets = Variable(targets)

        if torch.cuda.is_available():
            features, targets = features.cuda(), targets.cuda()

        ### FORWARD AND BACK PROP
        logits, probas = model(features)
        cost = cost_fn(logits, targets)
        optimizer.zero_grad()

        cost.backward()

        ### UPDATE MODEL PARAMETERS
        optimizer.step()
```

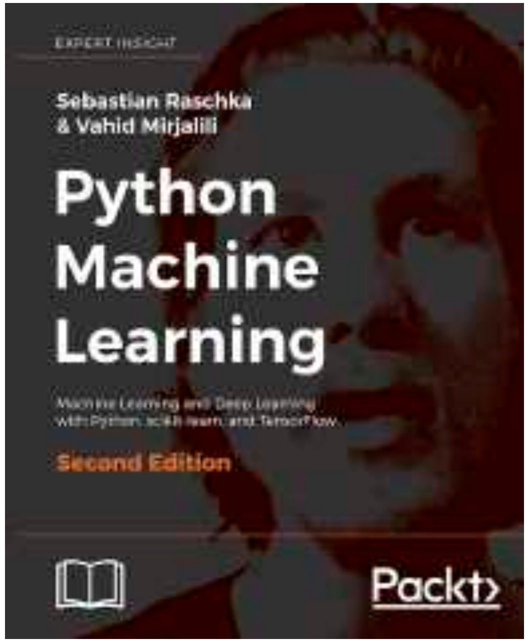
Further Resources



Math-heavy



Math-free scikit-learn intro



Mix of code & math
(~60% scikit-learn)

Thanks for attending!

Tutorial Material on GitHub:

https://github.com/rasbt/msu-datascience-ml-tutorial-2018

Contact:

- E-mail: mail@sebastianraschka.com
- Website: http://sebastianraschka.com
- Twitter: @rasbt
- GitHub: rasbt