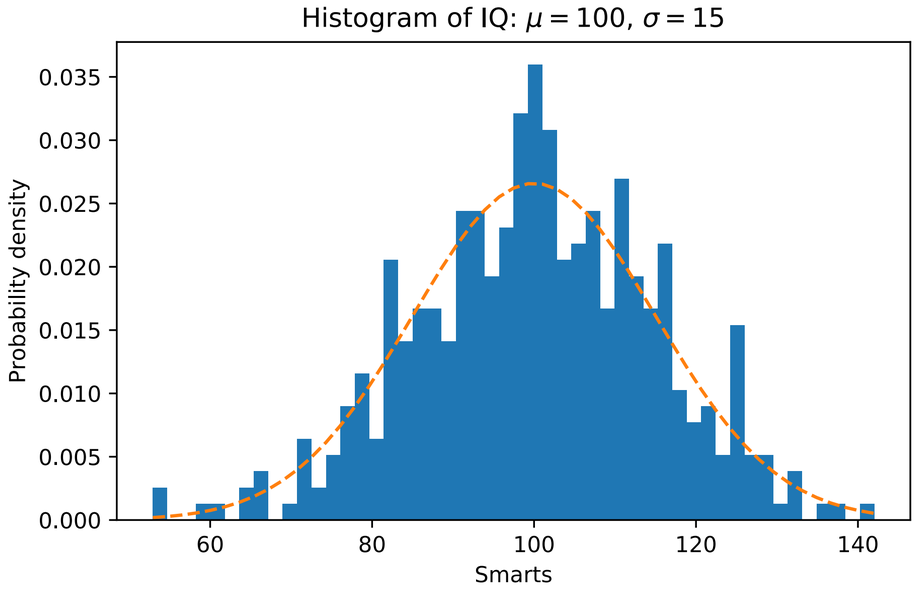
Programming Principles in Python (CSCI 503/490)

Machine Learning

Dr. David Koop

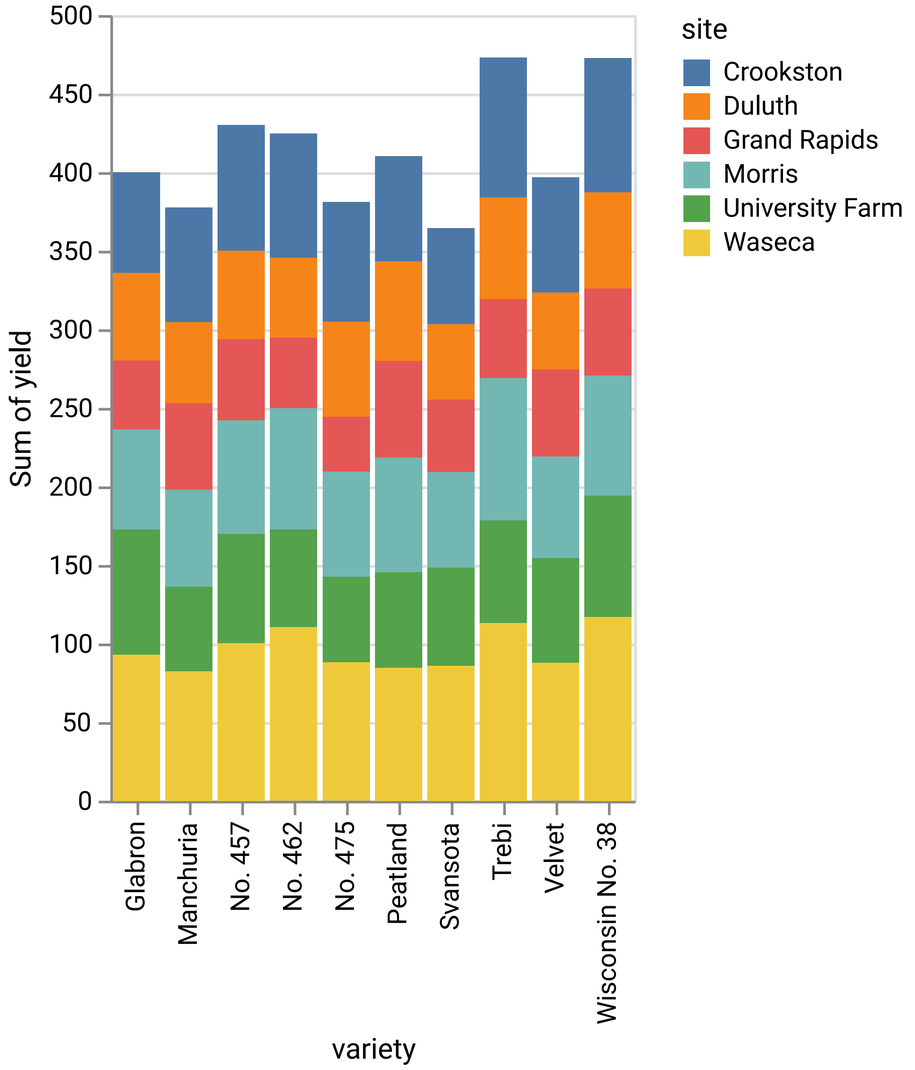
matplotlib

- Strengths:
  - Designed like Matlab
  - Many rendering backends
  - Can reproduce almost any plot
  - Proven, well-tested
- Weaknesses:
  - API is imperative
  - Not originally designed for the web
  - Dated styles

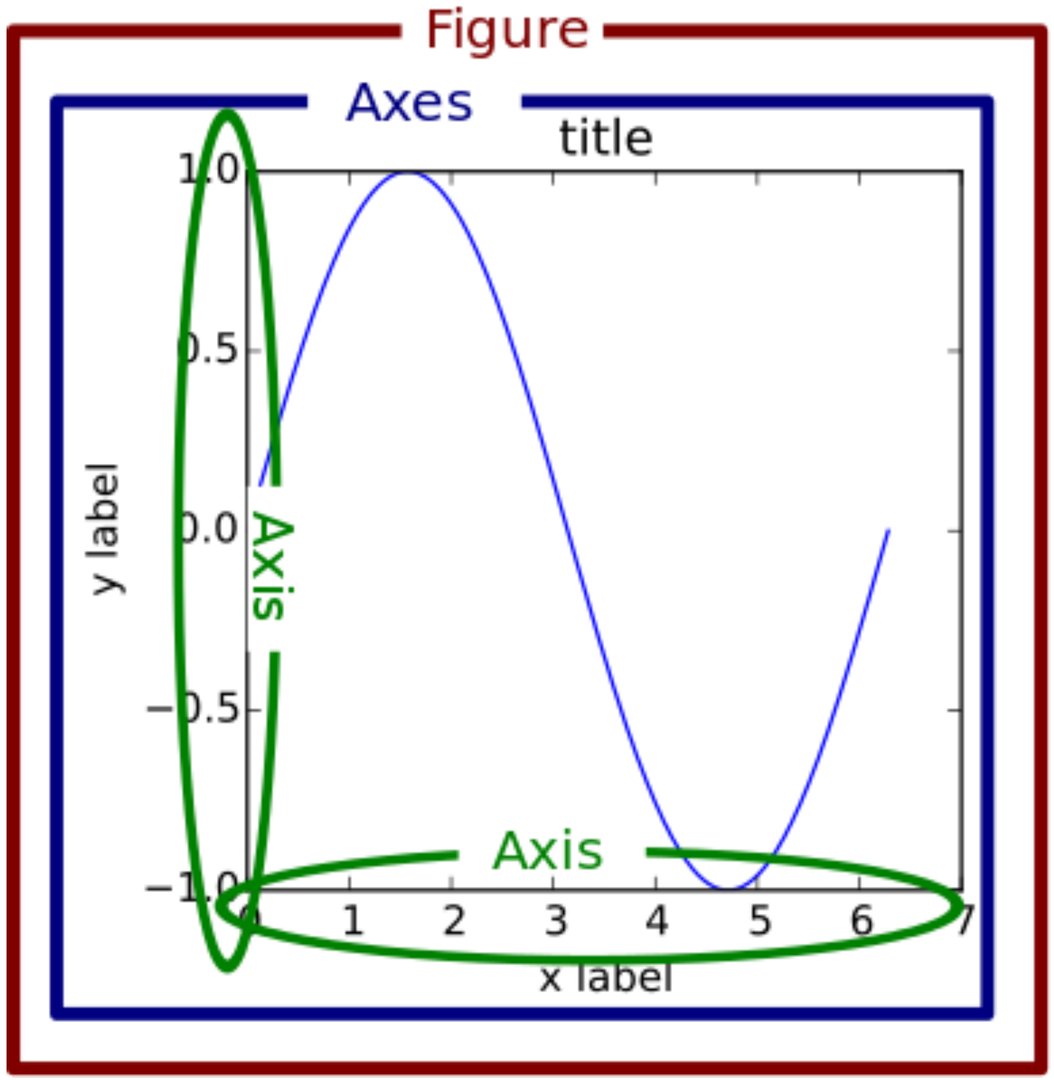


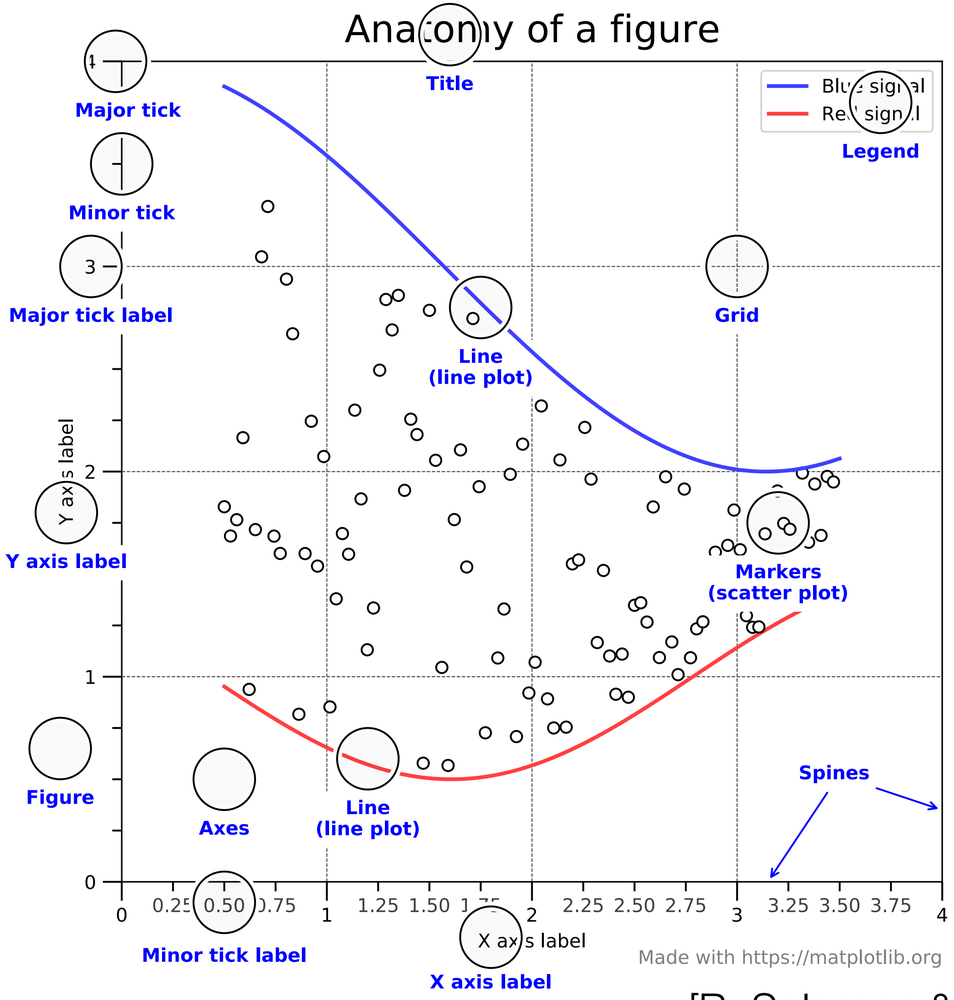
Vega-Altair

- Declarative Visualization
  - Specify **what** instead of how
  - Separate specification from execution
- Based on Vega-Lite which is browser-based
- Strengths:
  - Declarative visualization
  - Web technologies
- Drawbacks:
  - Moving data between Python and JS
  - Sometimes longer specifications



Anatomy of a Figure





[B. Solomon & matplotlib]

Visual Marks

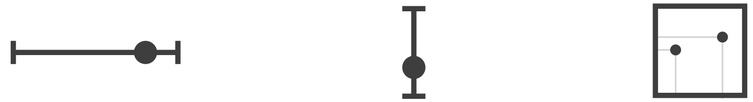
- **Marks** are the basic graphical elements in a visualization
- Marks classified by dimensionality:

Points **Lines** **Areas**

- Also can have surfaces, volumes
- Think of marks as a mathematical definition, or if familiar with tools like Adobe Illustrator or Inkscape, the path & point definitions
- Altair: area, bar, circle, geoshape, image, line, point, rect, rule, square, text, tick
  - Also compound marks: boxplot, errorband, errorbar

Encode via Visual Channels

- **Position**
  - Horizontal
  - Vertical
  - Both



- **Color**

- **Shape**

- **Tilt**

- **Size**
  - Length
  - Area
  - Volume

[Munzner (ill. Maguire), 2014]

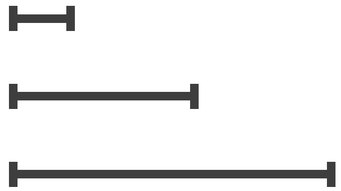
Data Attributes and Altair Types

➔ Categorical

➔ Ordered

➔ *Ordinal*

➔ *Quantitative*



[Munzner (ill. Maguire), 2014]

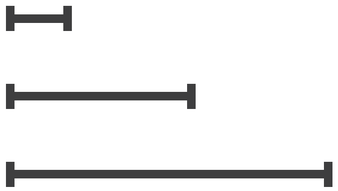
Data Attributes and Altair Types

➔ Categorical

➔ Ordered

➔ *Ordinal*

➔ *Quantitative*



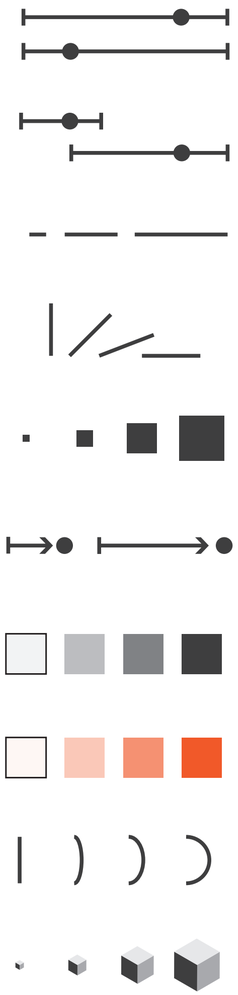
- Categorical data = Nominal (N)
- Ordinal data = Ordinal (O)
- Quantitative data = Quantitative (Q)
- Temporal data = Temporal (T)

[Munzner (ill. Maguire), 2014]

Different Channels for Different Attribute Types

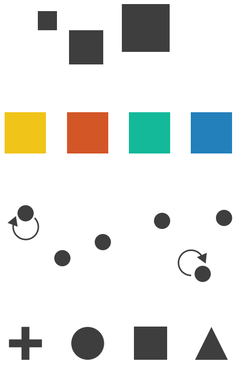
Magnitude Channels: Ordered Attributes

- Position on common scale
- Position on unaligned scale
- Length (1D size)
- Tilt/angle
- Area (2D size)
- Depth (3D position)
- Color luminance
- Color saturation
- Curvature
- Volume (3D size)



Identity Channels: Categorical Attributes

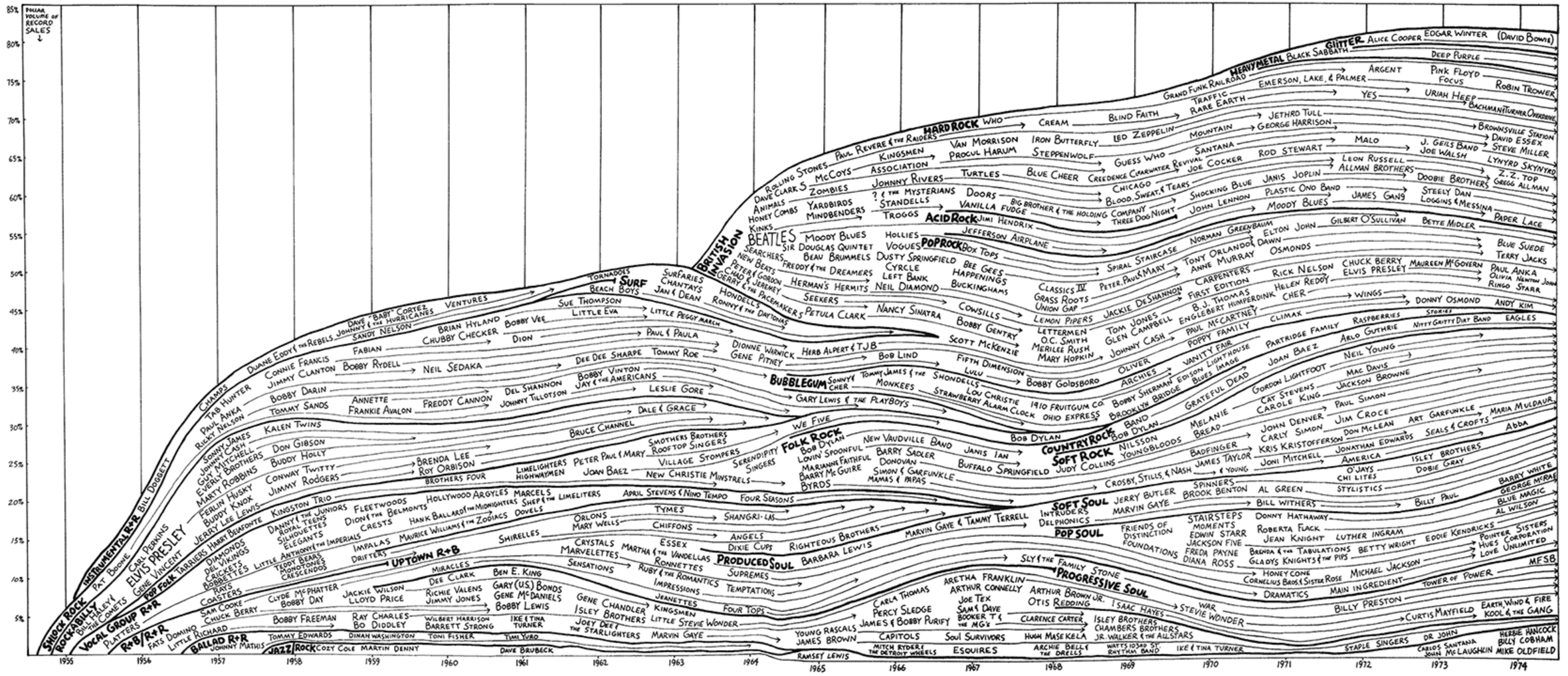
- Spatial region
- Color hue
- Motion
- Shape



Altair will use its rules to pick whether to use color hue or saturation based on the type

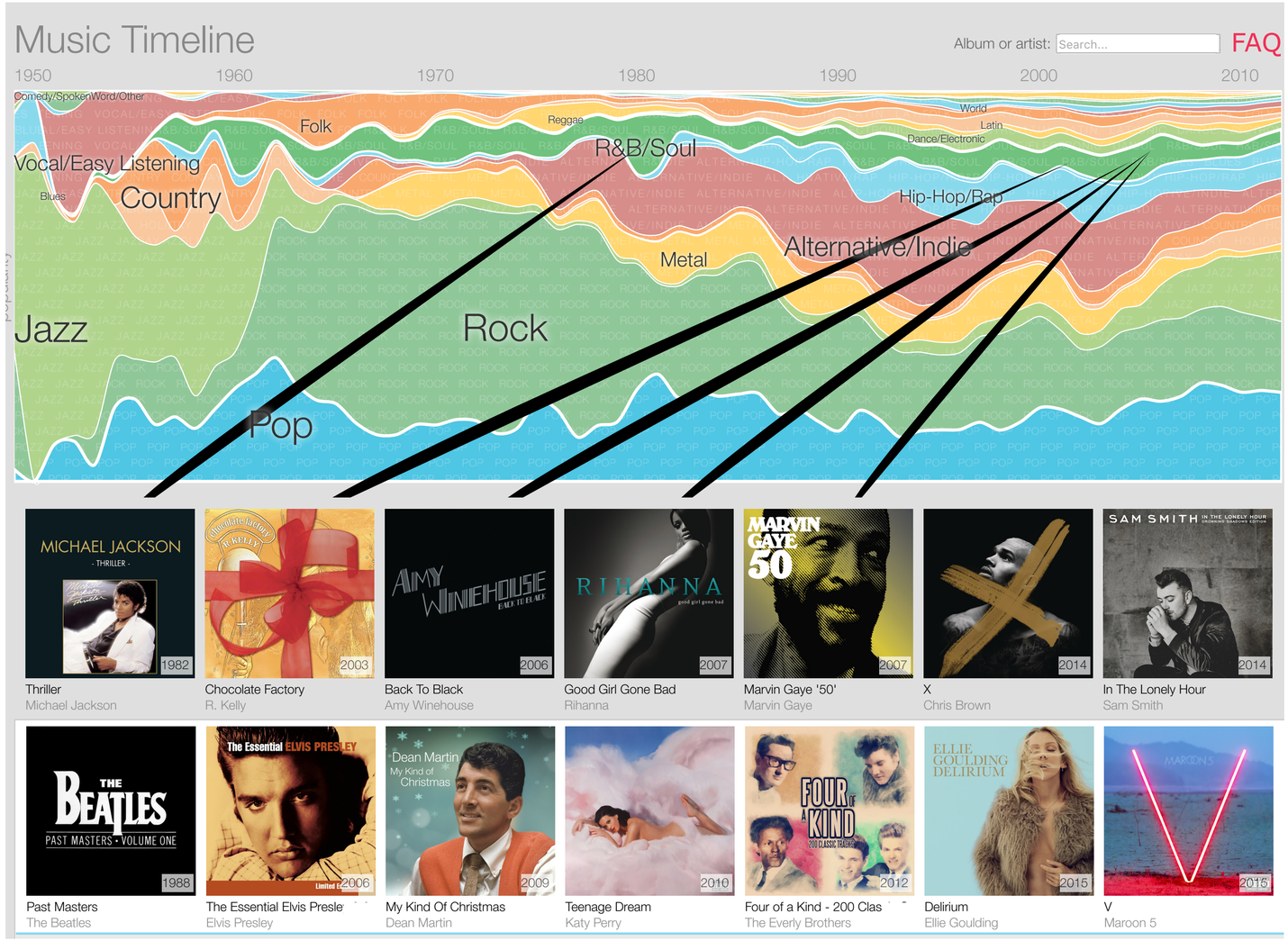
[Munzner (ill. Maguire), 2014]

Visualization



[Rock 'N' Roll is Here to Pay, R. Garofalo, 1977 (via Tufte)]

Also Visualization, but with Interaction



[Music Timeline, Google Research (no working version)]

Selection

- Selection is often used to initiate other changes
- User needs to select something to drive the next change
- What can be a selection target?
  - Items, links, attributes, (views)
- How?
  - mouse click, mouse hover, touch
  - keyboard modifiers, right/left mouse click, force
- Selection modes:
  - Single, multiple
  - Contiguous?

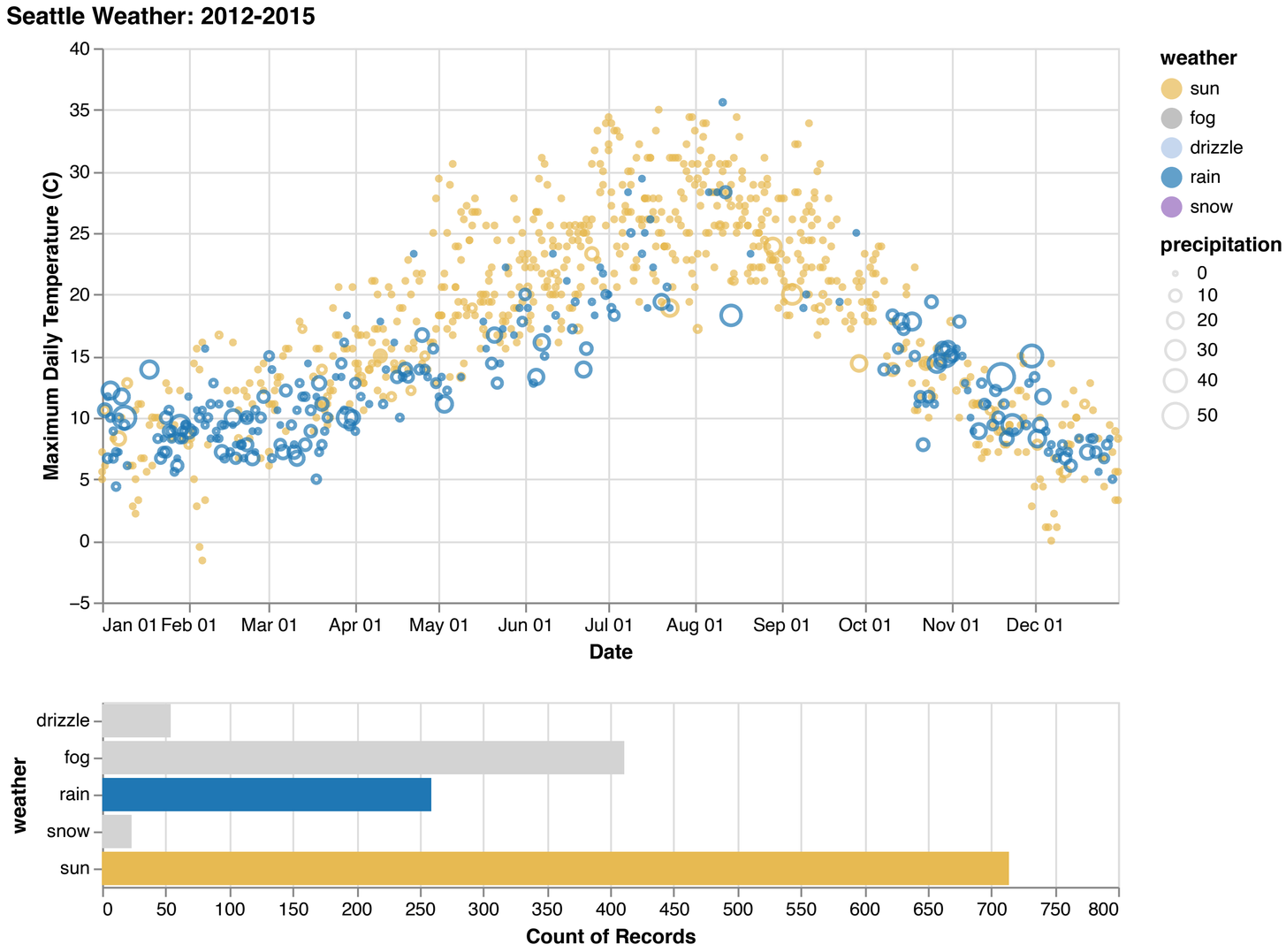
Highlighting

- Selection is the user action
- Feedback is important!
- How? Change selected item's visual encoding
  - Change color: want to achieve visual popout
  - Add outline mark: allows original color to be preserved
  - Change size (line width)
  - Add motion: marching ants

Highlighting

- Selection is the user action
- Feedback is important!
- How? Change selected item's visual encoding
  - Change color: want to achieve visual popout
  - Add outline mark: allows original color to be preserved
  - Change size (line width)
  - Add motion: marching ants

Interactive Visualization: Rain vs. Sun



Assignment 8

- Data and Visualization
- Work with polars or pandas
- Must use seaborn/matplotlib and altair (specified for problems)

Final Exam

- In-Person (CSCI 503-1 & 490): Wednesday, May 6, **8:00**-9:50am in PM 203
- Online (CSCI 503-2): Wednesday, May 6
- **More comprehensive** than Test 2
- Expect questions from topics covered on Test 1 and 2
- Expect questions from the last few weeks of class (concurrency, structure pattern matching, data, visualization, machine learning)
- Similar format

Machine Learning in Python

Tasks Machine Learning can Help With

- Identifying the zip code from handwritten digits on an envelope

- Detecting fraudulent activity in credit card transactions
- Identifying topics in a set of blog posts
- Grouping customers with similar preferences

[A. Müller & S. Guido, Introduction to Machine Learning with Python, J. Steppan (MNIST image)]

When to Use Machine Learning?

- ML is used when:
  - Human expertise does not exist (navigating on Mars)
  - Humans can’t explain their expertise (speech recognition)
  - Models must be customized (personalized medicine)
  - Models are based on huge amounts of data (genomics)
- ML isn’t always useful:
  - Calculating payroll…

[E. Alpaydin via E. Eaton]

Questions when building a machine learning solution

- What question(s) am I trying to answer? Do I think the data collected can answer that question?
- What is the best way to phrase my question(s) as a machine learning problem?
- Have I collected enough data to represent the problem I want to solve?
- What features of the data did I extract, and will these enable the right predictions?
- How will I measure success in my application?

[A. Müller & S. Guido]

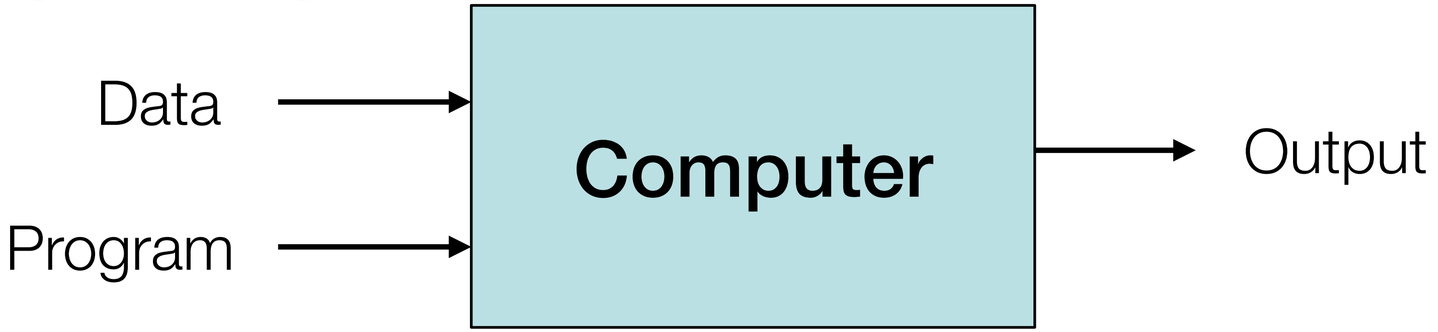
Machine Learning Workflow Overview

1. Should I use ML on this problem?
   - Is there a pattern to detect? Can I solve it analytically? Do I have data?
2. Gather and organize data.
   - Preprocessing, cleaning, visualizing.
3. Establishing a baseline.
4. Choosing a model, loss, regularization, …
5. Optimization (could be simple, could be a Phd…).
6. Hyperparameter search.
7. Analyze performance & mistakes, and iterate back to step 4 (or 2).

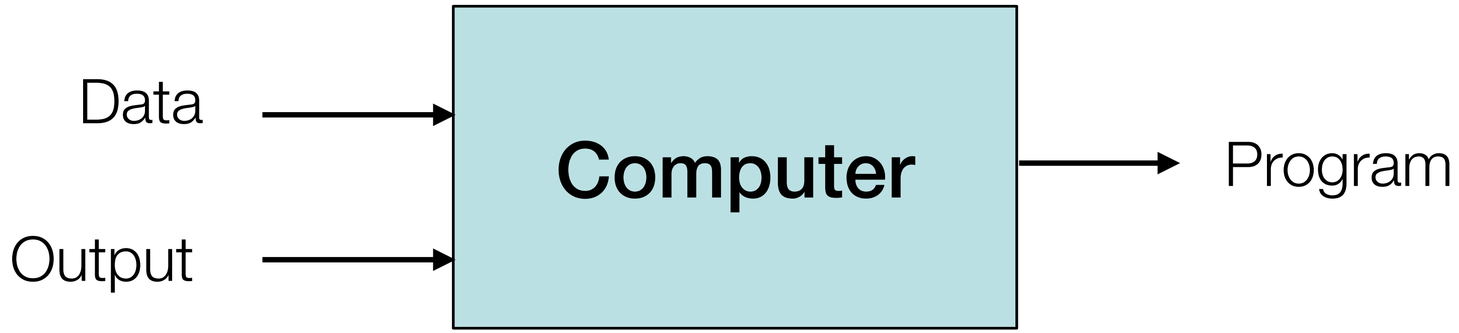
[R. Grosse et al.]

Machine Learning

- Traditional Programming



- Machine Learning



[P. Domingos]

Machine Learning

- Every machine learning algorithm has three components:
  - Representation
  - Evaluation
  - Optimization

Representation

- Decision trees
- Sets of rules / Logic programs
- Instances
- Graphical models (Bayes/Markov nets)
- Neural networks
- Support vector machines
- Model ensembles
- Etc.

[P. Domingos]

Evaluation

- Accuracy
- Precision and recall
- Squared error
- Likelihood
- Posterior probability
- Cost / Utility
- Margin
- Entropy
- K-L divergence
- Etc.

[P. Domingos]

Optimization

- Combinatorial optimization
  - E.g.: Greedy search
- Convex optimization
  - E.g.: Gradient descent
- Constrained optimization
  - E.g.: Linear programming

[P. Domingos]

Types of Learning

- Supervised (inductive) learning
  - Training data includes desired outputs
- Unsupervised learning
  - Training data does not include desired outputs
- Semi-supervised learning
  - Training data includes a few desired outputs
- Reinforcement learning
  - Rewards from sequence of actions

[P. Domingos]

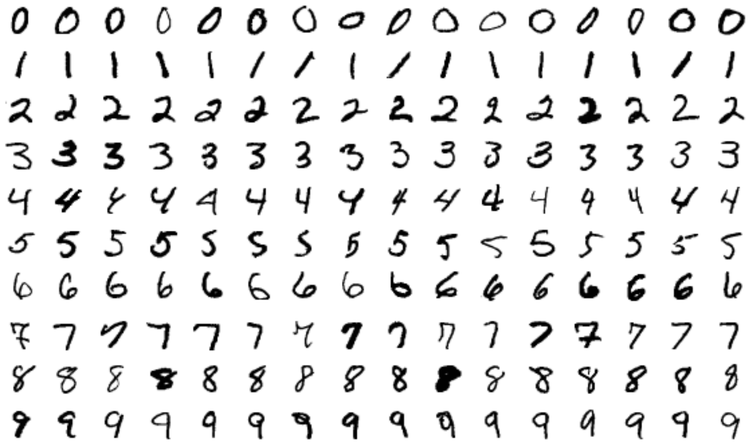
Areas of Machine Learning

- Supervised learning
  - Decision tree induction
  - Rule induction
  - Instance-based learning
  - Bayesian learning
  - Neural networks
  - Support vector machines
  - Model ensembles
  - Learning theory
- Unsupervised learning
  - Clustering
  - Dimensionality reduction

[P. Domingos]

Supervised & Unsupervised Tasks

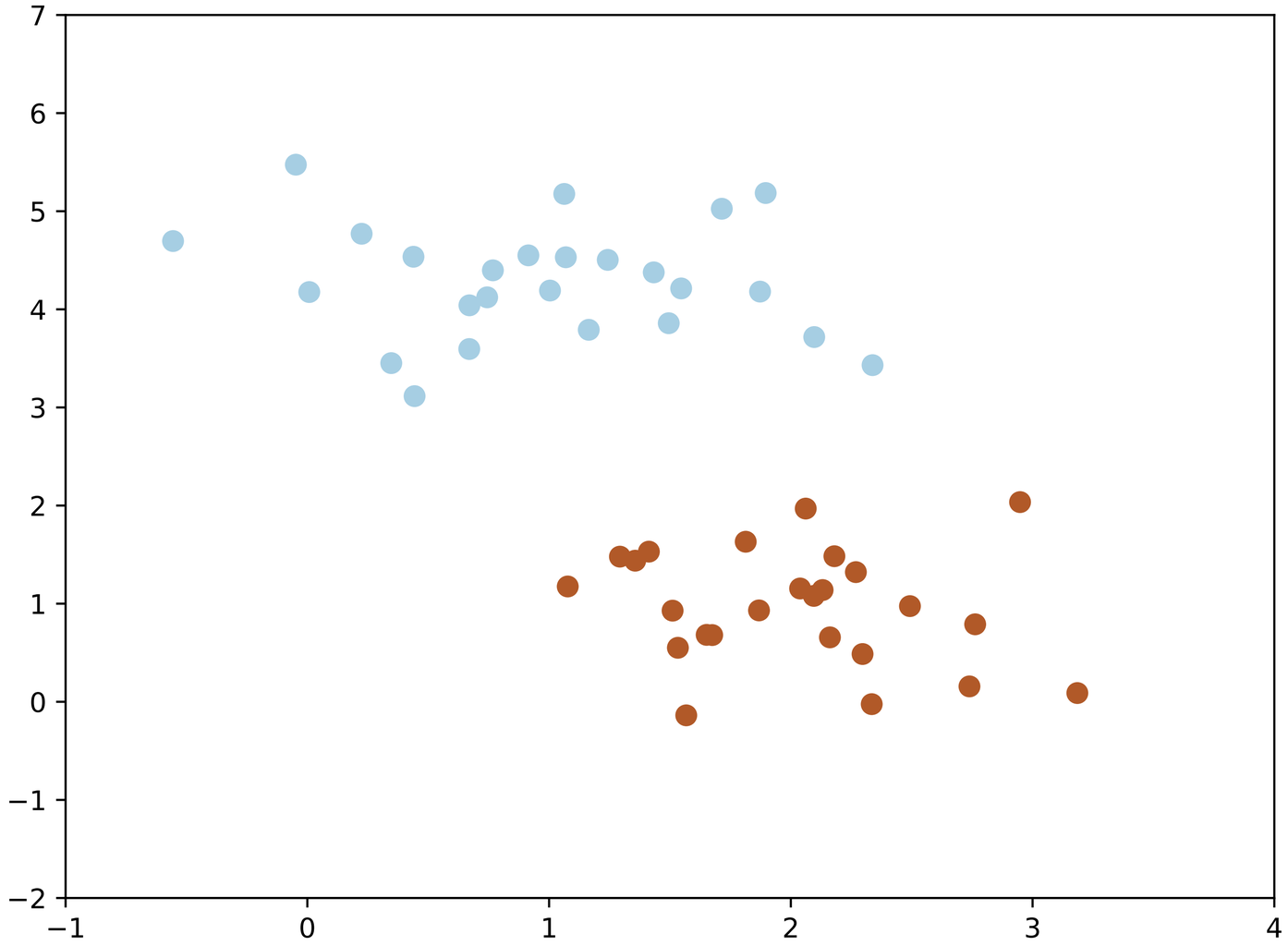
- Identifying the zip code from handwritten digits on an envelope (**supervised**)



- Detecting fraudulent activity in credit card transactions (**supervised**)
- Identifying topics in a set of blog posts (**unsupervised**)
- Grouping customers with similar preferences (**unsupervised**)

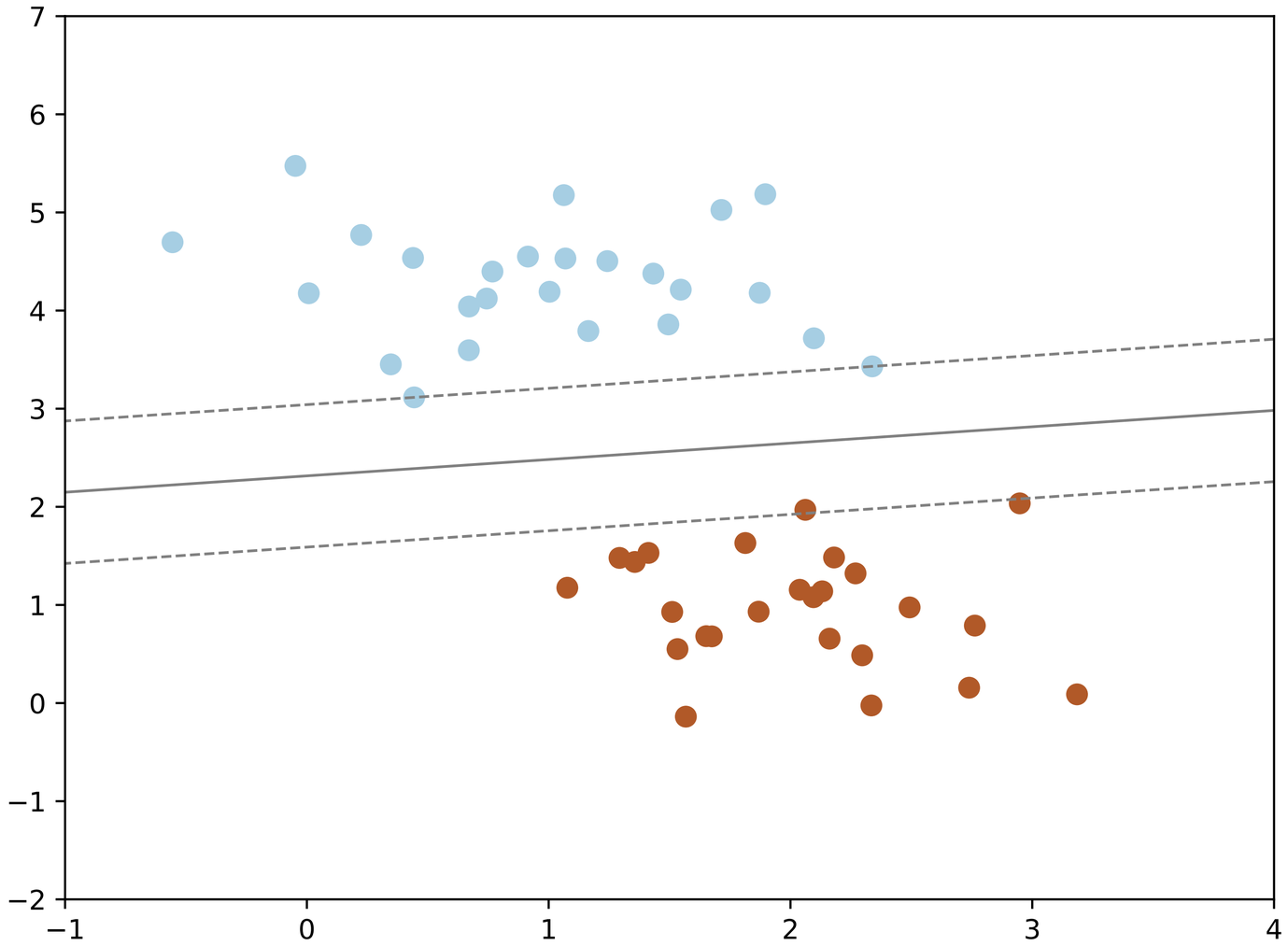
[A. Müller & S. Guido, Introduction to Machine Learning with Python, J. Steppan (MNIST image)]

Supervised Learning



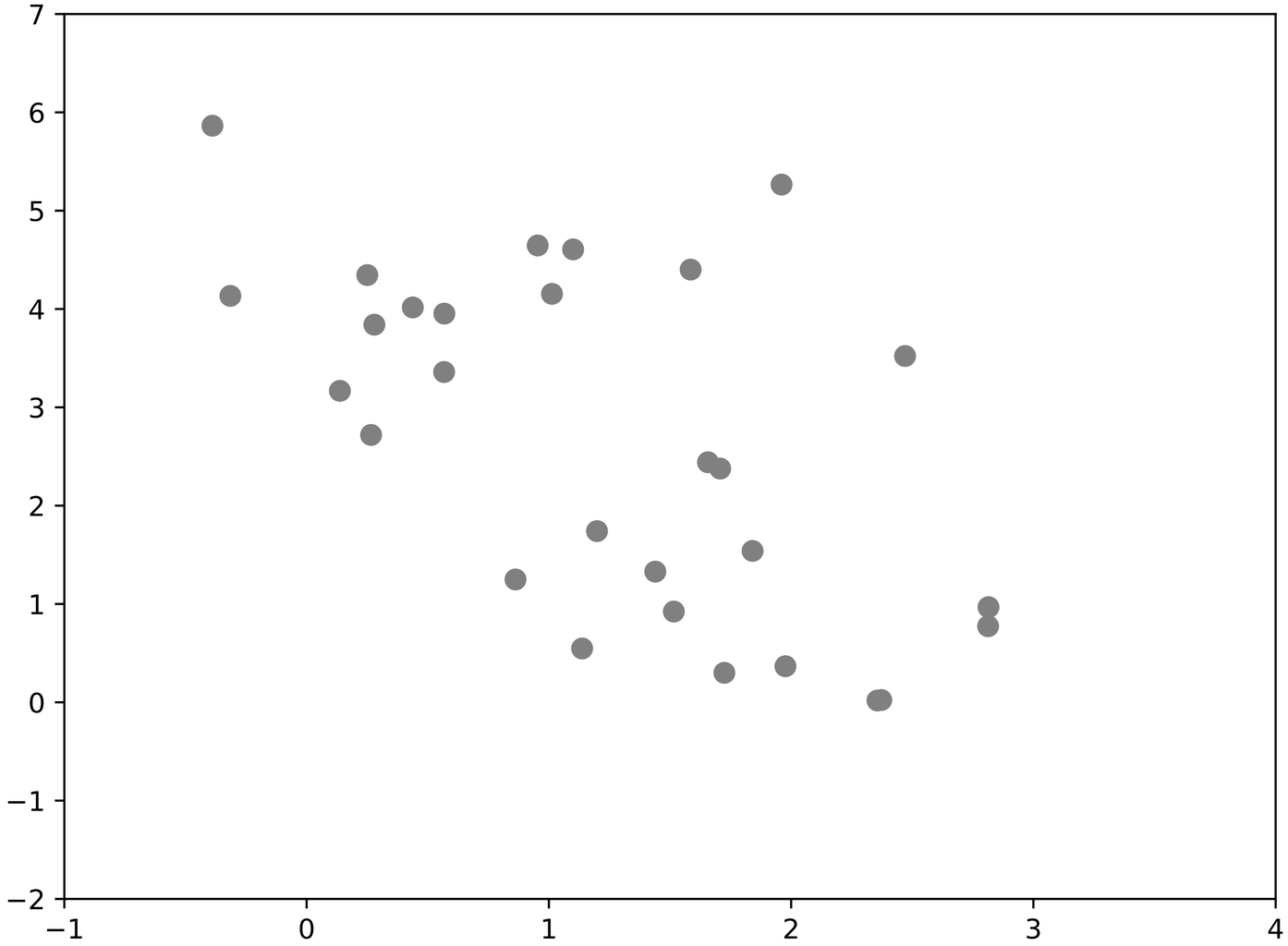
[J. VanderPlas]

Supervised Learning: Learned Algorithm (Fit)



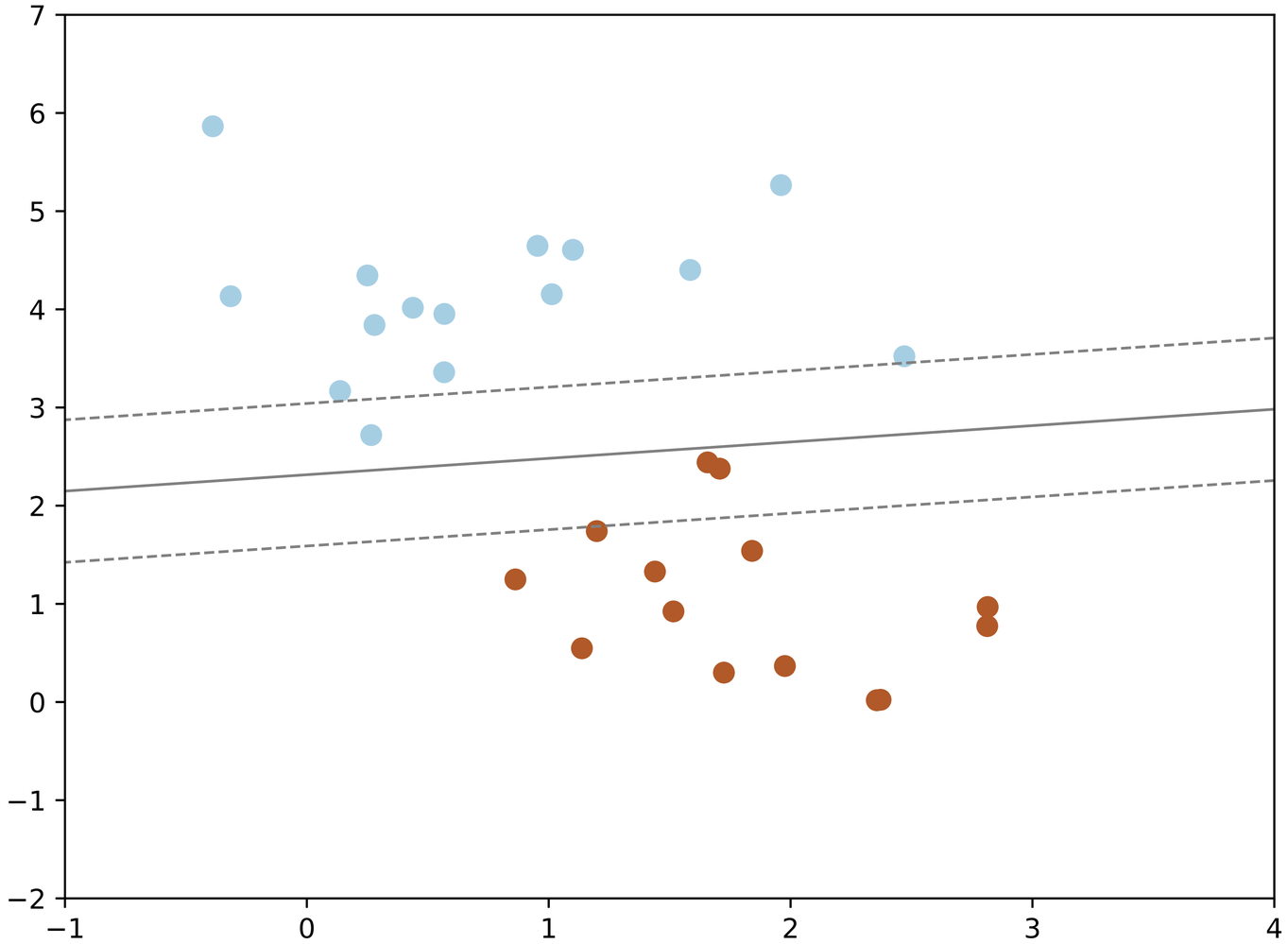
[J. VanderPlas]

Supervised Learning: Prediction



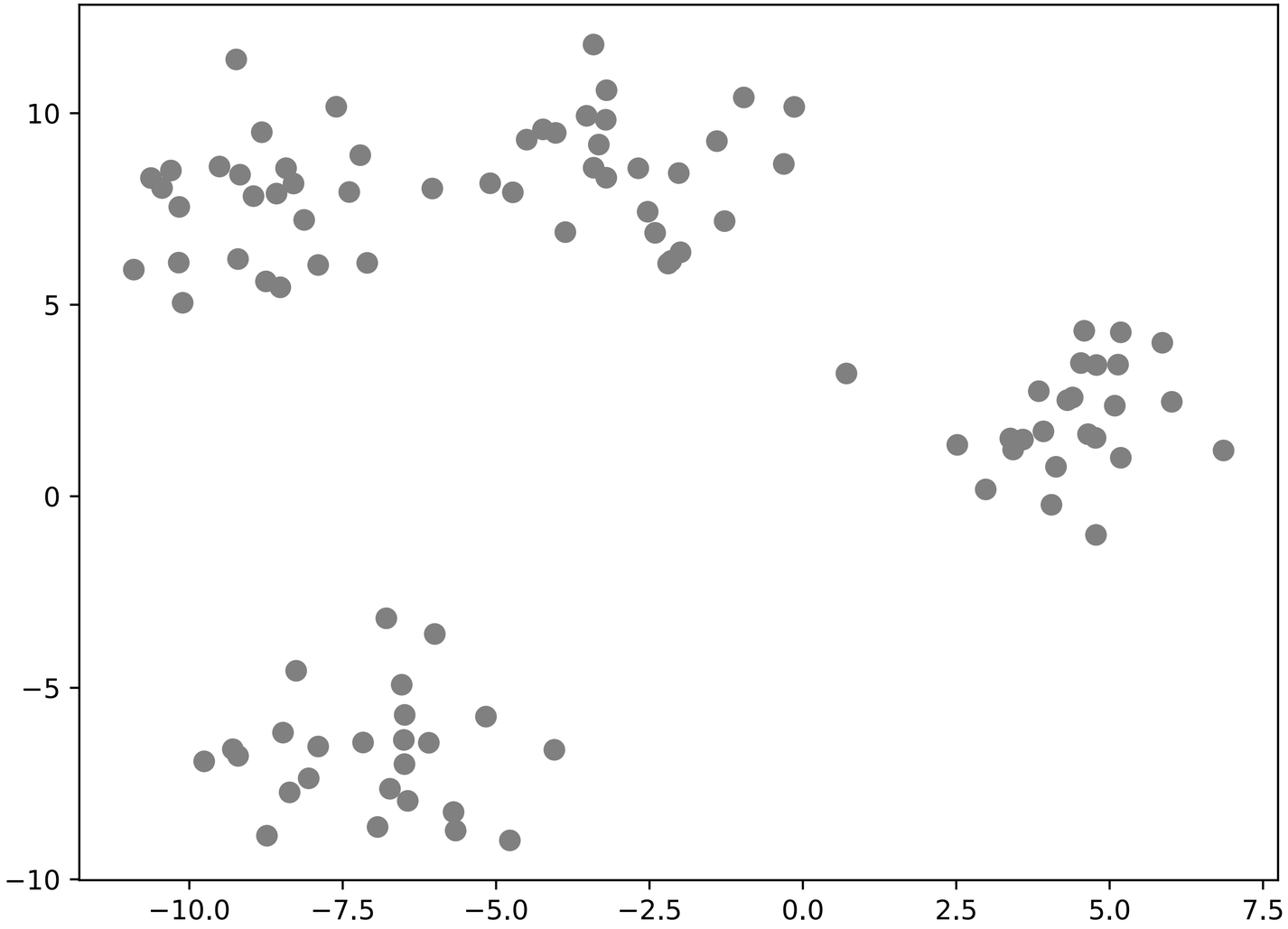
[J. VanderPlas]

Supervised Learning: Prediction



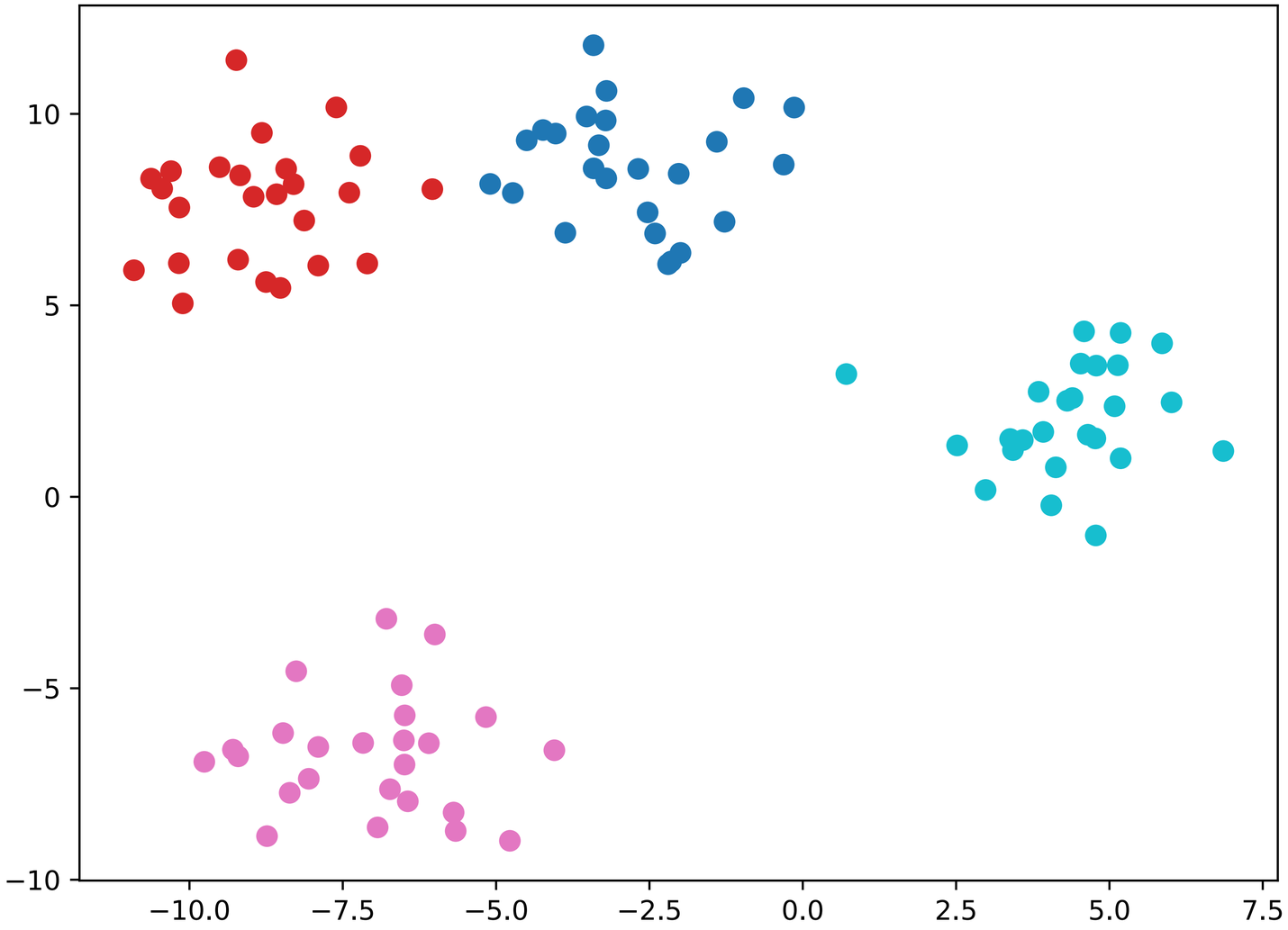
[J. VanderPlas]

Unsupervised Learning: Input



[J. VanderPlas]

Unsupervised Learning: Output



[J. VanderPlas]

Scikit-Learn

- Started as a Google Summer of Code project! (D. Cournapeau, 2007)
- Rewritten by scientists at INRIA (France) in 2010
- Written in Python using numpy, some optimizations using C (cython)
- The "gold standard" for machine learning in python

[scikit-learn]

scikit-learn Principles

- Consistency: all objects share consistent, documented interface
- Inspection: parameters and parameter values determined by learning algorithms are stored and exposed as public attributes
- Non-proliferation of classes: only learning algs are classes, not datasets or parameters; easier to combine with other libraries
- Composition: create and reuse building blocks
- Sensible defaults: user-defined parameters should have meaningful defaults

[L. Buitinck et al.]

scikit-learn entities

- Data: numpy matrices (also pandas series, data frames), process batches
- Estimators: all supervised & unsupervised algs implement **common** interface
  - estimator initialization does not do learning, only attaches parameters
  - `fit` does the learning, learned parameters exposed with trailing underscore
- Predictor: extends estimator with `predict` method
  - also provides `score` method to return value indicating prediction quality
- Transformer: help modify or filter data before learning
  - Preprocessing, feature selection, feature extraction, and dimensionality reduction vis `transform` method
  - Can combine `fit` and `transform` via `fit_transform`

[L. Buitinck et al.]

Penguin Example

scikit-learn Template

1. Choose model class
2. Instantiate model
3. Fit model to data
4. Predict on new data

```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(Xtrain, ytrain)
y_model = model.predict(Xtest)
```

5. (Check accuracy)

```
from sklearn.metrics import accuracy_score
accuracy_score(ytest, y_model)
```

Deep Learning

- Deep learning is tied to neural networks, attempting to mimic how human neurons work together
- Hierarchical with multiple layers
- Usually takes advantage of GPUs
- Frameworks:
  - pytorch
  - TensorFlow
  - keras
  - theano