Advanced Python Data Analytics

Machine Learning with Python

March 20th, 2018

Kang Pyo Lee
UI3 / ITS-RS

UI3
Iowa Informatics Initiative

Course Outline

- **Session 1: Lecture on Basic Concepts (12:30 – 1:30)**
  - Basics of Machine Learning
  - Python as a Data Analytics Tool
- **Session 2: Demonstration of Python Machine Learning (1:30 – 3:30)**
  - Part 0: Jupyter Notebook
  - Part 1: Supervised Learning – Regression (k-NN, Linear Regression)
  - Part 2: Supervised Learning – Classification (k-NN, Logistic Regression, SVMs, Neural Networks)
  - Part 3: Unsupervised Learning – Clustering (K-Means Clustering)
  - Part 4: Unsupervised Learning – Dimensionality Reduction (PCA)

Introduction to the Instructor

- **Name**: Kang Pyo Lee
- **Motto**: “Learn from data!”
- **Education**: Ph.D. in Computer Science at Seoul National University in 2012
- **Previous Work**: Data Scientist at Samsung Big Data Center
- **Current Work**: Research Data Scientist at UI3 and ITS-RS
- **Research Interests**: data science, big data, social media analytics, etc.

Goal & Scope of This Course

Machine learning is a complex topic

However, Python can help you build and evaluate machine learning models very quickly and easily

Goal & Scope of This Course

This course will focus on how to do machine learning with Python

- How to **build models** with Python
- How to **evaluate the models** with Python
- How to **manipulate data** with Python

Target Audience of This Course

This course would be best for aspiring machine learning practitioners

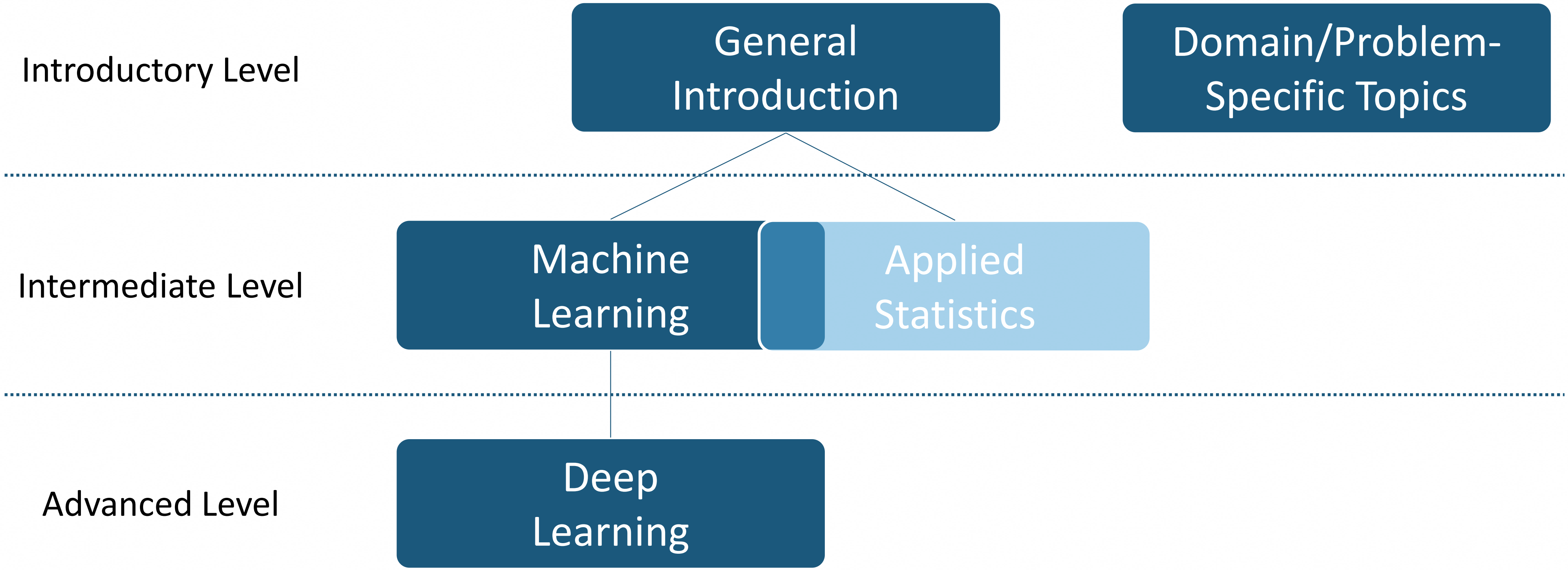
who want to learn how to apply machine learning techniques to their own data and problems

Target Audience of This Course

What about beginners in machine learning?

This course will help them get started with machine learning in Python

Big Picture of (Python) Data Analytics Training



Data Analytics Hierarchy

Artificial Intelligence

Machine Learning

Supervised Learning

Unsupervised Learning

Deep Learning

Supervised vs. Unsupervised Learning

Supervised Learning **Vs.** Unsupervised Learning

Criteria – whether or not there is feedback available to the learning system	
Learns from gold standard (a.k.a. training data, example data, labeled data, etc.)	Learns with no gold standard
Based on example inputs (*X*) and their outputs (*y*), aims to learn a general rule that maps new inputs (*X_new*) to their best possible outputs (*predictions*)	Learns on its own to find structures, or patterns, inherent in its inputs (*X*)
Regression, classification	Clustering, dimensionality reduction
Easier and more straightforward to evaluate	Harder to evaluate

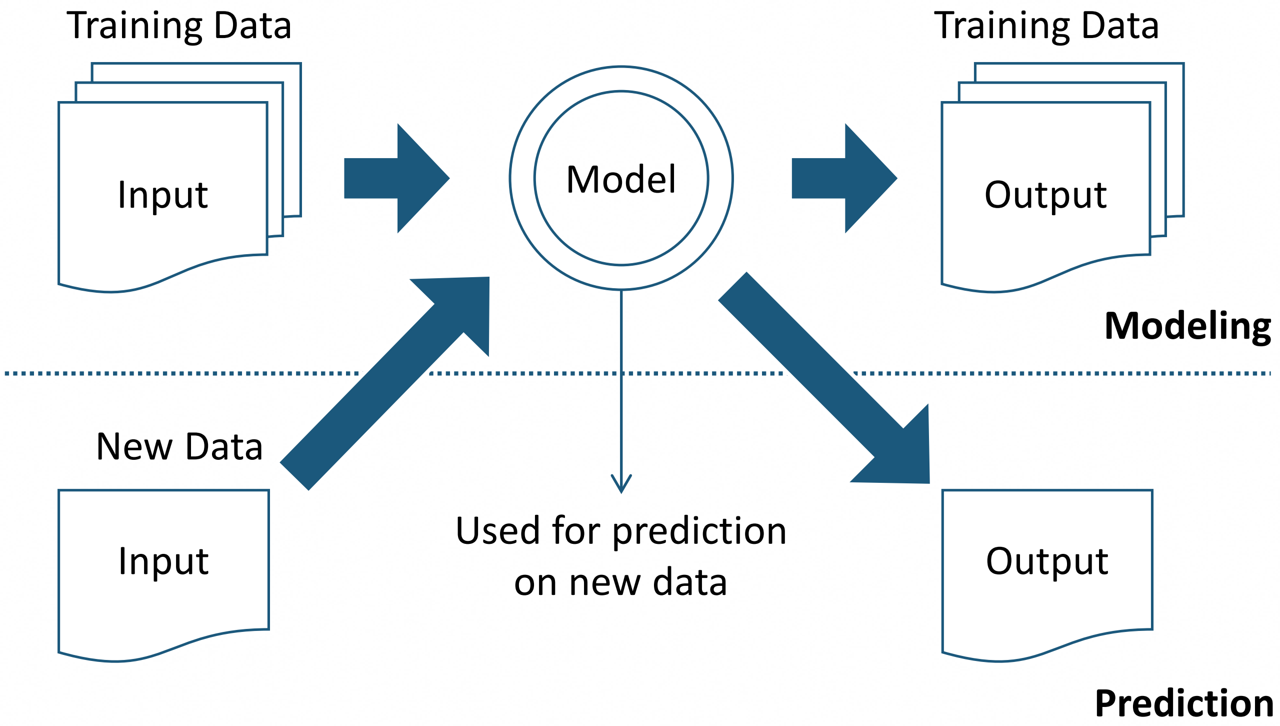
prediction **pattern**

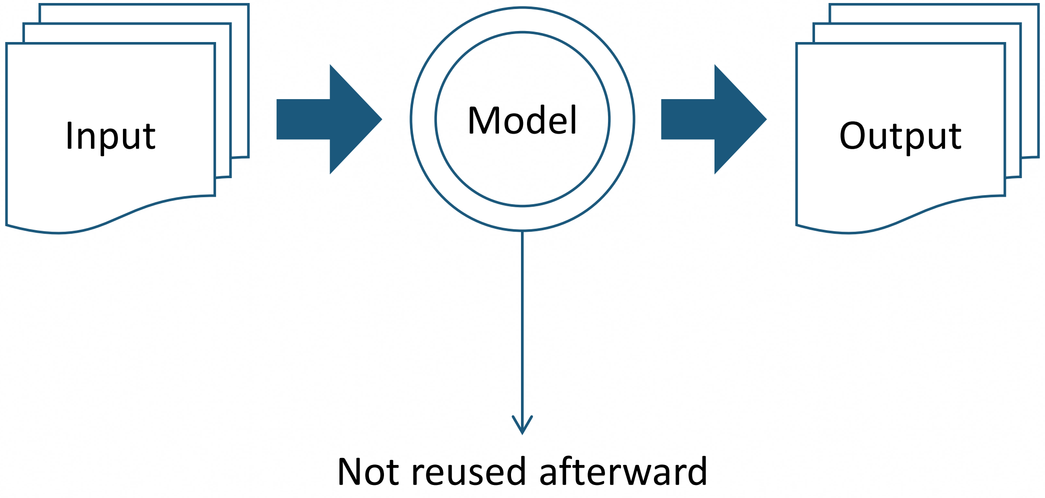
Supervised vs. Unsupervised Learning

Supervised Learning

Vs.

Unsupervised Learning





Regression vs. Classification

Regression Vs. Classification

Both belong to supervised learning	
Criteria – whether or not there is continuity between possible outcomes	
Aims to predict a **continuous number**	Aims to predict a **class label**, which is a choice from a predefined list of possibilities
E.g., predicting a person's annual income from their education, their age, where they live, etc.	• Binary classification: only two classes (e.g., yes/no, negative/positive, survive/die, spam/nonspam) • Multiclass classification: more than two classes (e.g., weather as sunny, cloudy, rainy, or snowy)
k-Nearest Neighbors (k-NN), Linear Regression	k-Nearest Neighbors (k-NN), Logistic Regression, Support Vector Machines (SVMs), Naïve Bayes Classifiers, Decision Trees, Neural Networks

Classification vs. Clustering

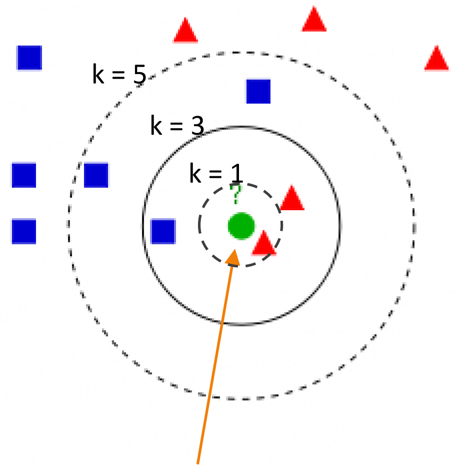
Classification Vs. Clustering

Both aim to divide the data into meaningful segments	
Criteria – whether or not there are predefined classes	
Aims to classify the data into one of the **predefined categorical classes**	Aims to map the data into one of several clusters of **similar data items**
Supervised learning → training sample provided	Unsupervised learning → no training sample
E.g., weather as sunny, cloudy, rainy, or snowy	E.g., clustering of similar news articles in a large news article data collection
k-Nearest Neighbors (k-NN), Logistic Regression, Support Vector Machines (SVMs), Naïve Bayes Classifiers, Decision Trees, Neural Networks	k-Means Clustering, Agglomerative Clustering, DBSCAN, Topic Modeling

Supervised Learning – k-Nearest Neighbors (k-NNs)

- One of the simplest machine learning algorithms
- How it works
  - For a new data point, finds the point in the training set that is **closest, or nearest,** to the new point
  - *k* = the number of the closest neighbors to consider
  - Makes a prediction using the majority class among these *k* nearest neighbors
- Used for both regression and classification
- Strengths
  - Very easy to understand
  - Often gives reasonable performance without a lot of adjustments
  - A good baseline method to try before considering advanced techniques
- Weaknesses
  - Slow in prediction for large training datasets
  - Does not perform well on datasets with many features (hundreds or more) or sparse datasets
  - Not often used in practice

Classification into Two Classes



New data point

Supervised Learning – Linear Regression

- Makes a prediction about a **continuous number** using a **linear function** of the input features
- Finds the parameters w and b that minimize the error between predictions ($\hat{y}$) and the true values (y)

Model → $$\underbrace{\hat{y}}_{\text{income}} = w[0] * \underbrace{x[0]}_{\text{education}} + w[1] * \underbrace{x[1]}_{\text{age}} + \cdots + w[p] * x[p] + b$$

$$\begin{aligned} &where\ \hat{y}: the\ prediction\ the\ model\ makes \\ &\qquad x[0]\ to\ x[p]: features \\ &\qquad w\ and\ b: parameters\ to\ be\ learned \end{aligned}$$

Training Dataset

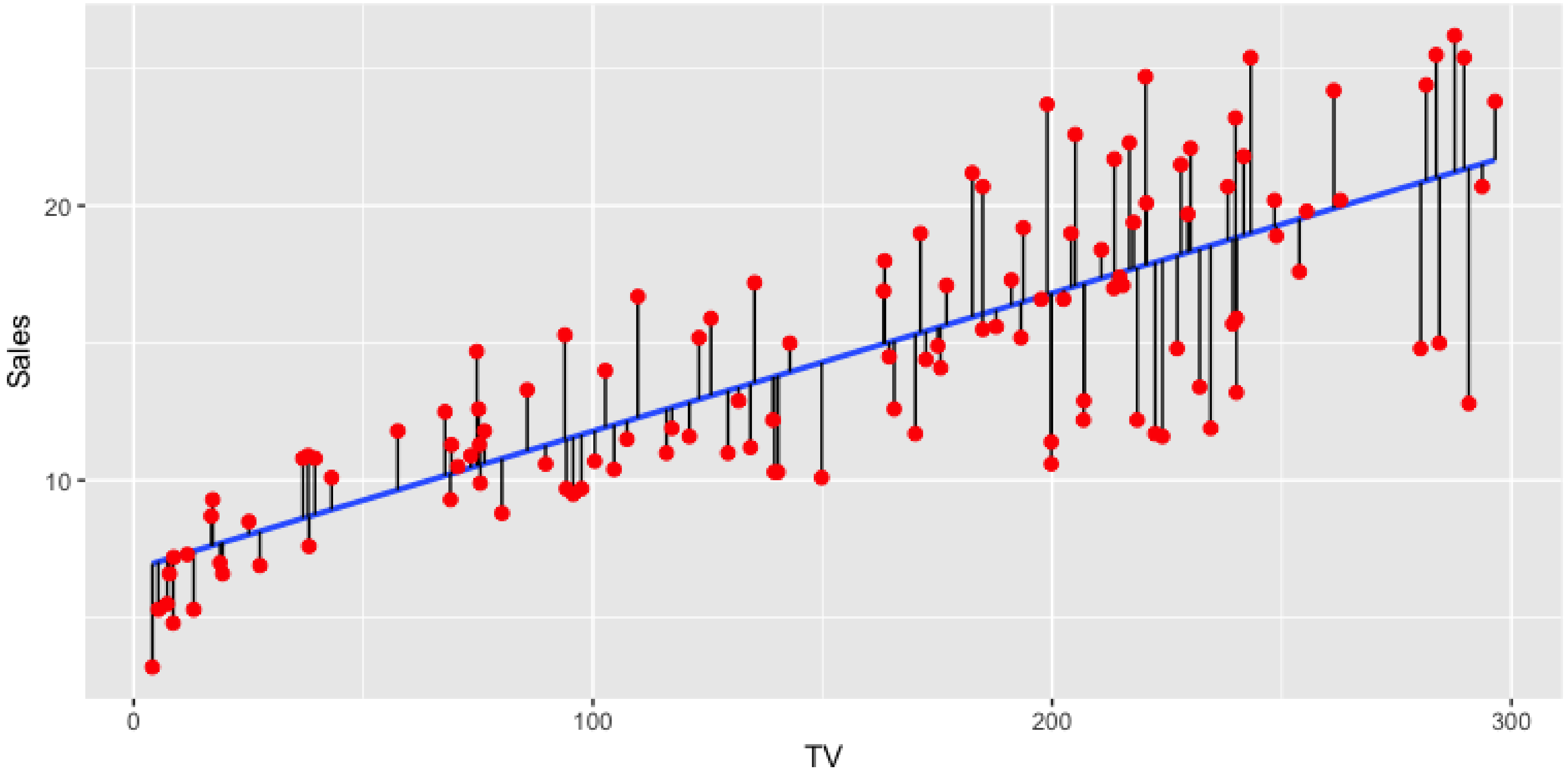
1st example → $$y_0 = w[0] * x_0[0] + w[1] * x_0[1] + \cdots + w[p] * x_0[p] + b$$

2nd example → $$y_1 = w[0] * x_1[0] + w[1] * x_1[1] + \cdots + w[p] * x_1[p] + b$$

$$\vdots$$

Supervised Learning – Linear Regression

Find a strict line that minimizes the sum of squared errors between predictions ($\hat{y}$) and the true values (y)



Supervised Learning – Logistic Regression

- Makes a prediction about a **class label** using a **linear function** of the input features
- Finds the parameters w and b that minimizes the error between predictions ($\hat{y}$) and the true values (y)

Model → $\hat{y} = g(w[0] * x[0] + w[1] * x[1] + \cdots + w[p] * x[p] + b)$

spam/nonspam length of the title contains a URL

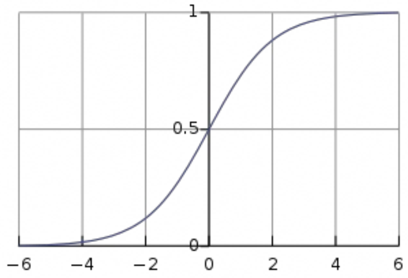
$where\ \hat{y}: the\ prediction\ the\ model\ makes$

$x[0]\ to\ x[p]: features$

$w\ and\ b: parameters\ to\ be\ learned$

If g is larger than 0.5, we predict the class as +1; if smaller than 0.5, we predict the class as -1

Binary classification



$g = sigmoid$

Training Dataset

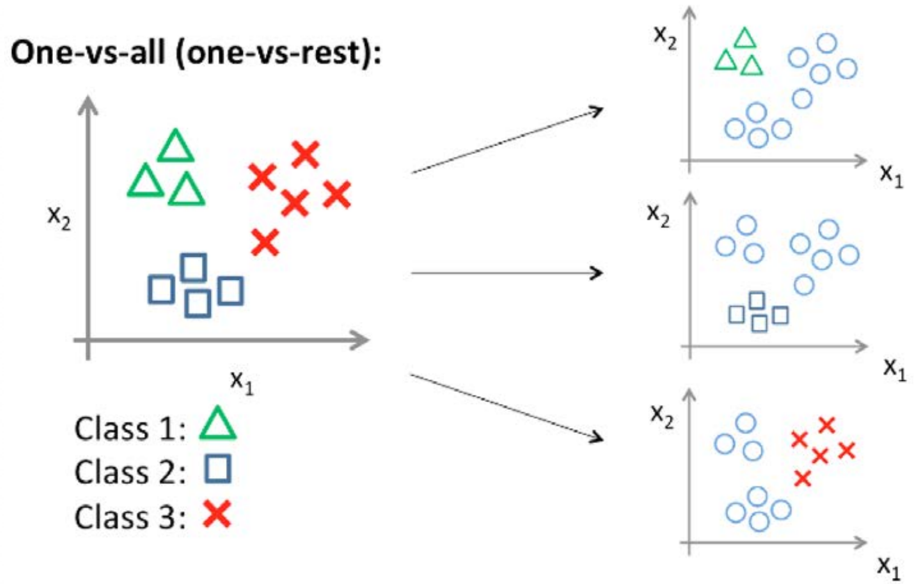
1st example → $y_0 = g(w[0] * x_0[0] + w[1] * x_0[1] + \cdots + w[p] * x_0[p] + b)$

2nd example → $y_1 = g(w[0] * x_1[0] + w[1] * x_1[1] + \cdots + w[p] * x_1[p] + b)$

⋮

Supervised Learning – Logistic Regression

- Multiclass classification
  - E.g., weather prediction as sunny, cloudy, rainy, or snowy
  - ***One-vs.-all*** (*one-vs.-rest*) approach
    - A multiclass classification problem with *n* classes can be decomposed into *n* binary classification problems
    - A binary model is learned for each class that separates that class from all of the other classes, resulting in as many binary models as there are classes (= *n*)
    - To make a prediction, all binary classifiers are run on a new point, and the classifier with the highest score on its single class wins, and this class label is returned as the prediction



Supervised Learning – Linear Models

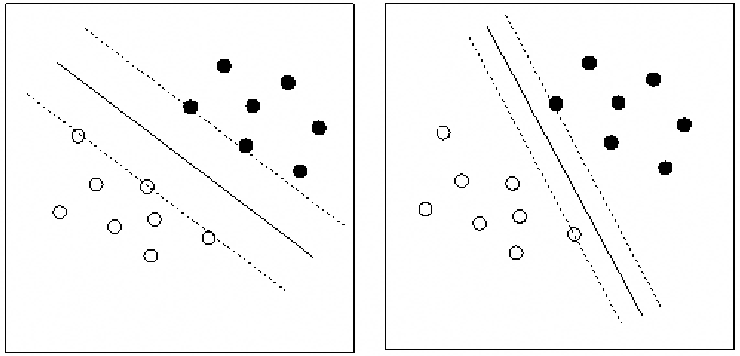
- Strengths
  - Fast to train and to predict
  - Scale to very large datasets and work well with sparse data
  - Relatively easy to understand how a prediction is made using linear functions
  - Often perform well when the number of features is large compared to the number of training samples

- Weaknesses
  - Based on an assumption that the target variable can be predicted by a linear combination of feature variables, which may be too weak to apply to real world problems
  - Other models may yield better generalization performance in lower-dimensional spaces

Supervised Learning – Support Vector Machines (SVMs)

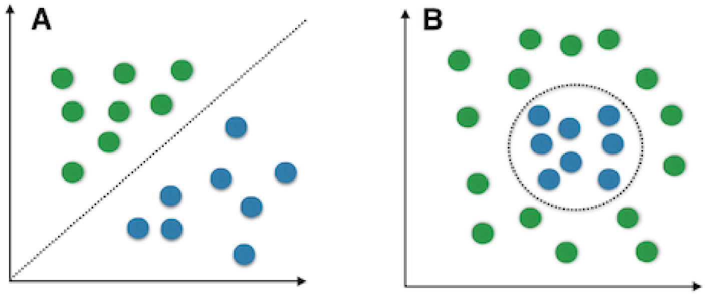
- Based on the **large margin** intuition
  - Find the maximum-margin hyperplane that represents the largest separation, or margin, between two classes
- Typically only a subset of the training points matter for defining the decision boundary: the ones that lie on the border between the classes → called **support vectors**
- To make a prediction for a new data point,
  - The distance to each of the support vectors is measured
  - A classification decision is made based on the distances to the support vector and the weights of the support vector that were learned during training
- The distance between data points can be measured by Gaussian kernel:

$$k_{rbf}(x_1, x_2) = \exp(-\underbrace{\gamma}_{\text{Controls the width of the Gaussian kernel}} \underbrace{\|x_1 - x_2\|^2}_{\text{Euclidean distance of two data points}})$$



(a) Larger margin (b) Smaller margin

Linear vs. nonlinear problems



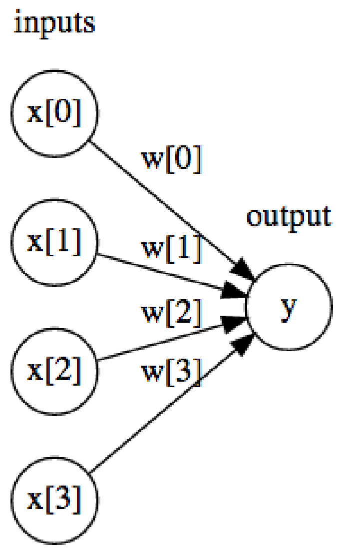
Supervised Learning – Support Vector Machines (SVMs)

- Strengths
  - Perform very well on a variety of datasets
  - Allow for complex decision boundaries, even if the data has only a few features
  - Work well on low-dimensional data with few features and high-dimensional data with many features

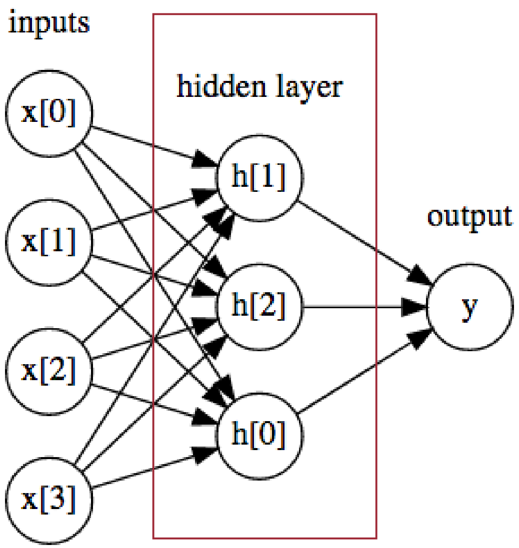
- Weaknesses
  - Very sensitive to the scaling of the data and the settings of the parameters
    - Do not scale very well with the number of training samples in terms of runtime and memory usage
    - Require careful preprocessing of the data and tuning of the parameters
  - Hard to understand why a particular decision was made

Supervised Learning – Neural Networks

- A.k.a. artificial neural networks (ANNs) or multilayer perceptrons (MLPs)
- Inspired by the biological neural networks that constitute animal brains
- Generalizations of linear models that perform **multiple stages of processing** to come to a decision



Logistic regression



MLPs with a single hidden layer

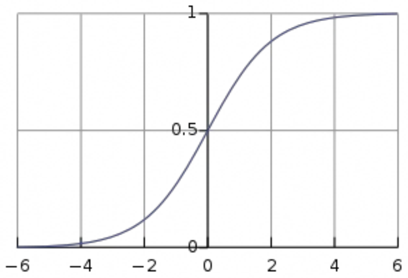
$$h[0] = g(w[0,0] * x[0] + w[1,0] * x[1] + w[2,0] * x[2] + w[3,0] * x[3])$$

$$h[1] = g(w[0,1] * x[0] + w[1,1] * x[1] + w[2,1] * x[2] + w[3,1] * x[3])$$

$$h[2] = g(w[0,2] * x[0] + w[1,2] * x[1] + w[2,2] * x[2] + w[3,2] * x[3])$$

$$y = g(v[0] * h[0] + v[1] * h[1] + v[2] * h[2])$$

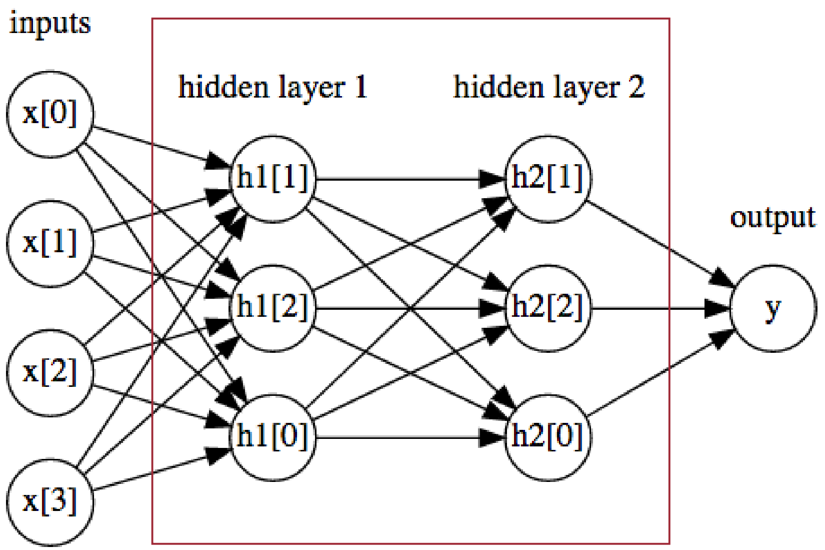
activation function



$g = sigmoid$

Supervised Learning – Neural Networks

- You can control the complexity of neural networks with
  - the number of hidden layers
  - the number of hidden units in each hidden layer

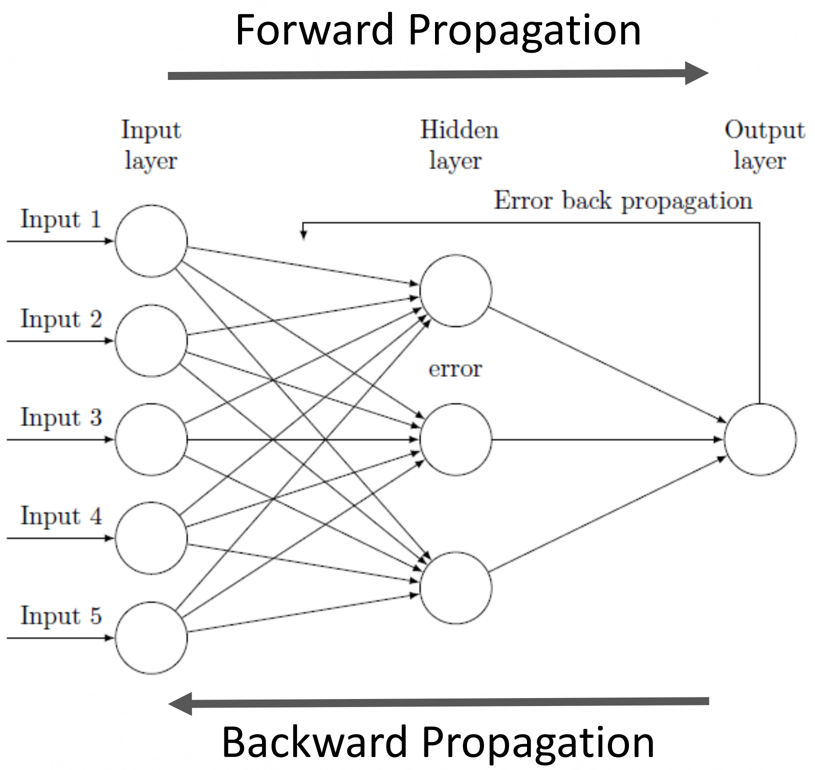


MLPs with two hidden layers

- Random initialization of weights
  - All initial weights are set randomly before learning is started → called random seeds
  - This random initialization can affect the model that is learned, particularly for small networks
  - One effective strategy for random initialization is to randomly select values uniformly in the range $[-\epsilon_{init}, \varepsilon_{init}]$, e.g., $[-0.12, 0.12]$, to ensure the weights are kept small and makes the learning more efficient

Supervised Learning – Neural Networks

- Error adjustment thorough **backpropagation**
  - Given a training example, first run a **forward** pass to compute all the activations through the network
  - For each node in each hidden layer, compute an error term that measures how much that node was "responsible" for any errors in our output during the **backward** pass
  - Repeat for the rest of the training examples



Supervised Learning – Neural Networks

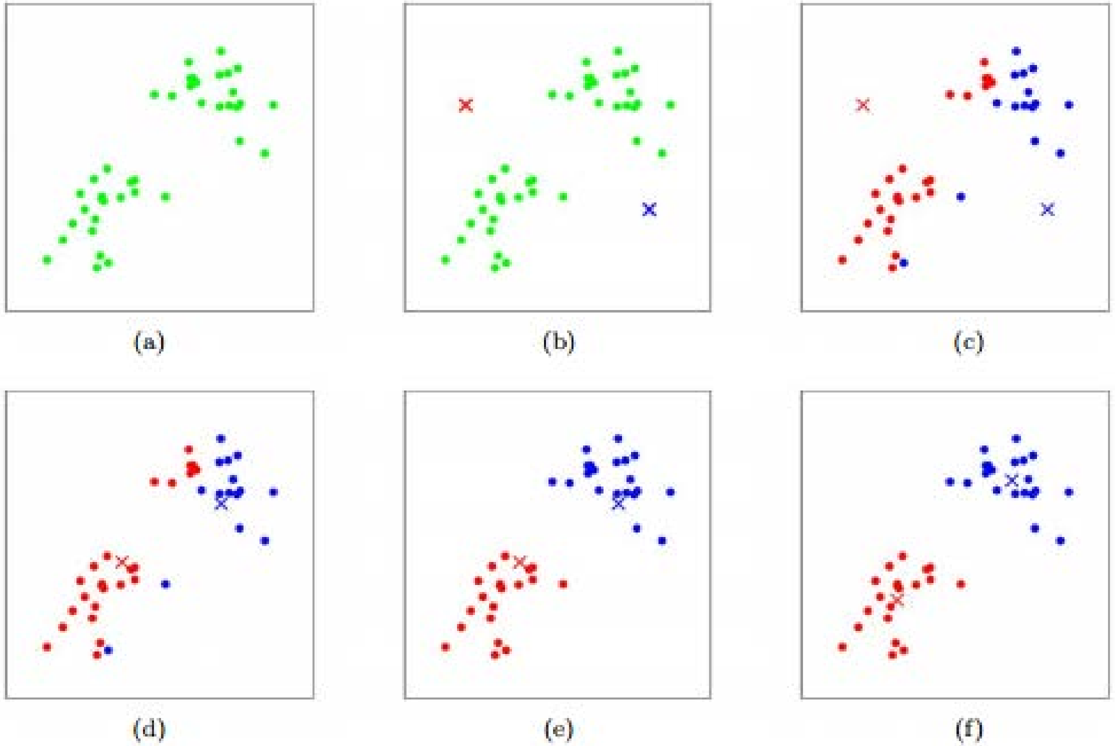
- Strengths
  - Able to capture information contained in large amounts of data and build incredibly complex models → the basis for deep learning
  - Often outperform other machine learning algorithms, given enough computation time, data and careful tuning of the parameters
  - Work best with homogeneous data, where all the features have similar meanings

- Weaknesses
  - Often takes long time to train
  - Require careful preprocessing of the data and tuning of parameters
  - Do not work well with heterogeneous data with very different kinds of features

Unsupervised Learning – k-Means Clustering

- One of the simplest and most commonly used clustering algorithms
- Finds cluster centers, or **centroids**, that are representative of certain regions of the data
- How it works
  - Given the number of clusters (*k*), randomly choose *k* centroids
  - Iterate between two steps:
    - Assign each data point to the closest centroid
    - Update each centroid as the mean of the data points that are assigned to it
  - Finish when the assignment of instances to clusters no longer changes

k = 2



Unsupervised Learning – k-Means Clustering

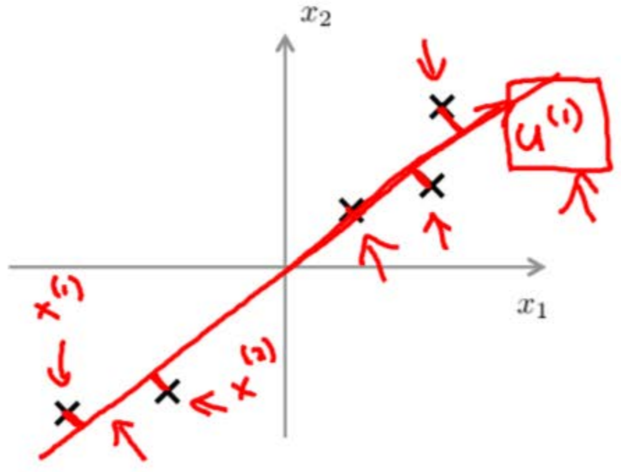
- Strengths
  - Relatively easy to understand and implement
  - Runs relatively quickly
  - Scales easily to large datasets

- Weaknesses
  - The outcome of the algorithm depends on the random initialization of centroids
  - Can only capture clusters of relatively simple shapes
  - Assumes that all directions are equally important for each cluster
  - Requires **the number of clusters (*k*)** you are looking for (which might not be known in a real-world application)

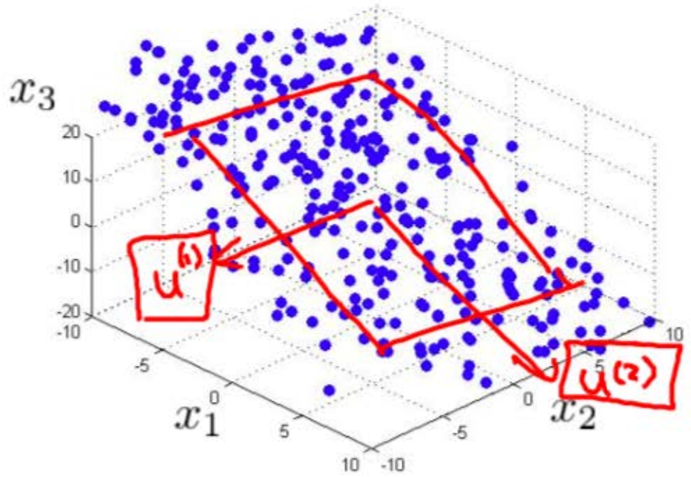
Set *k* to a small number → fewer clusters of more data points → a wider view on the clustered data
Set *k* to a large number → more clusters of fewer data points → a narrower view

Unsupervised Learning – Principal Component Analysis (PCA)

- Converts a set of examples of possibly *n* **correlated** variables into a set of values of *k* linearly **uncorrelated** variables called **principal components**
- Finds *k* vectors onto which to project the data, so as to minimize the projection error
- *N*-dimension is reduced to *k*-dimension, i.e., *k* < *n*
- How it works mathematically
  - Compute a covariance matrix
  - Compute eigenvectors of the covariance matrix
  - Choose the first *k* eigenvectors



Reduce data from 2D to 1D



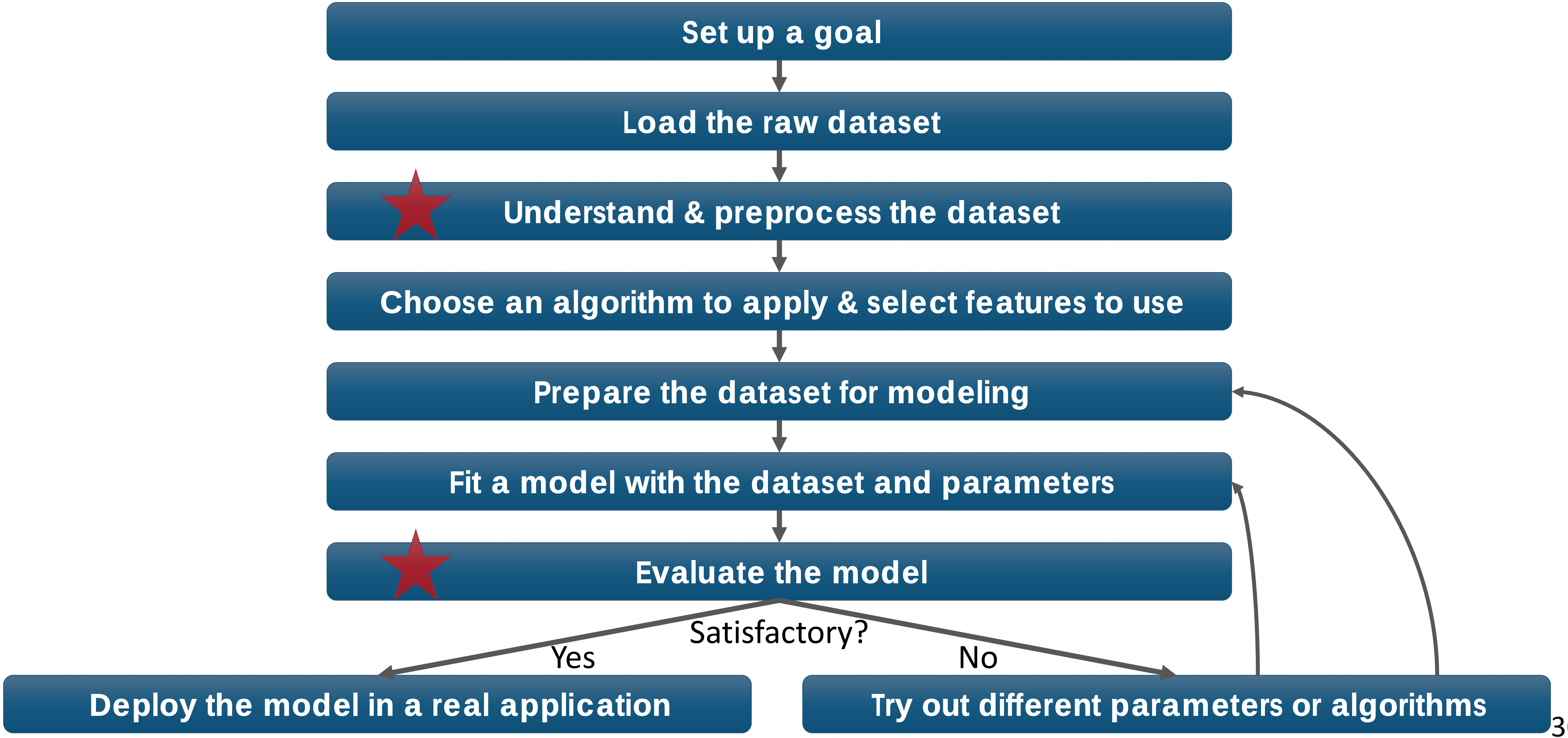
Reduce data from 3D to 2D

Unsupervised Learning – Principal Component Analysis (PCA)

- Strengths
  - Very useful for many practical applications such as feature extraction, data compression and visualization
  - Able to deal with large datasets both in examples and variables
  - No special assumptions on the data → can be applied on all datasets

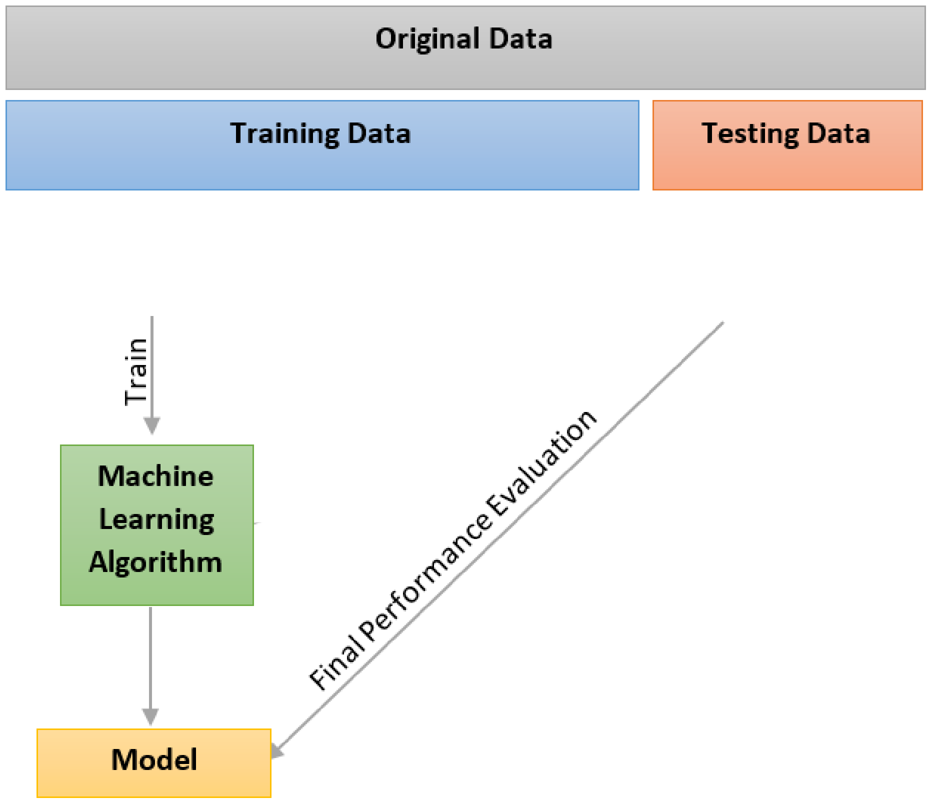
- Weaknesses
  - Hard to model nonlinear structure
  - The meaning of the original variables may be difficult to assess directly on reduced variables

Machine Learning Process



Evaluation of Supervised Models

- Randomly split the data into training and test sets to measure how well the model generalizes to new, previously unseen data
  - Randomly split the dataset into a **training set** (75%) and a **test set** (25%)
  - Build a model on the training set
  - Evaluate the model on the test set



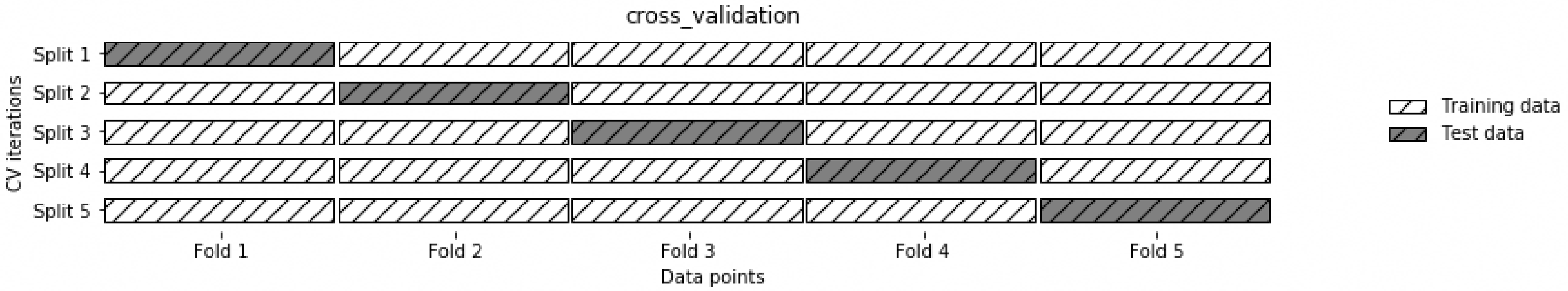
- We are NOT interested in how well the model fits the training set, BUT rather in how well it can make predictions for the test set that was not observed during training

Evaluation of Supervised Models

Accuracy score on **training** data	Accuracy score on **test** data		
Low	Low	→	Underfitting
Low	High	→	Good, but rare
High	Low	→	Overfitting
High	High	→	Excellent

Evaluation of Supervised Models

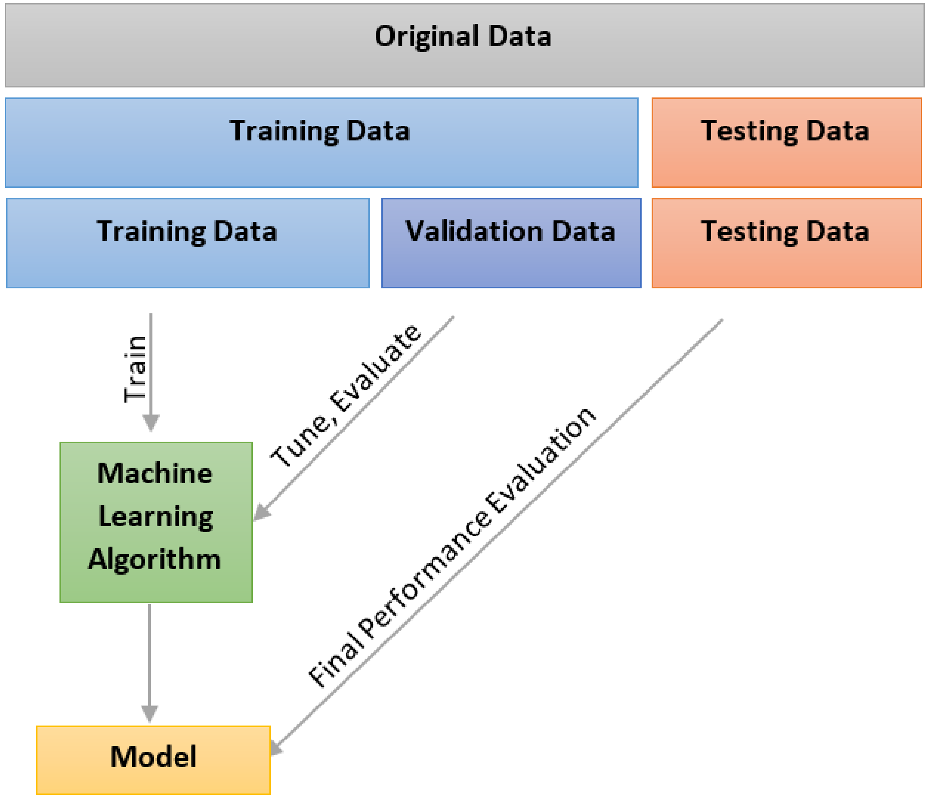
- **Cross validation**
  - A statistical method of evaluating generalization performance that is more stable and thorough than just using a split into a training and a test set
  - The data is split repeatedly and multiple models are trained from different training sets
- ***k*-fold** cross validation
  - The data is first partitioned into *k* parts of (approximately) equal size, called *folds*
  - The first model is trained on the first split using the first fold as the test set and using the remaining *k*-1 folds as the training set
  - The other models are learned on the other splits of training and test sets
  - *k* accuracy scores are collected from the *k* models



Five-fold cross validation

Evaluation of Supervised Models

- When tuning parameters
  - While checking the accuracy scores we tried many different parameters and selected the one with the best accuracy on the test set
  - Can we say the model will generalize well to completely new data? → No!
  - One way to resolve this problem is to split the data into three sets:
    - The training set to build the model
    - The **validation set** to select the best parameters of the model
    - The test set to evaluate the performance of the selected parameters
  - After selecting the best parameters using the validation set, we can rebuild a model using the parameter settings we've found, but this time **training on both the training data and the validation data**, so we can use as much training data as possible to build a final model



Python is a general-purposed high-level programming language

- Web development
- Networking
- Scientific computing
- Data analytics
- Etc.

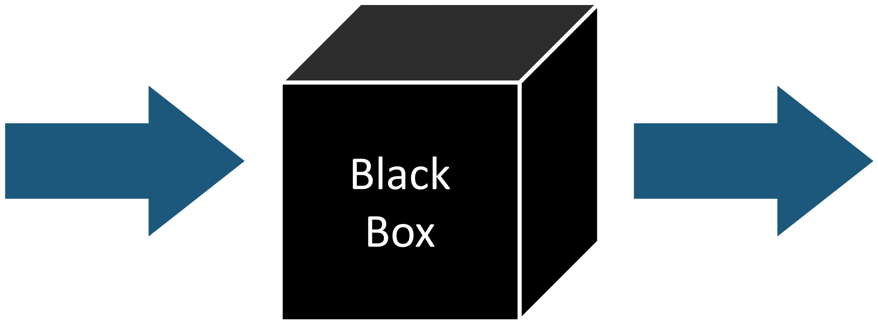
The nature of Python makes it a perfect-fit for data analytics

- Easy to learn
- Readable
- Scalable
- Easy integration with other apps
- An extensive set of libraries
- Active community & ecosystem

Machine Learning Libraries

Python Machine
Learning Library API

Input



Output

- a dataset
- parameters

a model

You don’t have to implement each machine learning algorithm yourself!

All you have to care about is the input and output of the algorithm

Machine Learning Libraries

Reasons you should use machine learning libraries
rather than writing the code yourself

1. **Convenient to use**

2. **Proven to be error-free**

3. **Much faster than your code**

* If you’re a student who wants to learn Machine Learning in depth, implementing the algorithms yourself would greatly help you understand how they actually work

Popular Python Data Analytics Libraries

Library	Usage
numpy, scipy	Numerical & scientific computing
pandas	Data manipulation & aggregation
mlpy, scikit-learn	Machine learning
keras, tensorflow, theano	Deep learning
statsmodels	Statistical analysis
nltk, gensim, textblob	Text processing
networkx	Network analysis & visualization
bokeh, matplotlib, plotly, seaborn	Visualization
beautifulsoup, scrapy, selenium	Web scraping

iPython & Jupyter Notebook

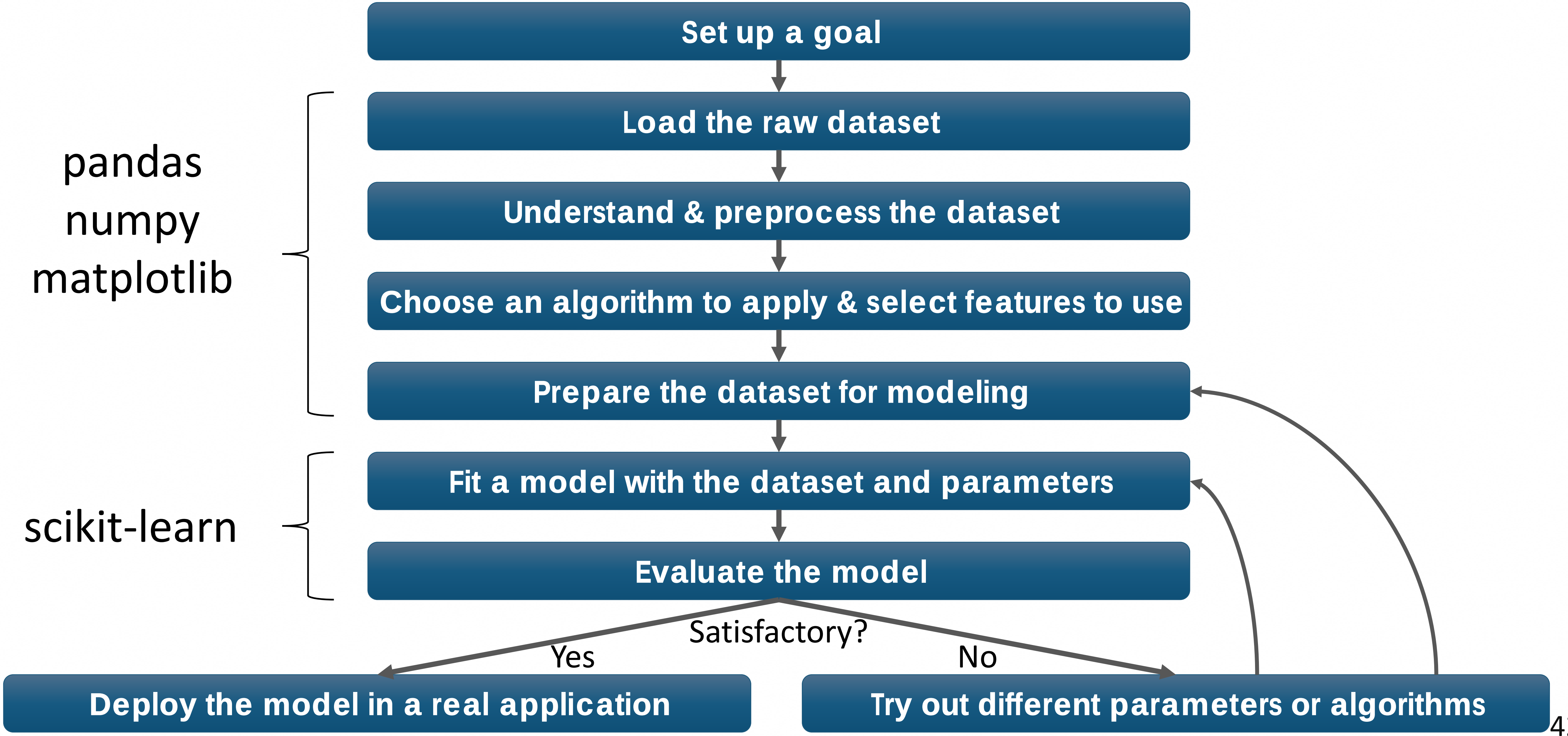
iPython is a Python command shell for interactive computing

Jupyter Notebook (former iPython Notebook) is a web-based interactive data analytics environment that supports iPython

Data Analytics Settings for This Course

Component	Name
Python Version	Python 3 (vs. Python 2)
Data Analytics Environment	Jupyter Notebook (vs. Wing IDE, PyCharm, PyDev, Spyder)
Data Analytics Software Toolkit	Anaconda (vs. Enthought Canopy)
Data Analytics Libraries	pandas & numpy for data manipulation matplotlib for visualization scikit-learn for machine learning

Data Analytics Libraries for This Course

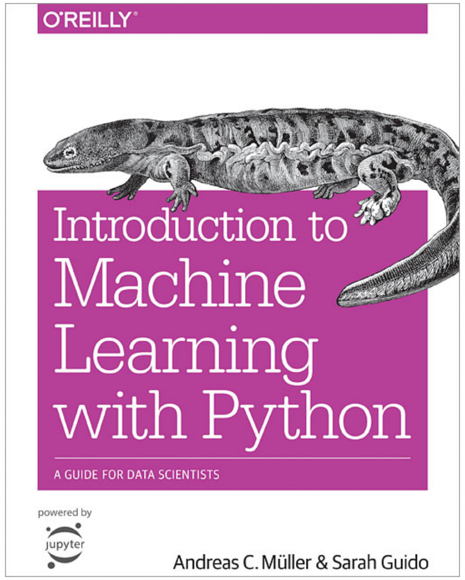


References

Introduction to Machine Learning with Python
(http://shop.oreilly.com/product/0636920030515.do)

Machine Learning course on Coursera
by Professor Andrew Ng at Stanford University
(https://www.coursera.org/learn/machine-learning/)

Wikipedia (https://en.wikipedia.org/)



Quick Survey on Prior Experience

Programming

- I have experience with Python
- I have experience with programming, but not with Python
- I have no experience with programming

Future Training

~~Machine Learning With Python (3 hours) – March 20, 2018~~

Web Scraping with Python (2 hours) – April 25, 2018

Introduction to Python Data Analytics (3 hours) – May 24, 2018

https://hpc.uiowa.edu/events