私が愛用しているツールたち

- python, jupyter-lab, nbdev, poetry, fastai, pycaret, prophet, streamlit, gurobi, mypulp, miro, etc. -

久保幹雄（東京海洋大学）

Amazonで検索

久保 幹雄

✓ フォロー中

フォローすると、最新刊やおすすめ作品の情報を入手できます。

久保 幹雄 Mikio Kubo
専門は，サプライ・チェインならびに組合せ最適化，
早稲田大学理工学研究科卒，博士（工学）．早稲田大学助手，東京商船大学助教授，ポルト大学招聘教授などを歴任，現在 東京海洋大学教授．代表的な著書として，「離散構造とアルゴリズムIV」近代科学社，「巡回セールスマン問題への招待」朝倉書店，「組合せ最適化とアルゴリズム」共立出版，「ロジスティクス工学」朝倉書店，「実務家のためのサプライ・チェイン最適化入門」朝倉書店，「ロジスティクスの数理」共立出版，「メタヒューリスティックスの数理」共立出版，「サプライ・チェイン最適化ハンドブック」朝倉書店，「サプライ・チェイン最適化の新潮流-統一モデルからリスク管理・人道支援まで-」 朝倉書店などがある．

http://www.logopt.com/mikiokubo/
^ もっと少なく読む

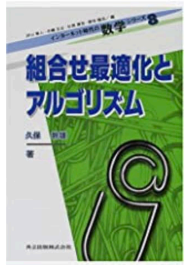
他のお客様が一緒に購入した商品：

Kindle版
￥3,168
32pt (1%)

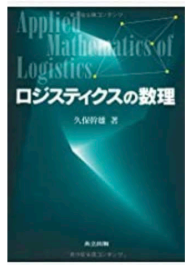
Kindle版
￥6,336
63pt (1%)

Kindle版
￥2,613
48pt (2%)

単行本
￥2,987



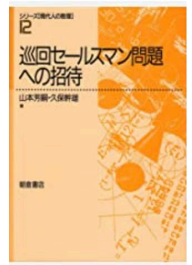
単行本
￥2,970
30pt (1%)



単行本
￥3,630
36pt (1%)

単行本
￥5,195

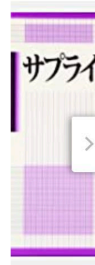
単行本（ソフトカバー）
￥3,520
35pt (1%)



単行本
￥4,514

単行本
￥500

大型本
￥28,600
286pt (1%)


単行本
￥18,700
187pt (1%)

久保 幹雄の作品

言語: 和書

すべてのフォーマット | Kindle 版 | ペーパーバック | 単行本（ソフトカバー） | 続きを見る

並べ替え: 出版年月が新しい順番

あたらしい数理最適化: Python言語とGurobiで解く 2016/09/06
久保 幹雄, ペドロソ ジョア・ペドロ, 村松 正和, レイス アブドル
Kindle版
￥3,168 ~~￥3,520~~
ポイント: **32pt** (1%)

単行本
￥3,520
ポイント: **35pt** (1%)
残り1点 ご注文はお早めに

★★★★☆ (3)

Python言語によるビジネスアナリティクス：実務家のための最適化・統計解析・機械学習 2016/09/01
久保 幹雄, 小林 和博, 斉藤 努, 並木 誠, 橋本 英樹
Kindle版
￥6,336 ~~￥7,040~~
ポイント: **63pt** (1%)

単行本
￥7,040
ポイント: **70pt** (1%)
残り2点（入荷予定あり）

★★★½☆ (4)

メニュー

開発環境

python, jupyter-lab, nbdev, poetry

機械学習

fastai, pycaret, prophet

Webアプリ作成とデプロイ

streamlit

数理最適化

gurobi, mypulp

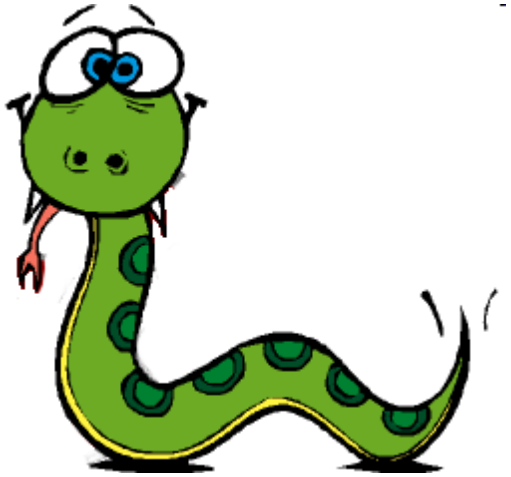
おまけ（コロナ禍でのSPRINT, 研究メモ，翻訳，研究連絡）

miro, notion, deepl, slack

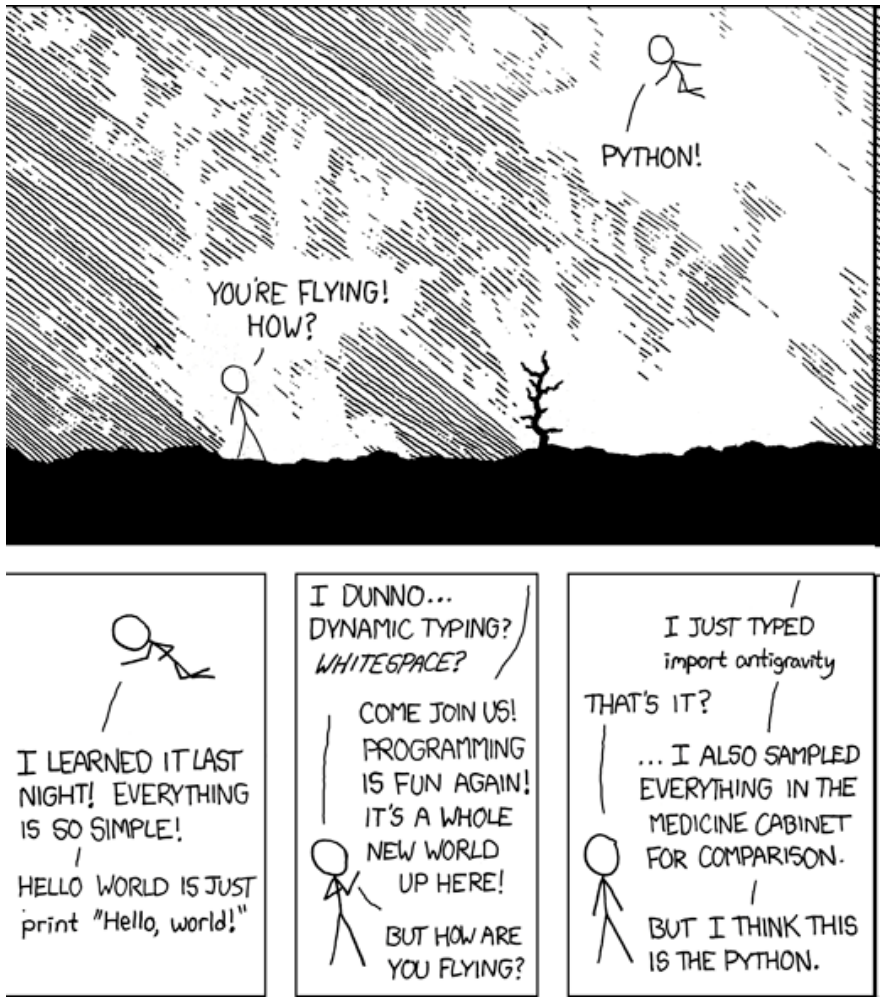
なぜ今Pythonか？

Pythonをお薦めする18の理由

久保 幹雄
東京海洋大学



Why Python (1) ?

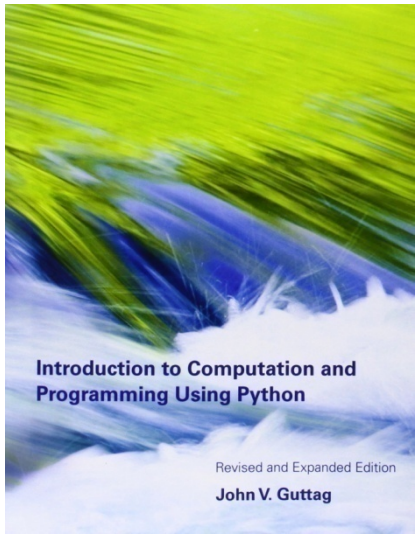


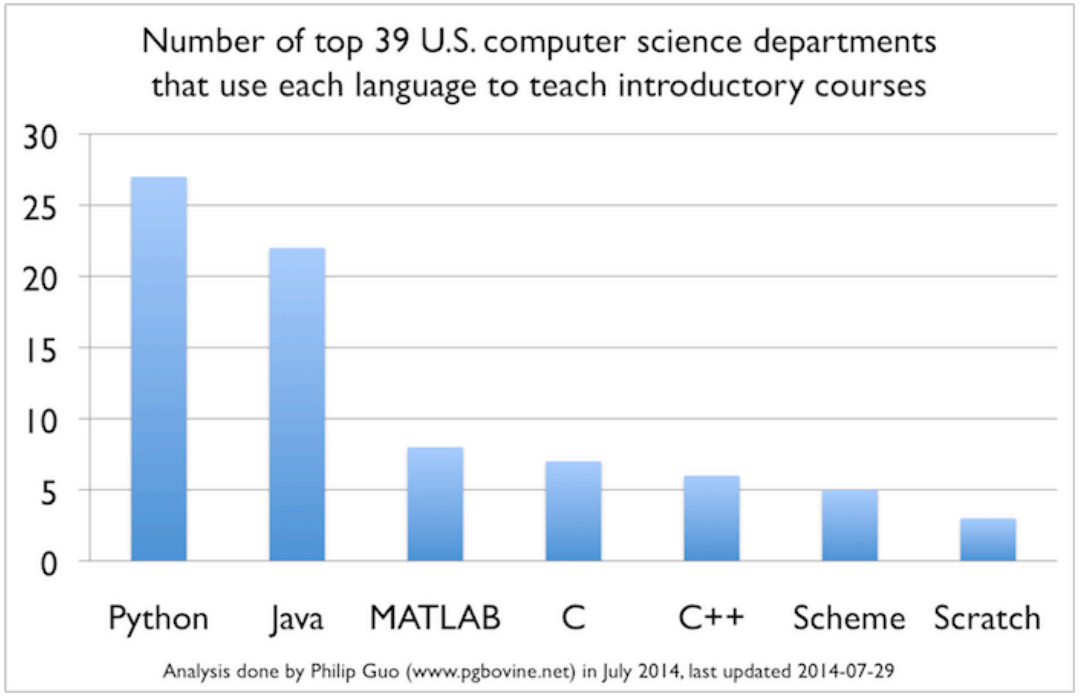
http://xkcd.com/353/

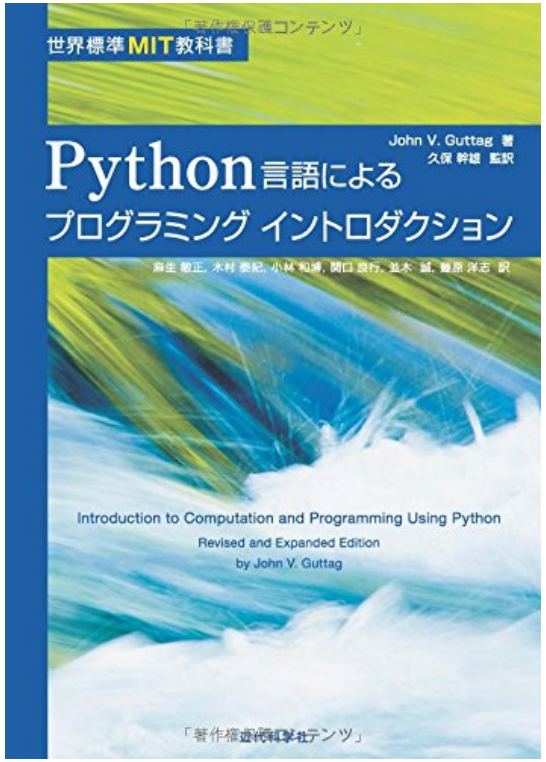
- モジュールをimportすれば何でもできる!
  - 最適化
  - データ解析
  - 統計
- 飛ぶこともできる !?
  import antigravity
- Programming is fun again!

Why Python (2) ?

- 米国計算機科学科で採用数No.1
- もちろんMITも！（Introduction to Computer Science and Programming Using Python by John Guttag）



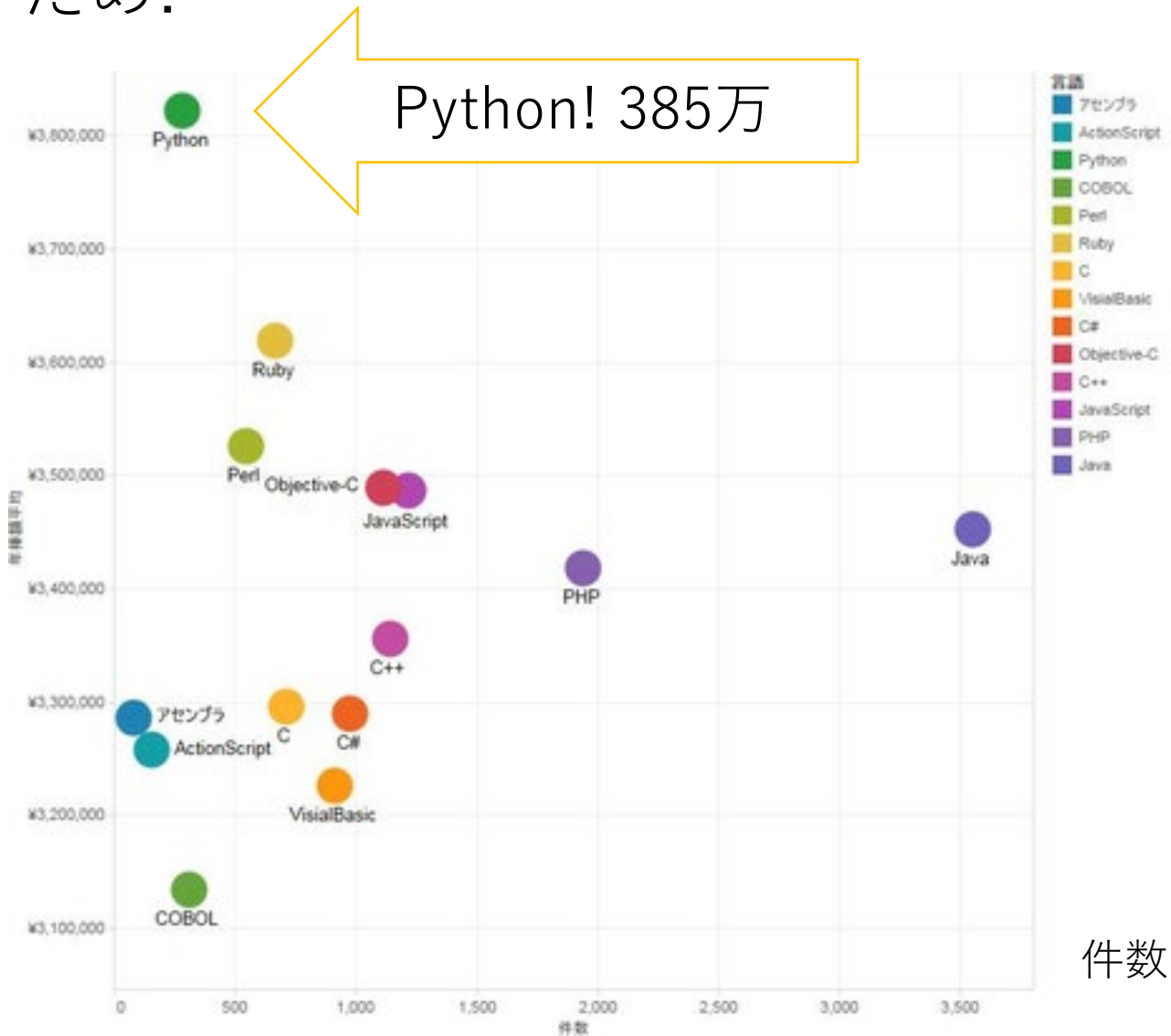




Why Python (3) ?

- お金を稼ぐため!

プログラマの平均
初任給
（日本）



出典：マイナビ

件数

Why Python (4)?

- もっとお金を稼ぐため!

Top 10 Programming Languages to Learn in 2014

No 1. Python
平均収入: $93,000
求人数: 24,533
Top Employers: Amazon, Dell, Google, eBay & Yahoo, Instagram, NASA, Yahoo
Example Sites: Google.com, Yahoo Maps, Reddit.com, Dropbox.com, Disqus.com

No 2. Java, No 3. Ruby, No. 4 C++, No5. JavaScript

No 6. C#, No. 7 PHP, No8. Perl ….

Why Python (5)?

- もっと，もっと，お金を稼ぐため!

Pythonはデータサイエンティストの必需品

Help Wanted: Black Belts in Data

by Rodrigo Orihuela and Dina Bass

Bloomberg Businessweek

Starting salaries for data scientists have gone north of $200,000

McKinsey projects that by 2018 demand for data scientists may be as much as 60 percent greater than the supply

http://www.bloomberg.com/news/articles/2015-06-04/help-wanted-black-belts-in-

Why Python (6)?

- キーワード（覚えるべき予約語）が
  33個 (Pyhon 3.4) と圧倒的に少ない（Python3.9だと36個）.

False	None	True	and	as
assert	break	class	continue	def
del	elif	else	except	finally
for	from	global	if	import
in	is	lambda	nonlocal	not
or	pass	raise	return	try
while	with	yield		

Why Python (7)?

字下げの強要で，誰でも読みやすいプログラム

```
if (x > 1) { y=x+1;
  z=x+y; }  else { y=0; z=0; }
```

（行儀の悪い）C++

```
if x > 1:
        y=x+1
        z=x+y
else:
        y=z=0
```

Pythonだと誰でも

この高さ（インデント）がそろっていないとエラーする！

Why Python (8)?

- 短時間で開発可能
  （行数が短く，モジュール豊富）

“Hello, world!”と出力するプログラム

C++版（覚える必要なし！）

```
#include <iostream>
    int main() {
       std::cout << "Hello, world!" <<
std::endl;
      return 0;
    }
```

Python 3 版

```
print （ “Hello, world!”)
```

python 2版

```
print “Hello, world”
```

Why Python (9)?

- 変数の宣言必要なし

```
Option Explicit

Sub Sample()
    Dim 基準値, N, Message
    基準値 = 100
    N = 基準値 * 2
    Message = "答は" & N & "です"
    MsgBox Mesaage
End Sub
```

Why Python (10)?

- インタープリタ（コンパイルする必要なし）

巷の声

コンパイルを
待つのは辛い

- **思考の中断**
- **待ち時間にほかのことを始める**

C#なんかで開発する場合だと、
細かい修正をして動作確認を繰り返す場合、
「F5押す→コンパイル待ち」となるので
そこらへんはムカついてくる。

コンパイルが遅すぎる

- 良いマシンを…（震え声）
- sbt の ~;compile とか ~;test でごまかす
- コンパイルが遅くなる言語機能を避ける…
- 何でもかんでもコンパイルしない（個人の意見です）

Why Python (11)?

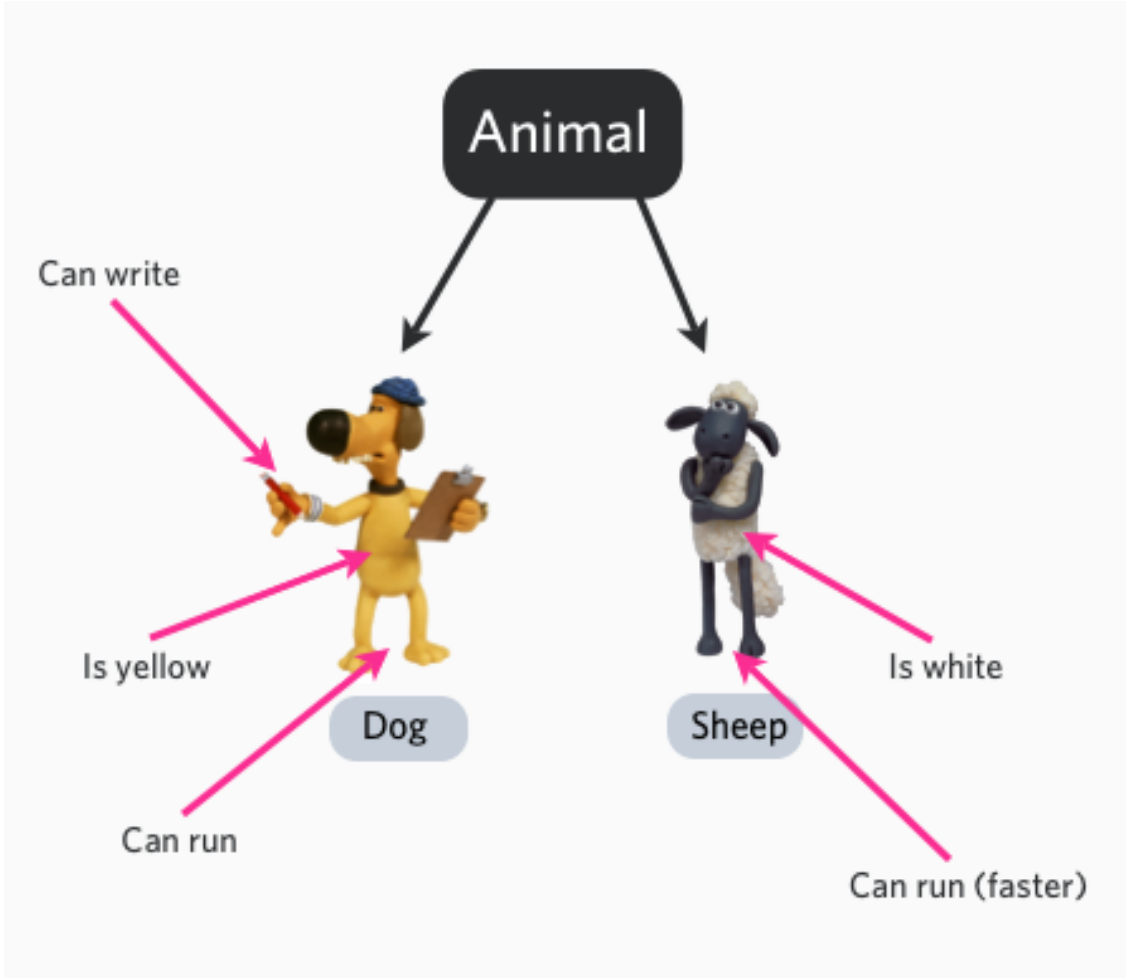
- メモリ管理も必要なし
  （自動ガーベッジ・コレクション）

Why Python (12)?

- 多くのプラットフォームで動作
  （Windows, Mac, Linux,　iphone, ... ）

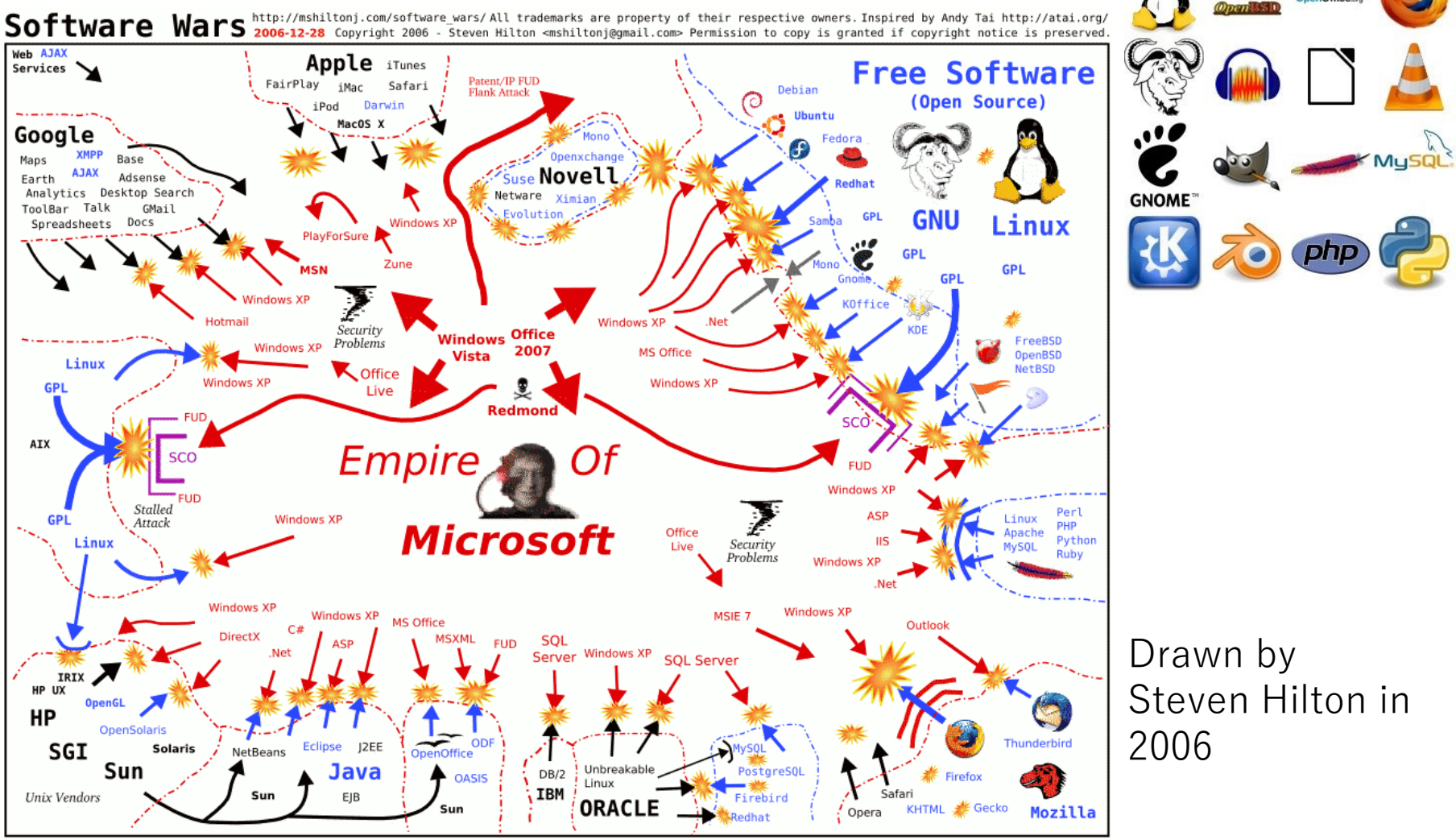
Why Python (13)?

- オブジェクト指向（すべてがオブジェクト）



Why Python (14)?

- しかもフリーソフト



Drawn by Steven Hilton in 2006

Why Python (15)?

- インストールが簡単
  Python IDLE付き (Windows, Mac, Linux)
  Batteries Included!

Why Python (16)?

- 追加モジュールのインストールも簡単

Anaconda

https://store.continuum.io/cshop/anaconda/

無料版で十分
100以上の便利なモジュールを含む
Mac, Linux, Windowsをサポート

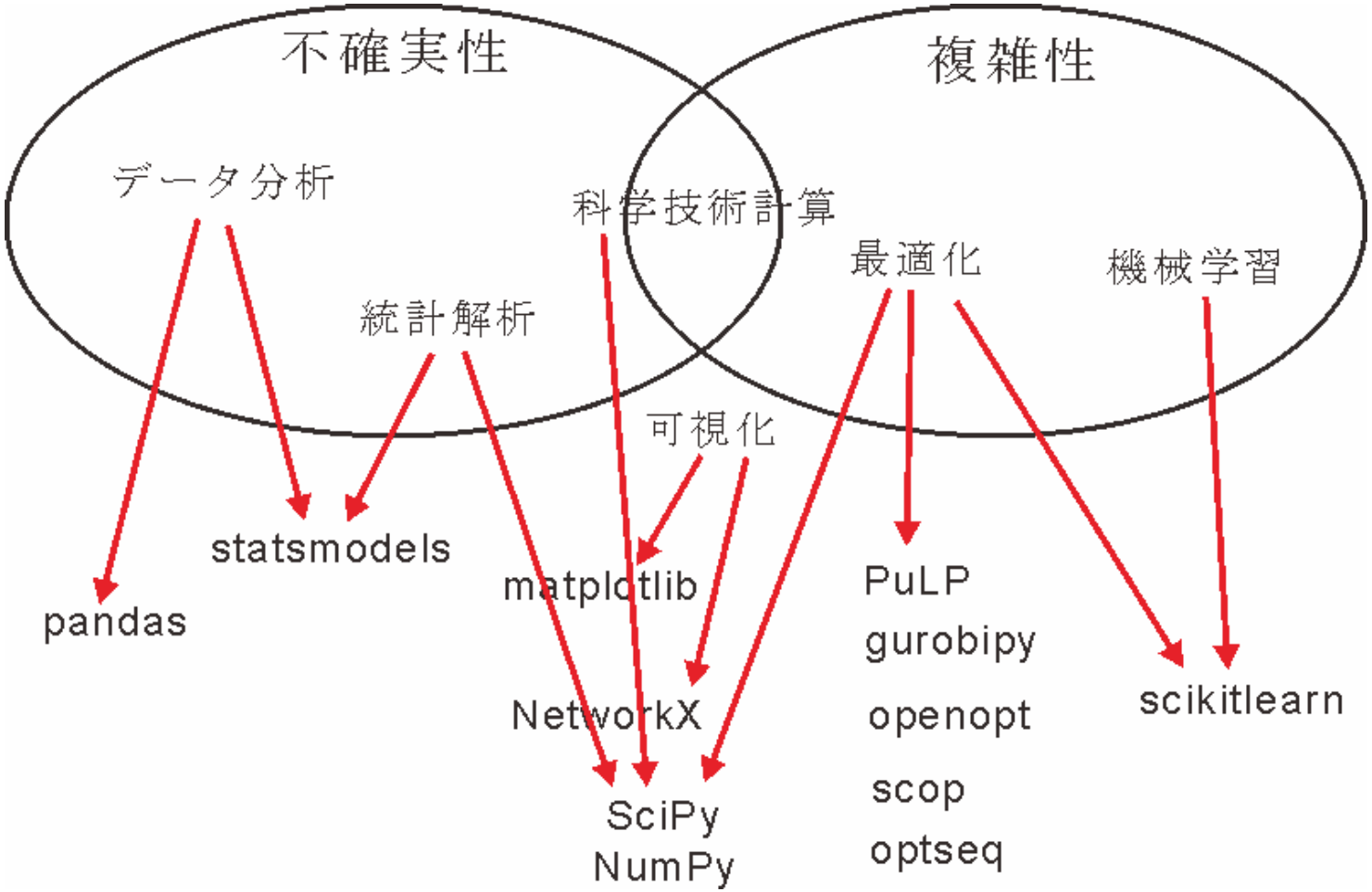
最近ではPoetryで仮想環境がおすすめ

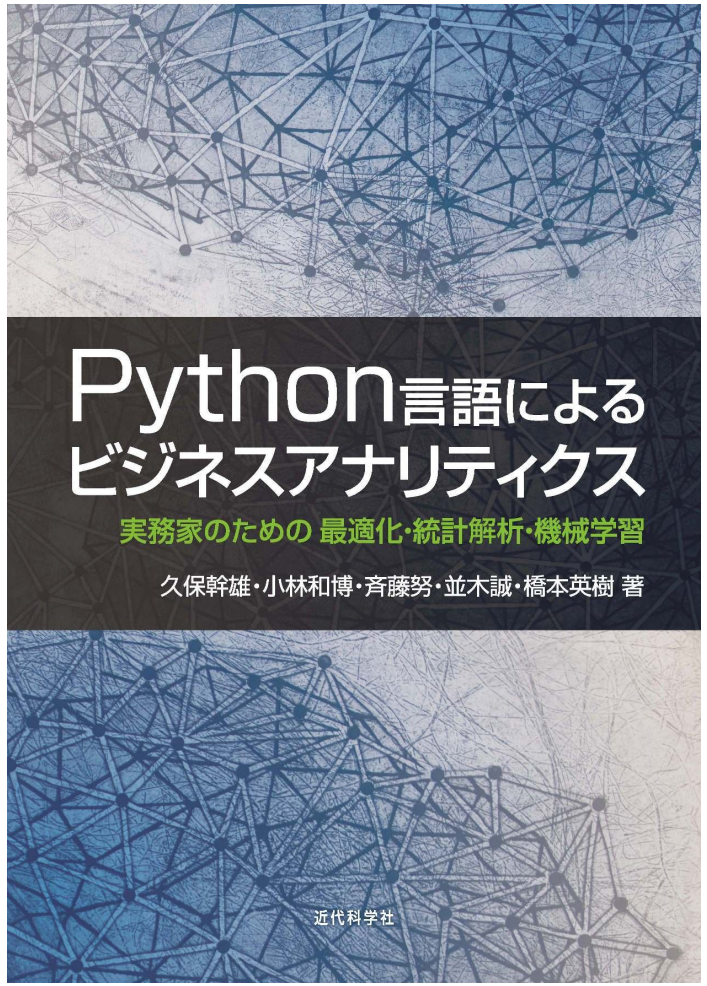
Why Python (17)?

- CやFORTRANとの接続が容易
- 過去のプログラム遺産を再利用するための糊の役目
- 高速化が必要な部分を C,FORTRAN で，インターフェイスは Pythonで

Why Python (18)?

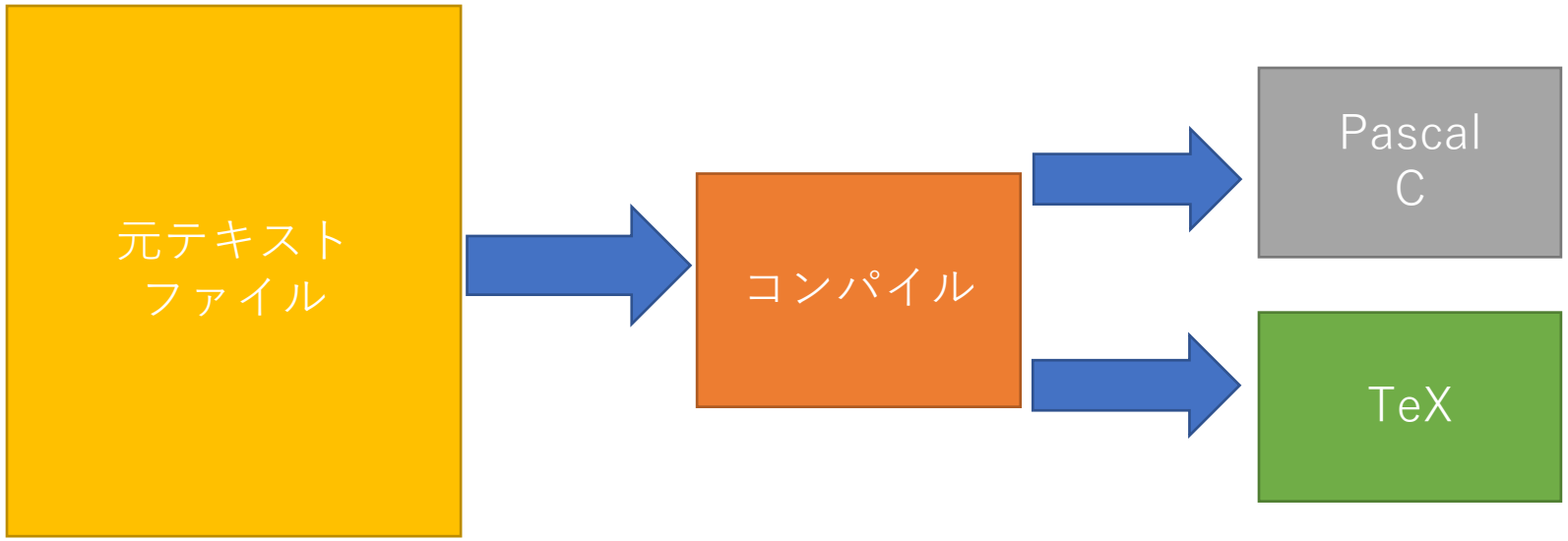
- ビジネス解析のためのモジュールが豊富

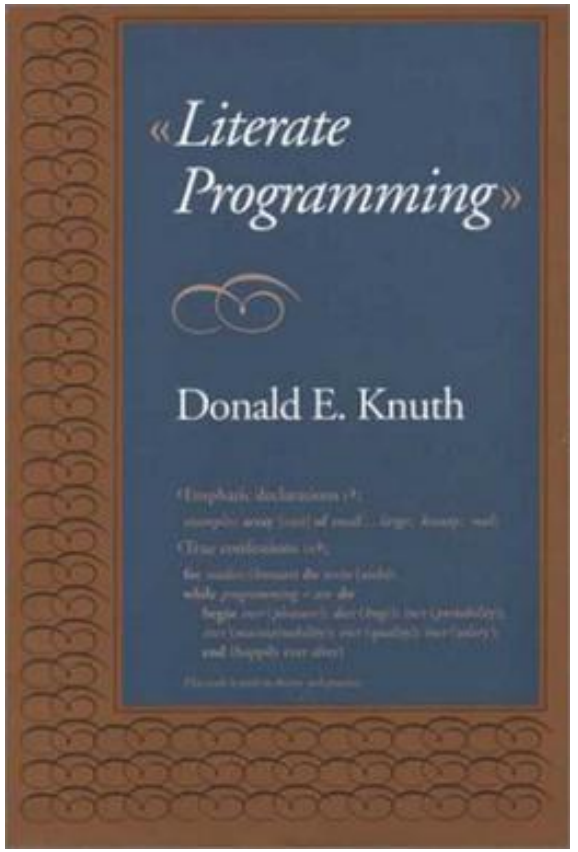




Jupyter+nbdevで文芸的プログラミング

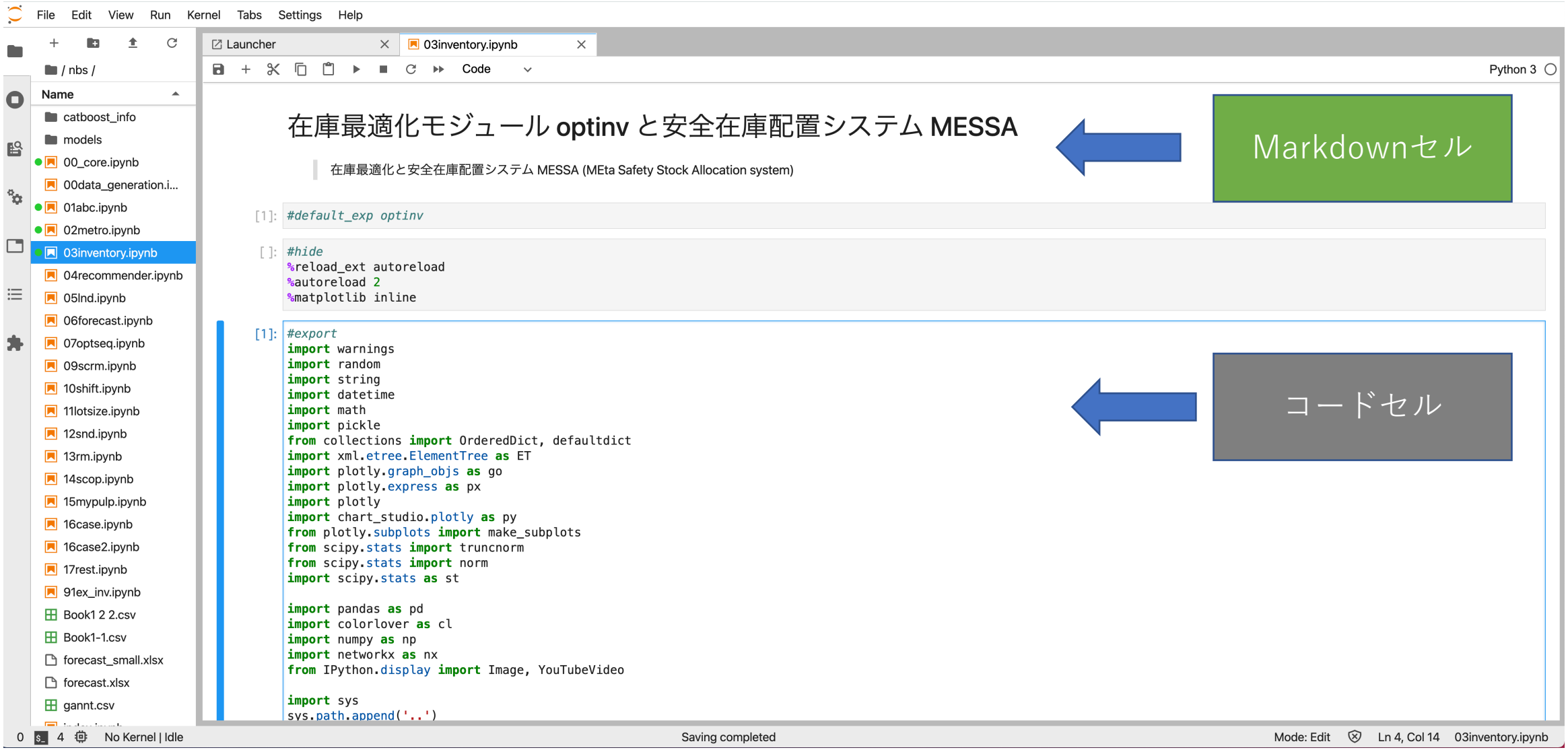
- Knuthが１９８４年に提案したプログラミングのパラダイム
- 文芸的プログラミングツール　WEB (CWEB)





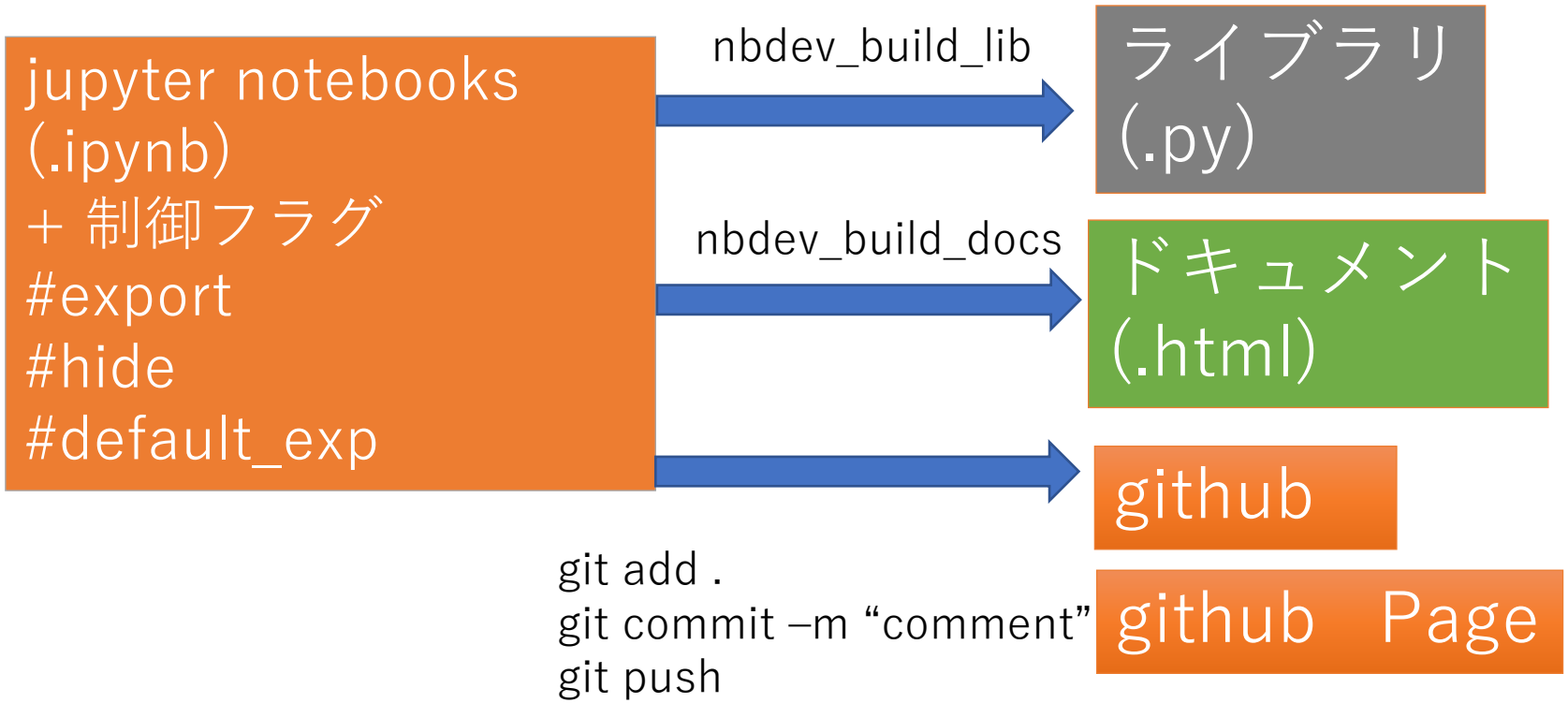
Jupyter Lab.

このままでも文芸的プログラミング => 隠したいコードやメモ，ライブラリ自動生成



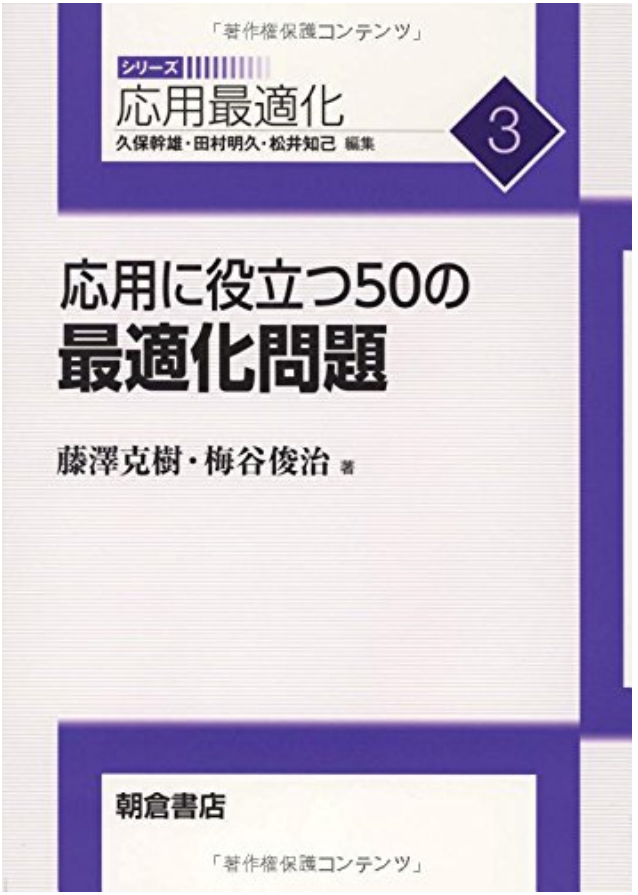
nbdev

- 深層学習パッケージ fastai の開発ためのツール
  https://nbdev.fast.ai/



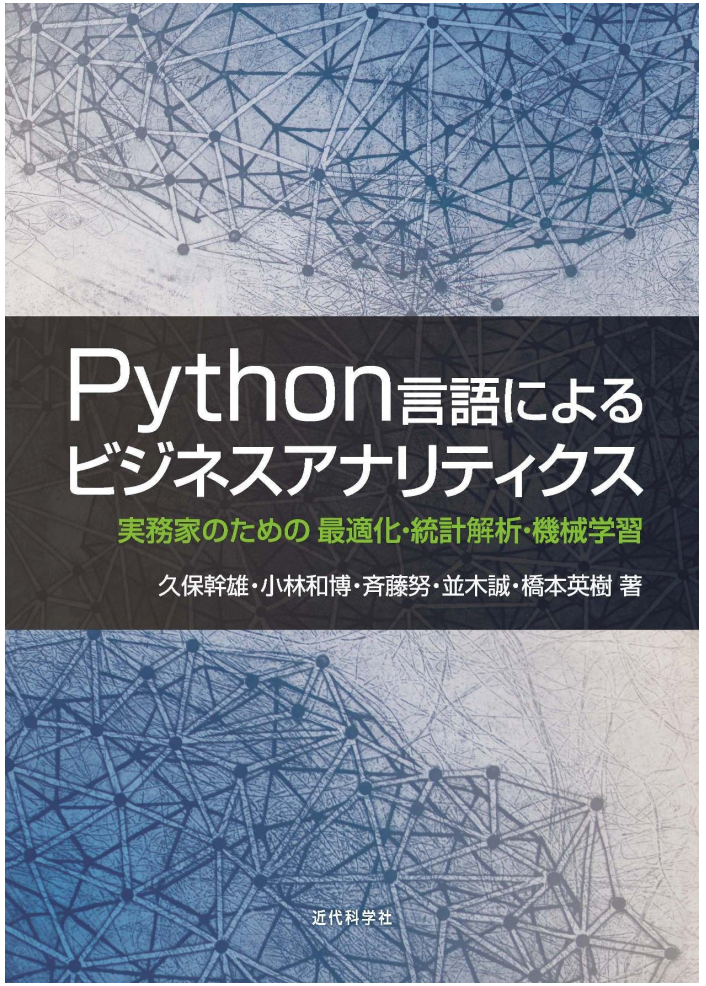
nbdev で公開中のプロジェクト (1)

- Python言語による実務で使える100+の最適化問題
  https://mikiokubo.github.io/opt100/



nbdev で公開中のプロジェクト (2)

- アナリティクス（データサイエンス）練習問題集
  https://mikiokubo.github.io/analytics/



nbdevの利点と弱点

menu

- 利点
  本にもなる（らしい）
  並列自動テスト （私はオフにしている）
  pypiとcondaに自動アップロード

- 弱点
  導入が面倒
  バグが多い
  開発が粗い（開発者が気まぐれ）

Poetry

- https://python-poetry.org/
- 仮想環境 + パッケージ管理
- パッケージの依存関係の処理
- パッケージの公開
- (pipenvよりは) 高速

menu

Pyproject.tomlの例

```
[tool.poetry.dependencies]
python = "^3.8"
uvicorn = "^0.12.3"
fastapi = "^0.61.2"
pymongo = "^3.11.1"
mongoengine = "^0.21.0"
dnspython = "^2.0.0"
colorlover = "^0.3.0"
pandas = "^1.1.4"
matplotlib = "^3.3.3"
networkx = "^2.5"
numpy = "^1.19.4"
plotly = "^4.13.0"
```

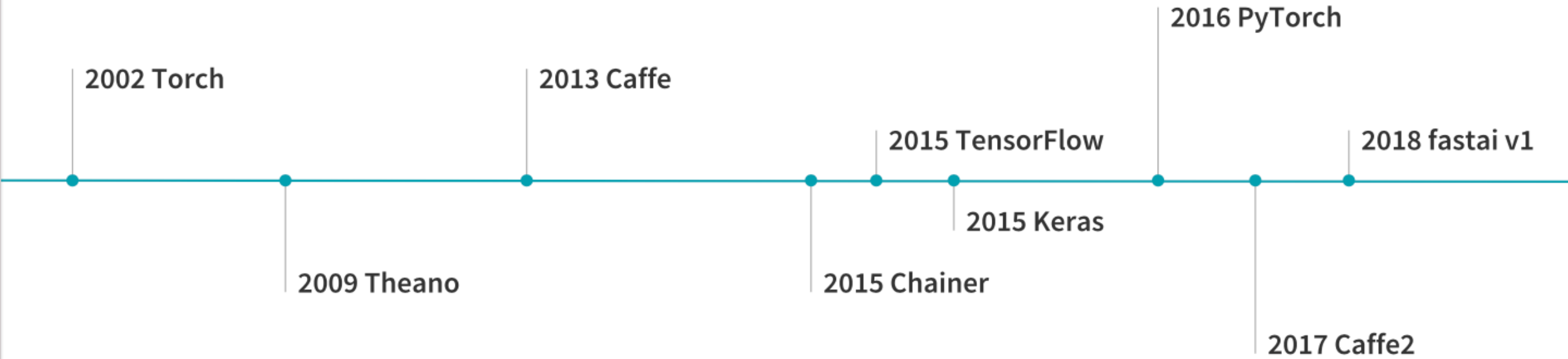
Fastai

- https://docs.fast.ai/
- PyTorchのラッパ
- 短いコードで深層学習
- 無料の講義ビデオ付き
- （割と）新しい手法が実装ずみ
- 画像，自然言語処理，表モデル，協調フィルタリング
- 自分でアーキテクチャを設計したいときにはPyTorch（ソースを改変）
- 強化学習はなし
- データサイエンス練習問題集
  https://mikiokubo.github.io/analytics/14fastai.html

fastai

Deep Learning Library

Brief History of DL Frameworks



Vision

1. import fastai

from fastai.vision import *

2. read MNIST data

mnist = untar_data(URLs.MNIST_TINY)

tfms = get_transforms(do_flip=False)

3. prepare data bunch

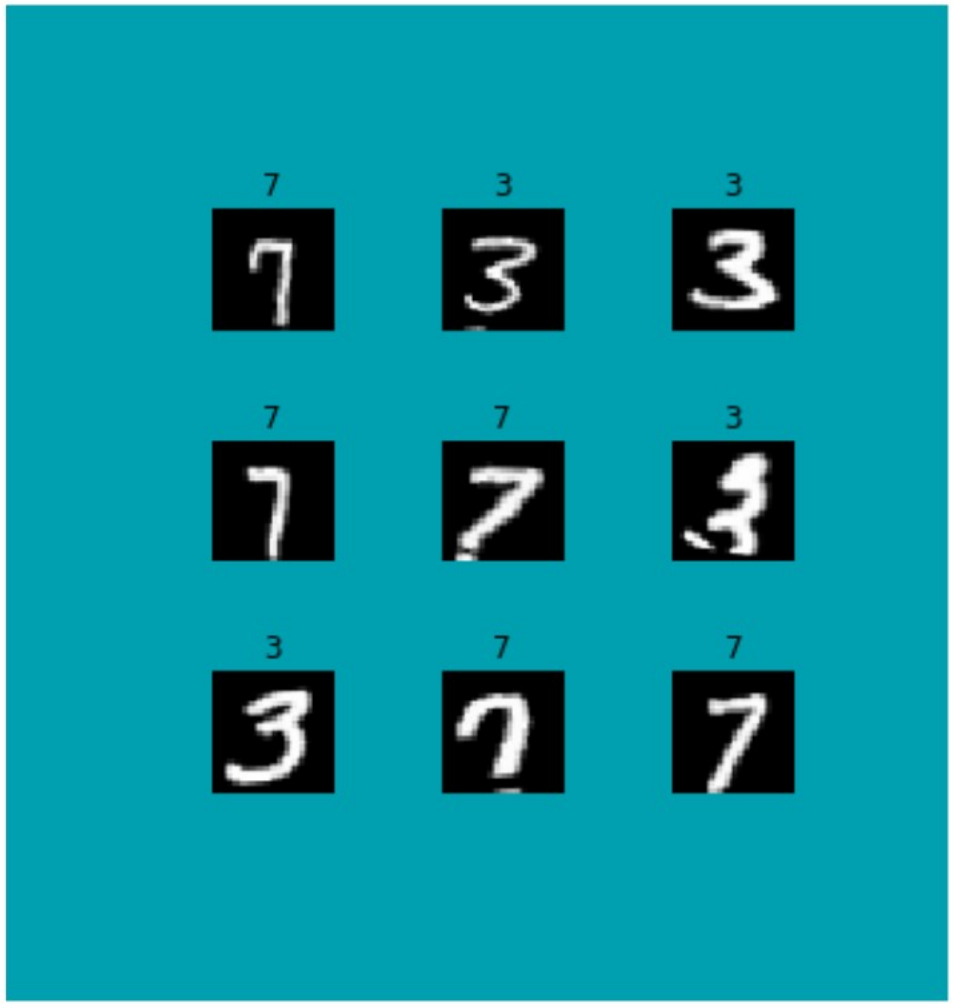
Data = (ImageItemList.from_folder(mnist)

- .split_by_folder() .label_from_folder()
- .transform(tfms, size=32)
- .databunch()
- .normalize(imagenet_stats))

4. show data

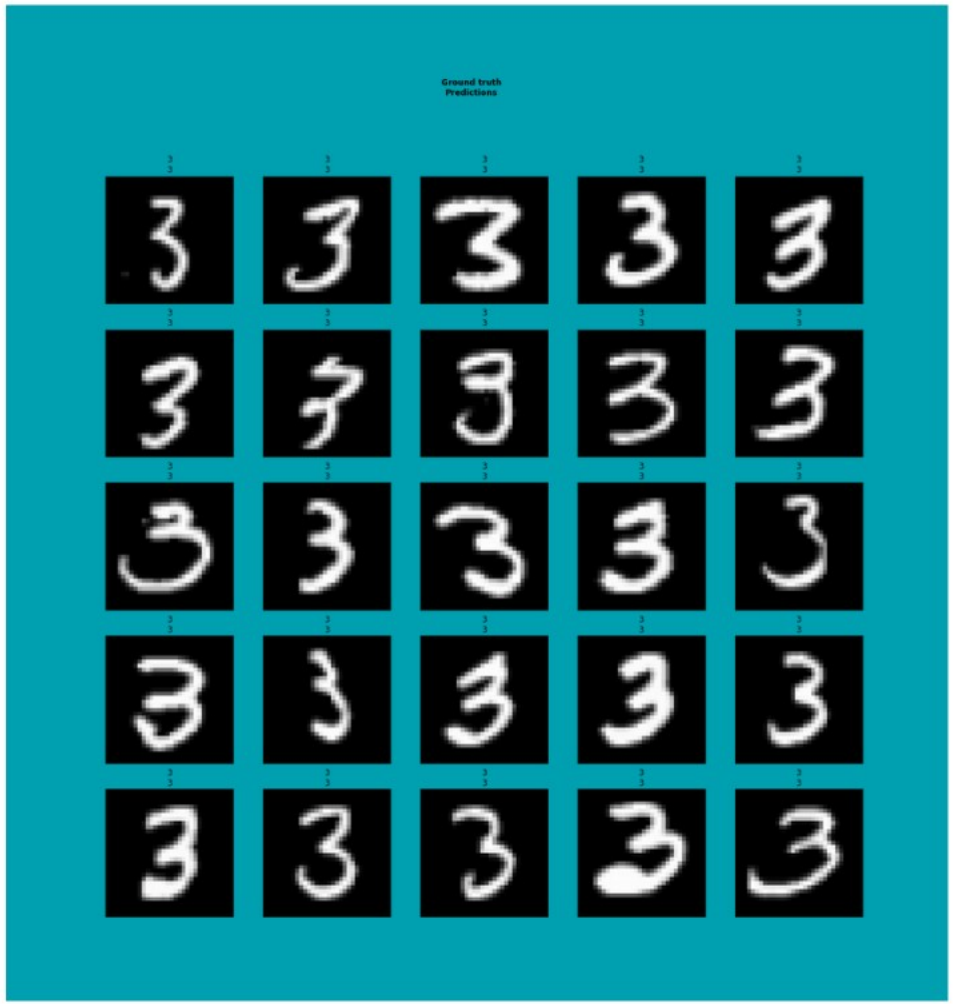
data.show_batch(rows=3, figsize=(4,4))

Show data




```
data.show_batch(rows=3, figsize=(4,4))
```

Train & show results




```
learn = create_cnn(data, models.resnet18, metrics=accuracy)
learn.fit_one_cycle(1,1e-2)

learn.show_results()
```

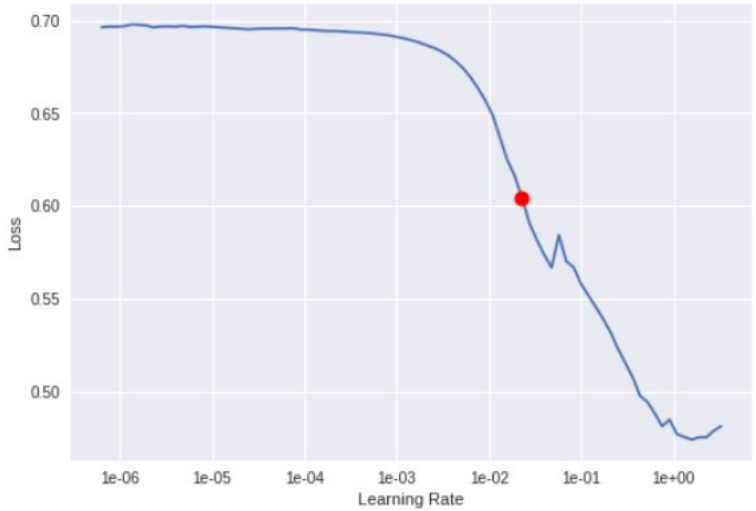
Make another model

```
model = simple_cnn((3,16,16,2))
learn = Learner(data,model,metrics=[accuracy])

learn.summary
```

```
opt_func=functools.partial(<class 'torch.optim.adam.Adam'>, betas=(0.9, 0.99)), loss_func=<fastai.layers.FlattenedLoss object at
0x7fa5f5fd2080>, metrics=[<function accuracy at 0x7fa5f70738c8>], true_wd=True, bn_wd=True, wd=0.01, train_bn=True,
path=PosixPath('/root/.fastai/data/mnist_sample'), model_dir='models', callback_fns=[<class 'fastai.basic_train.Recorder'>],
callbacks=[], layer_groups=[Sequential(
  (0): Conv2d(3, 16, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
  (1): ReLU(inplace)
  (2): Conv2d(16, 16, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
  (3): ReLU(inplace)
  (4): Conv2d(16, 2, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
  (5): ReLU(inplace)
  (6): AdaptiveAvgPool2d(output_size=1)
  (7): Lambda()
```

Find a good learning rate

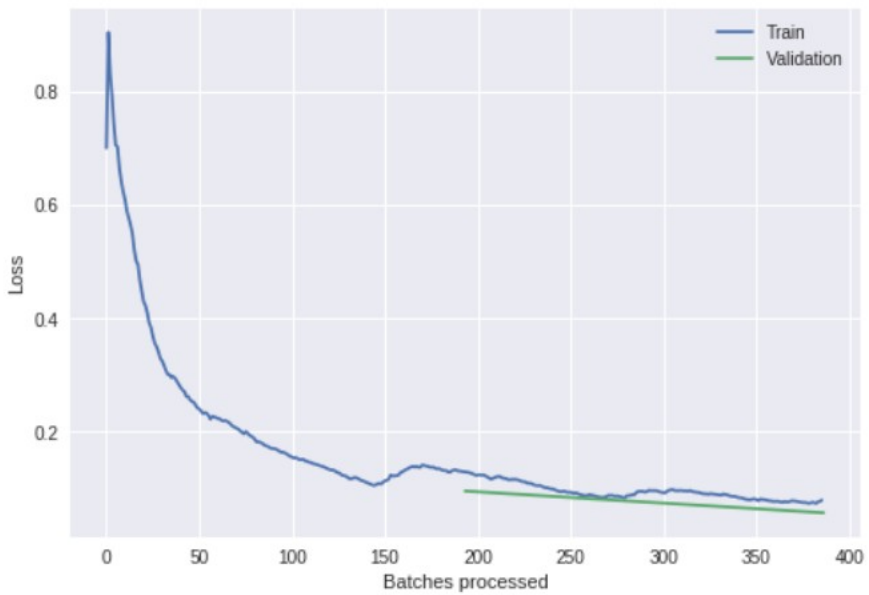


learn.lr_find()

learn.recorder.plot()

5e-2 is a good choice !

Train



- **Train 2 epochs using lr=0.05**

  learn.fit(2,5e-2)

  □epoch□train_loss□valid_loss□accuracy

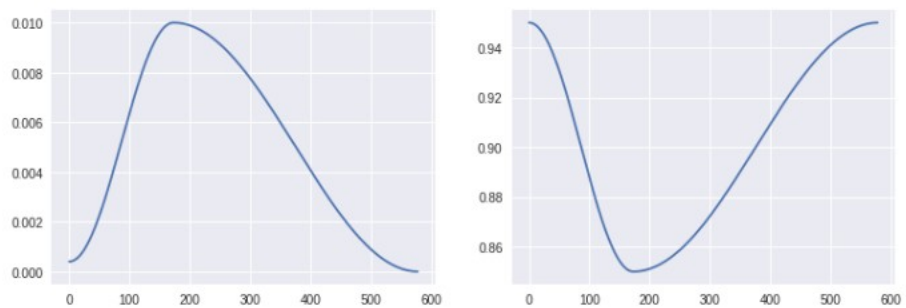
  □1□0.129588□0.095395□0.968106

  □2□0.079258□0.056953□0.981845

  □

  learn.recorder.plot_losses()

Use fit-one-cycle



- **Train 3 epochs using fit-one cycle method**

learn.fit_one_cycle(3,max_lr=5e-2)

>>>

epoch train_loss valid_loss accuracy

1 0.104979 0.059402 0.976938

2 0.034898 0.020833 0.992149

3 0.020161 0.016893 0.992149

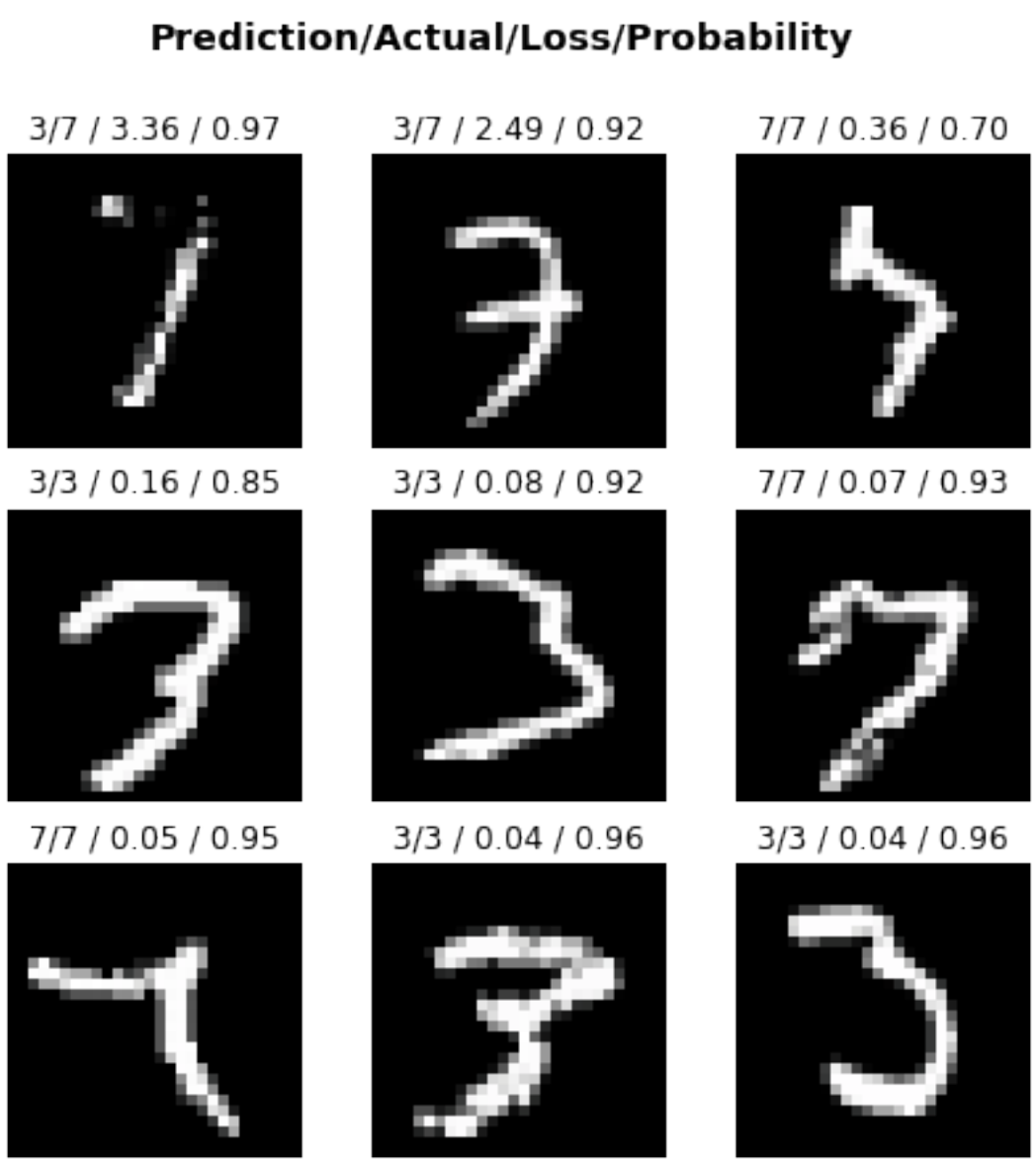
lr (left), momentum (right)

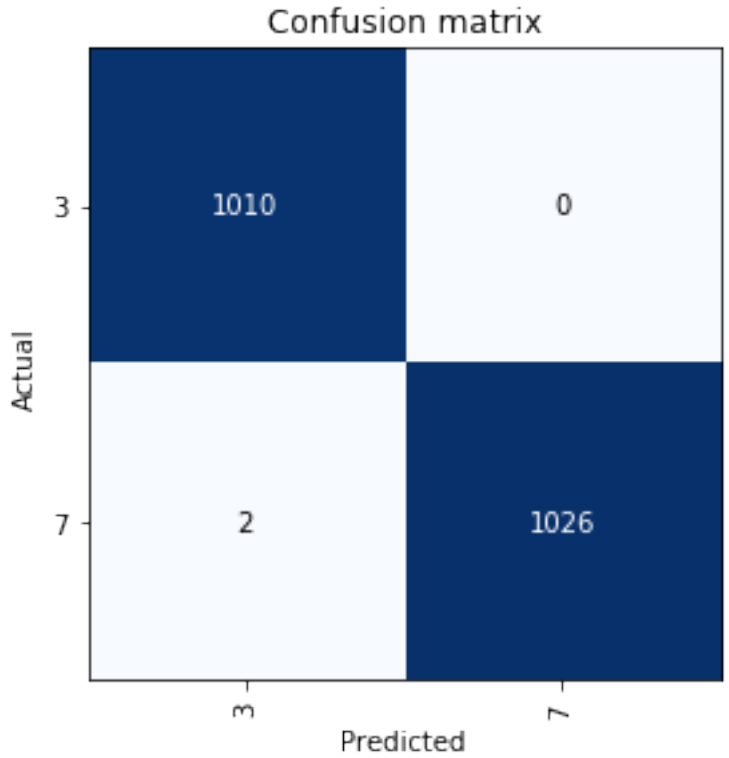
preds,y,losses = learn.get_preds(with_loss=True)

interp = ClassificationInterpretation(data, preds, y, losses)

interp.plot_top_losses(9, figsize=(7,7))

interp.plot_confusion_matrix()

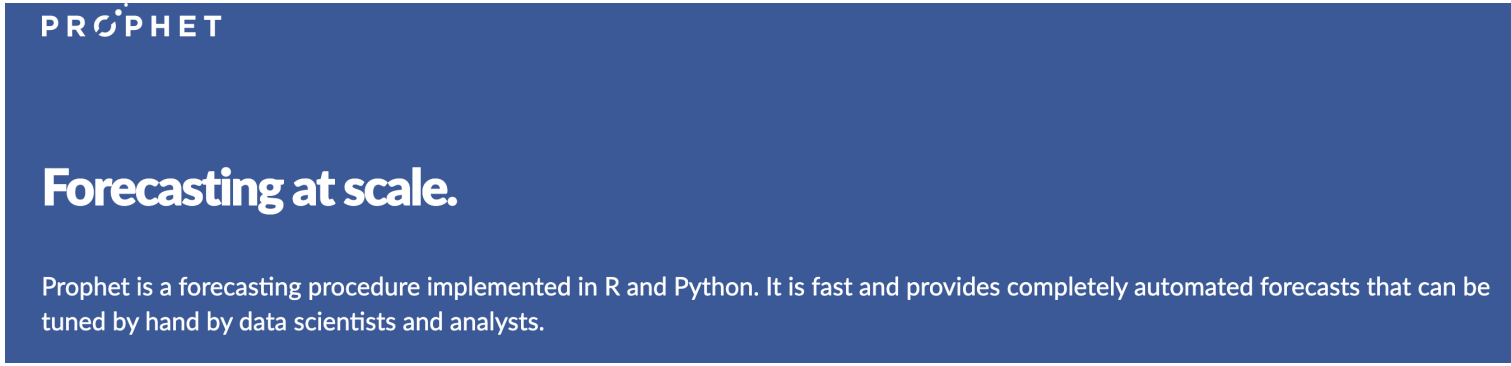

Prediction/Actual/Loss/Probability
3/7 / 3.36 / 0.97
3/7 / 2.49 / 0.92
7/7 / 0.36 / 0.70
3/3 / 0.16 / 0.85
3/3 / 0.08 / 0.92
7/7 / 0.07 / 0.93
7/7 / 0.05 / 0.95
3/3 / 0.04 / 0.96
3/3 / 0.04 / 0.96


Confusion matrix
Actual
3
7
1010
0
2
1026
3
7
Predicted

Pycaret

- https://pycaret.org/
- （自動）機械学習
- 前処理＋モデル構築＋訓練＋アンサンブル＋デプロイ
- 回帰，分類，クラスタリング，異常検知，自然言語処理，アソシエーション・ルール・マイニング
- 結果の可視化
- モデルの解釈 (SHAP: SHapley Additive exPlanations)
- データサイエンス練習問題集 https://mikiokubo.github.io/analytics/19pycaret.html

Prophet



- https://facebook.github.io/prophet/
- （自動）需要予測
- ベイズ推論（pystan利用）
- データサイエンス練習問題集 https://mikiokubo.github.io/analytics/15forecast.html

（自動）機械学習(Auto ML)と深層学習(DL)とベイズ推論を用いた予測手法について

東京海洋大学

久保幹雄

（量的）予測と回帰

- 予測（時系列）モデル
  過去のデータが与えられたとき，未来の値を予測

$$\hat{y}_t = f(y_{t-1}, y_{t-2}, \cdots, y_1)$$

- 回帰モデル
  訓練データ集合が与えられたとき，検証データ集合における損出関数を最小にするパラメータを推定

$$\hat{y} = g(x, \theta)$$

従属変数（目的変数）　独立変数（説明変数）

時系列モデル

- ARIMAモデル

$$y_t - y_{t-d} = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q}$$

integrated　　autoregressive　　moving average

- 3次指数平滑(Holt-Winter)モデル

$$s_0 = x_0$$

$$s_t = \alpha(x_t - c_{t-L}) + (1-\alpha)(s_{t-1} + b_{t-1})$$

$$b_t = \beta(s_t - s_{t-1}) + (1-\beta)b_{t-1}$$

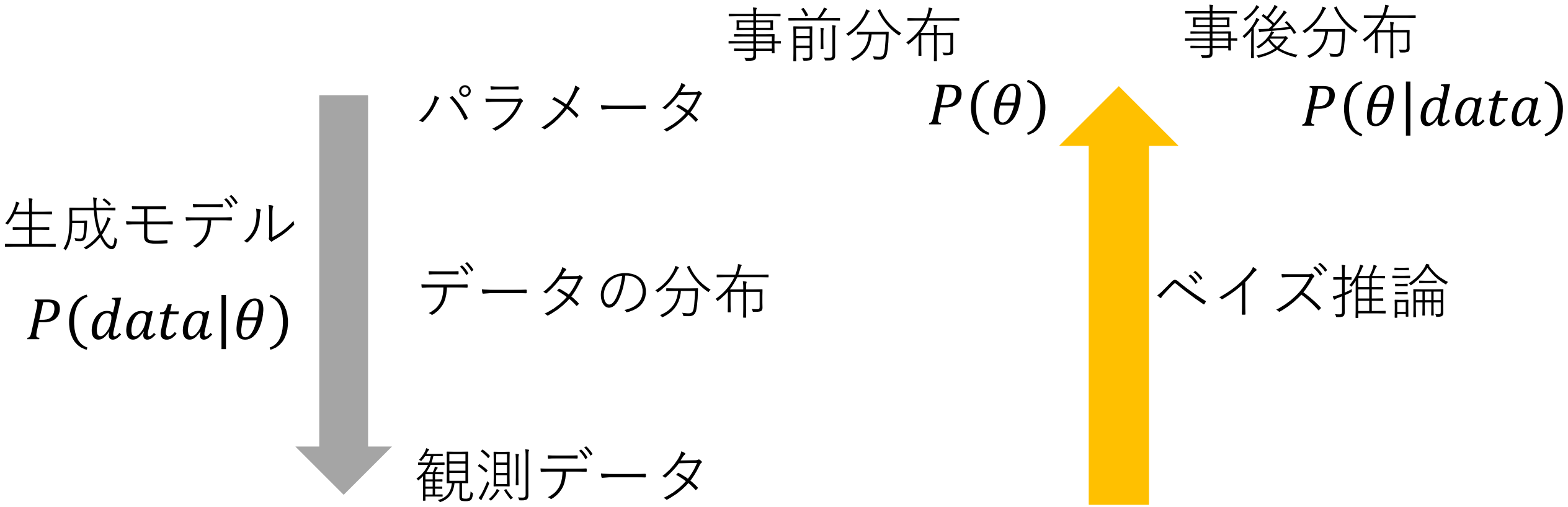
$$c_t = \gamma(x_t - s_{t-1} - b_{t-1}) + (1-\gamma)c_{t-L}$$

$$y_{t+m} = s_t + mb_t + c_{t-L+1+(m-1) \bmod L}$$

手法の比較

機械学習 （深層学習を含む）		ベイズ推論 （確率的プログラミング）
	⟺	明確な推論 （ホワイトボックス）
	⟺	完全な柔軟性
	⟺	分布まで考慮した 不確実性

確率的プログラミング



ベイズの公式　$P(\theta|data) \propto P(data|\theta)P(\theta)$

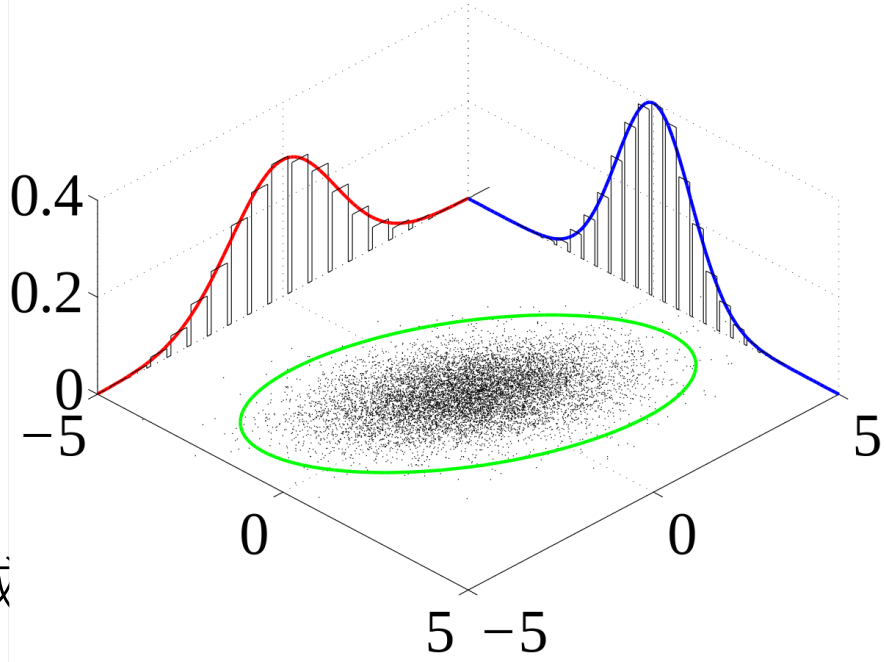
確率的プログラミングの手法と実装

- マルコフ連鎖モンテカルロ法
  （Markov Chain Monte Carlo methods：MCMC）
  ＝データに合った近似確率分布を生成するシミュレーション
- PyMC3 (4)
- Stan

予測に特化したとして実装として Prophet (Facebook)

=> Stanを用いた予測モデル

MCMC



- 適当な初期パラメータ θ
- 収束するまで以下を反復
  - 提案分布にしたがい新しい点 θ^{new} を生成
  - 推移確率を計算

$$q = P(\theta^{new}|data)/P(\theta|data)$$

  - 推移確率が1以上なら確率1で，そうでなければ確率 qで　$\theta = \theta^{new}$

Prophetのモデル(一般化加法モデル)

$$y_t = g_t + s_t + h_t + \epsilon_t$$

y_t　：予測値

g_t　：傾向変動（変化点ありの線形・ロジスティック曲線）

s_t　：季節変動（フーリエ級数）

h_t　：休日（イベント）効果（特定日を指定して分布を予測）

ϵ_t　：誤差項

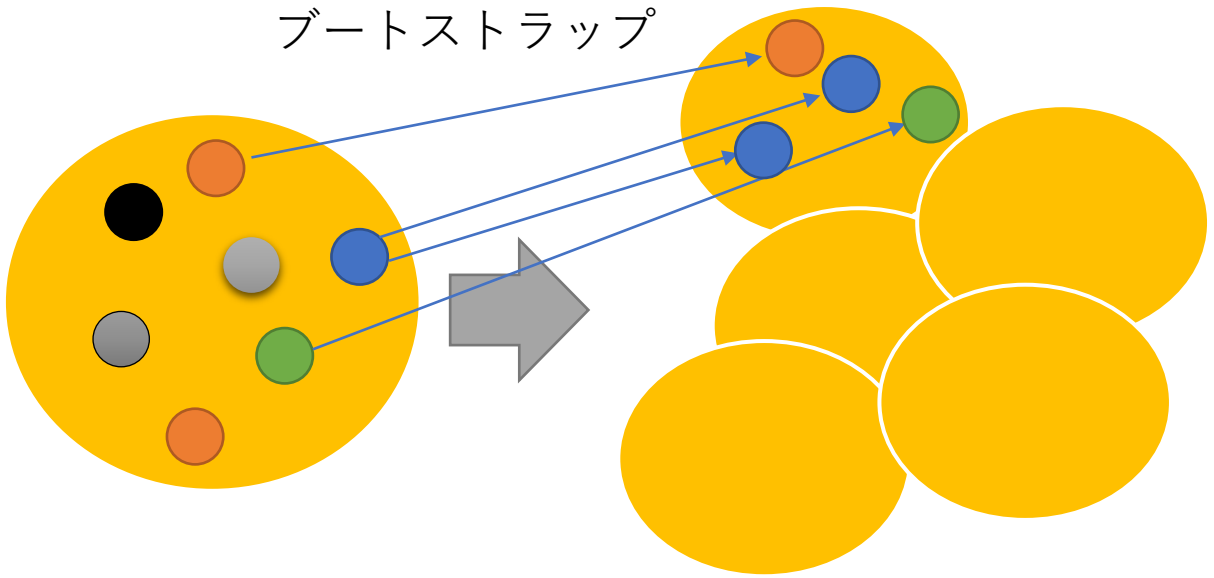
機械学習による予測（回帰）

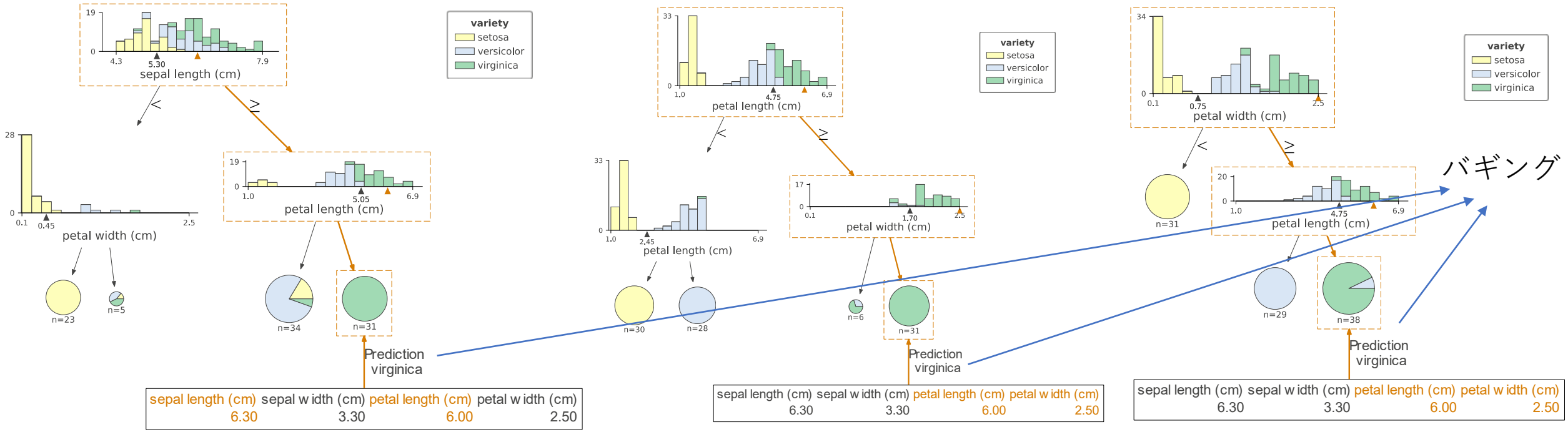
- 決定木のアンサンブル
  - バギング = bootstrap aggregating（ランダム森，Extra木）
  - ブースティング（適応型ブースティング，勾配ブースティング(GBM), Light GBM，Extreme GBM，カテゴリーブースティング）

- 深層学習
  ＝複雑な関数を任意の精度で近似して出力するアーキテクチャ
  - 畳み込みニューラルネット（2次元畳み込み）
  - LSTM（Long Short Term Memory）
    =実用的な回帰型ニューラルネットワーク(Recurrent NN)
  - 埋め込みニューラルネット

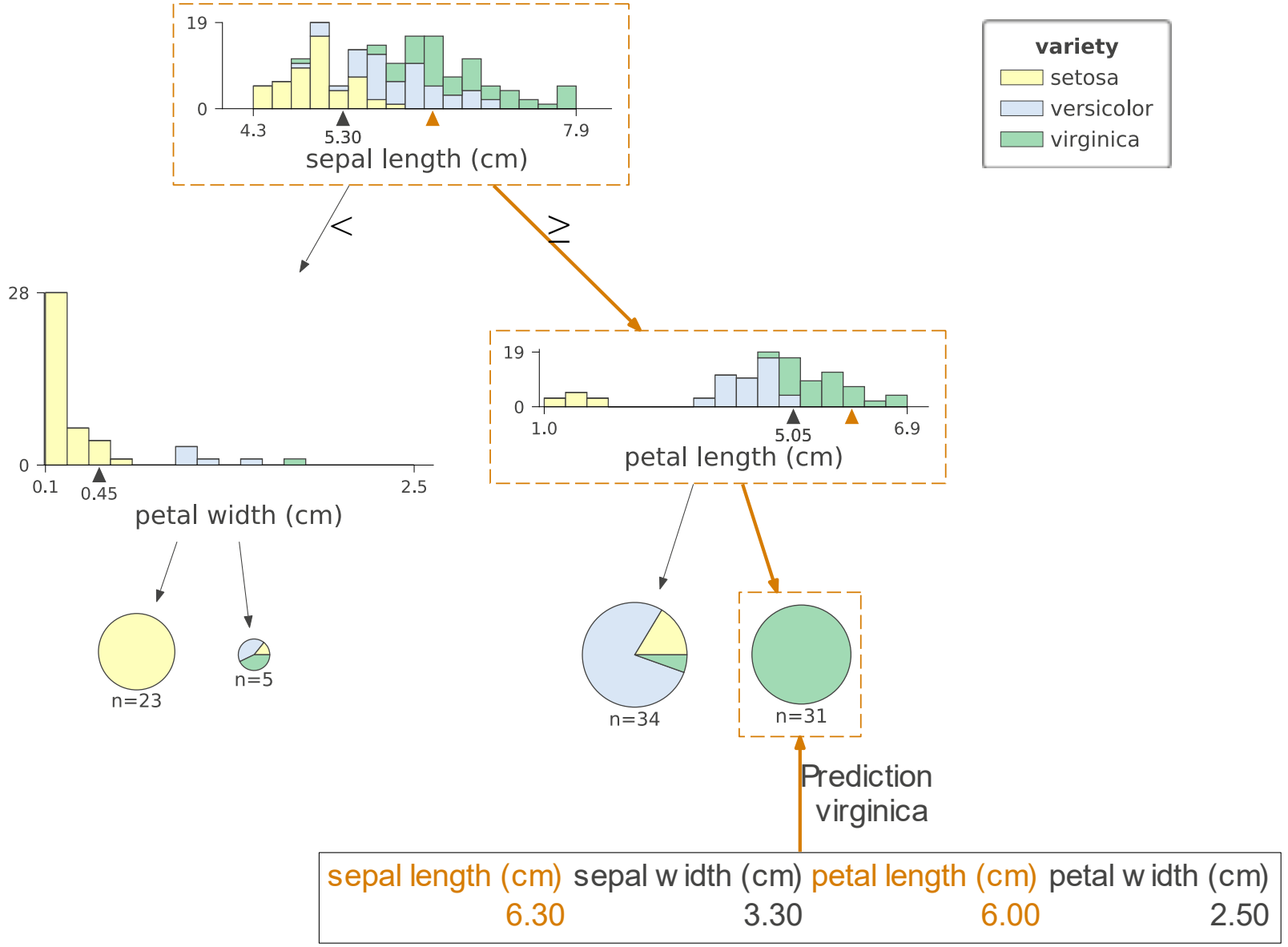
ランダム森

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5	3.6	1.4	0.2	Iris-setosa

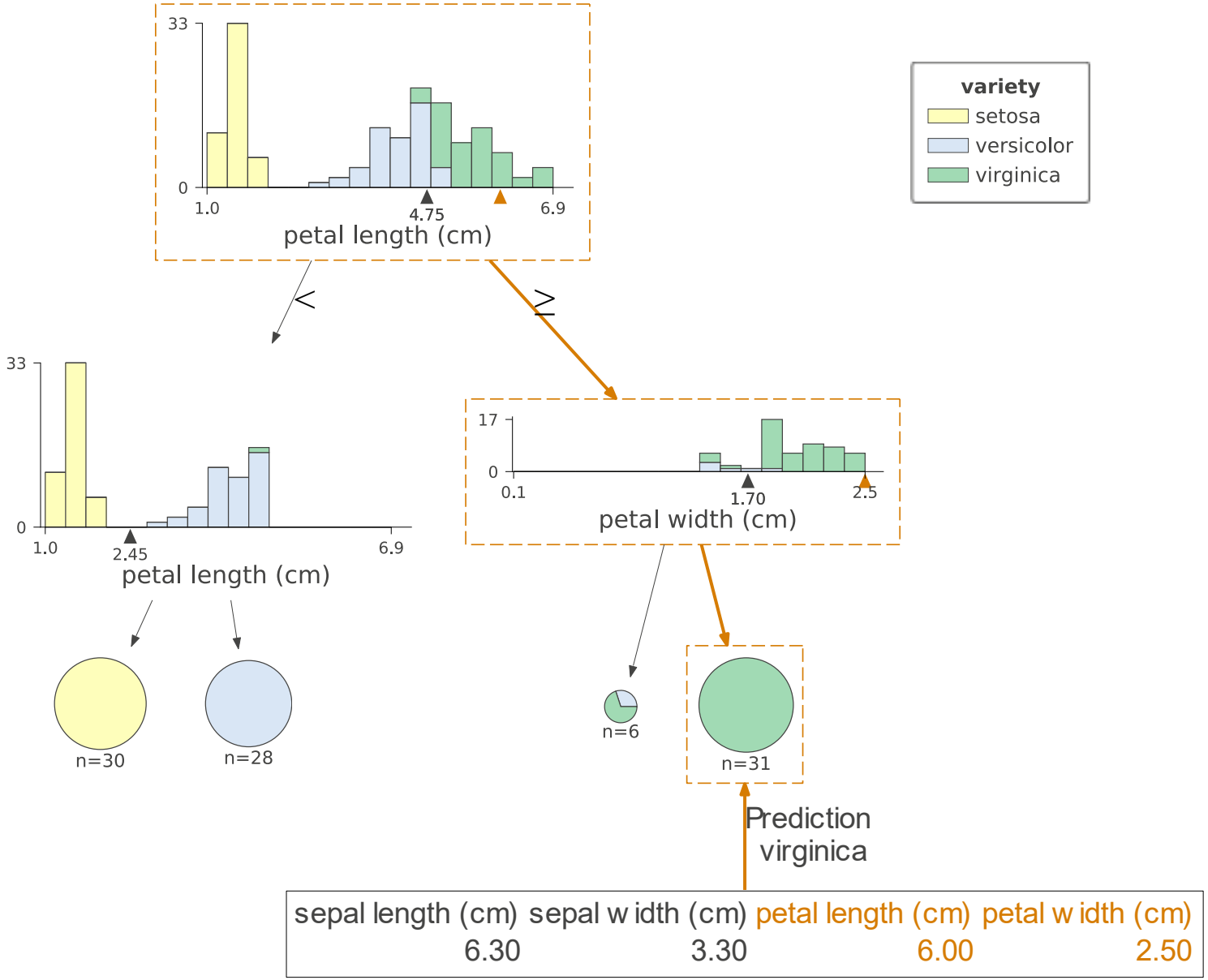




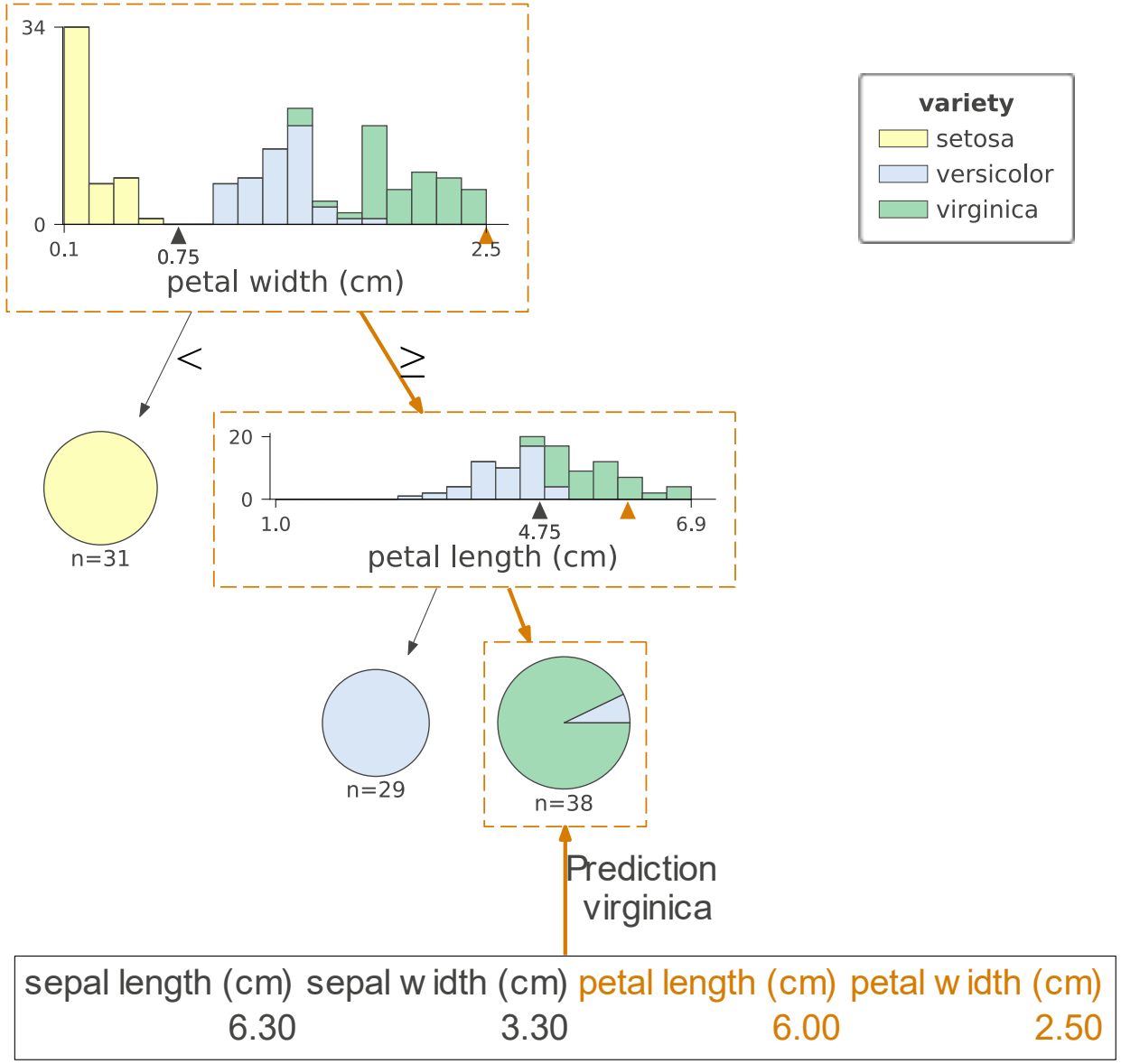
決定木 1


19
0
4.3
5.30
7.9
sepal length (cm)
<
≥
variety
setosa
versicolor
virginica
28
0
0.1
0.45
2.5
petal width (cm)
n=23
n=5
19
0
1.0
5.05
6.9
petal length (cm)
n=34
n=31
Prediction
virginica
sepal length (cm) sepal width (cm) petal length (cm) petal width (cm)
6.30 3.30 6.00 2.50

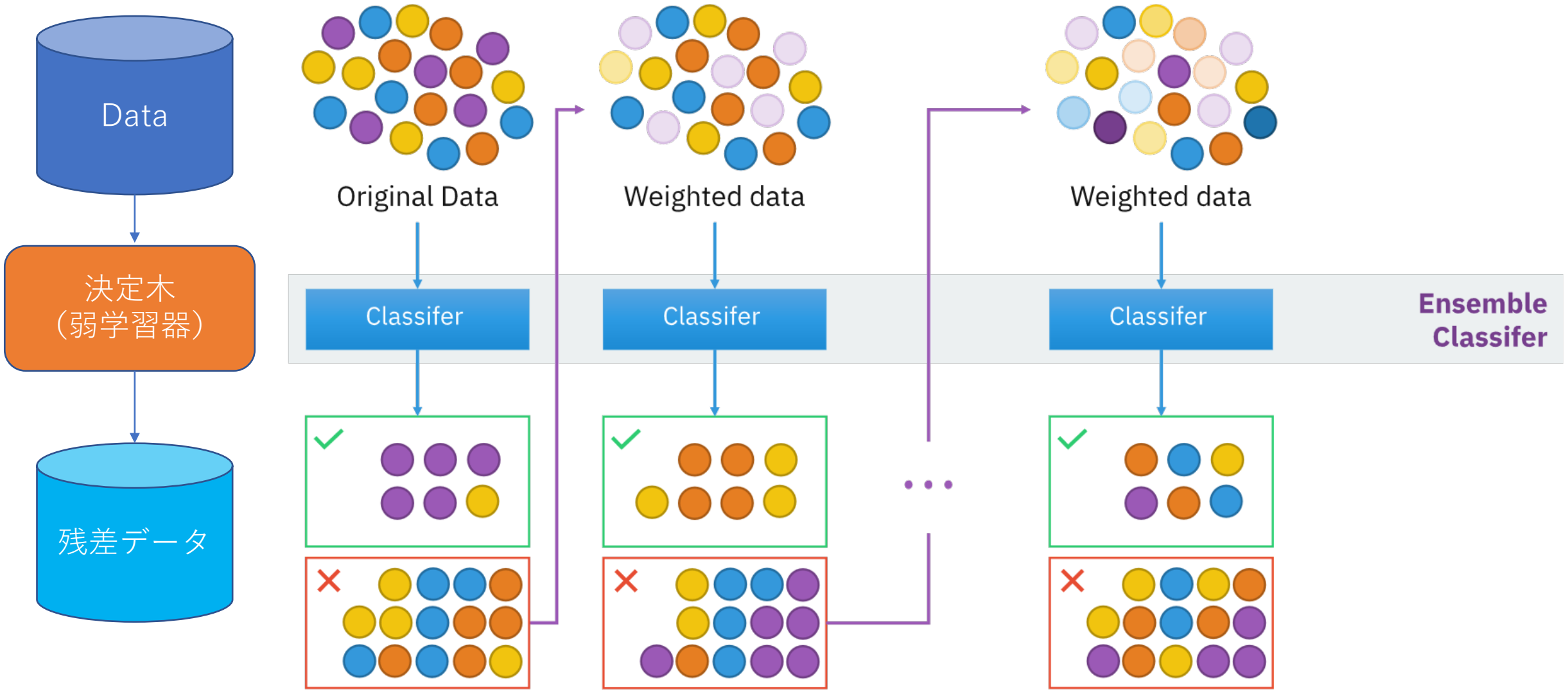
決定木 2


33
0
1.0
4.75
6.9
petal length (cm)
variety
setosa
versicolor
virginica
<
≥
33
0
1.0
2.45
6.9
petal length (cm)
17
0
0.1
1.70
2.5
petal width (cm)
n=30
n=28
n=6
n=31
Prediction
virginica
sepal length (cm) sepal w idth (cm) petal length (cm) petal w idth (cm)
6.30 3.30 6.00 2.50

決定木 3


34
0
0.1
0.75
2.5
petal width (cm)
<
≥
n=31
20
0
1.0
4.75
6.9
petal length (cm)
n=29
n=38
Prediction
virginica
variety
setosa
versicolor
virginica
sepal length (cm) sepal w idth (cm) petal length (cm) petal w idth (cm)
6.30 3.30 6.00 2.50

ブースティング

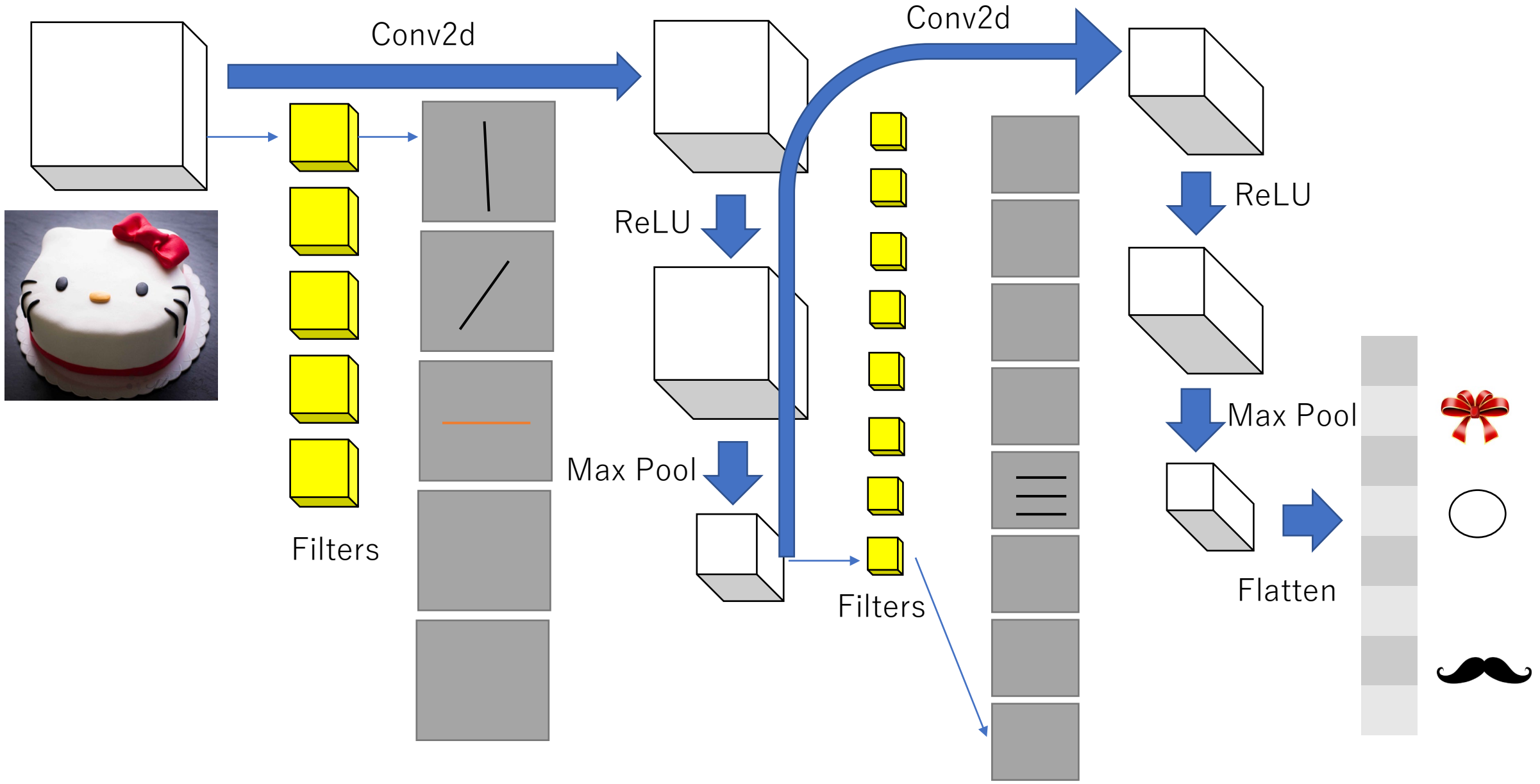

Data
決定木
(弱学習器)
残差データ
Original Data
Weighted data
Weighted data
Classifer
Classifer
Classifer
Ensemble
Classifer

自動機械学習 Auto ML

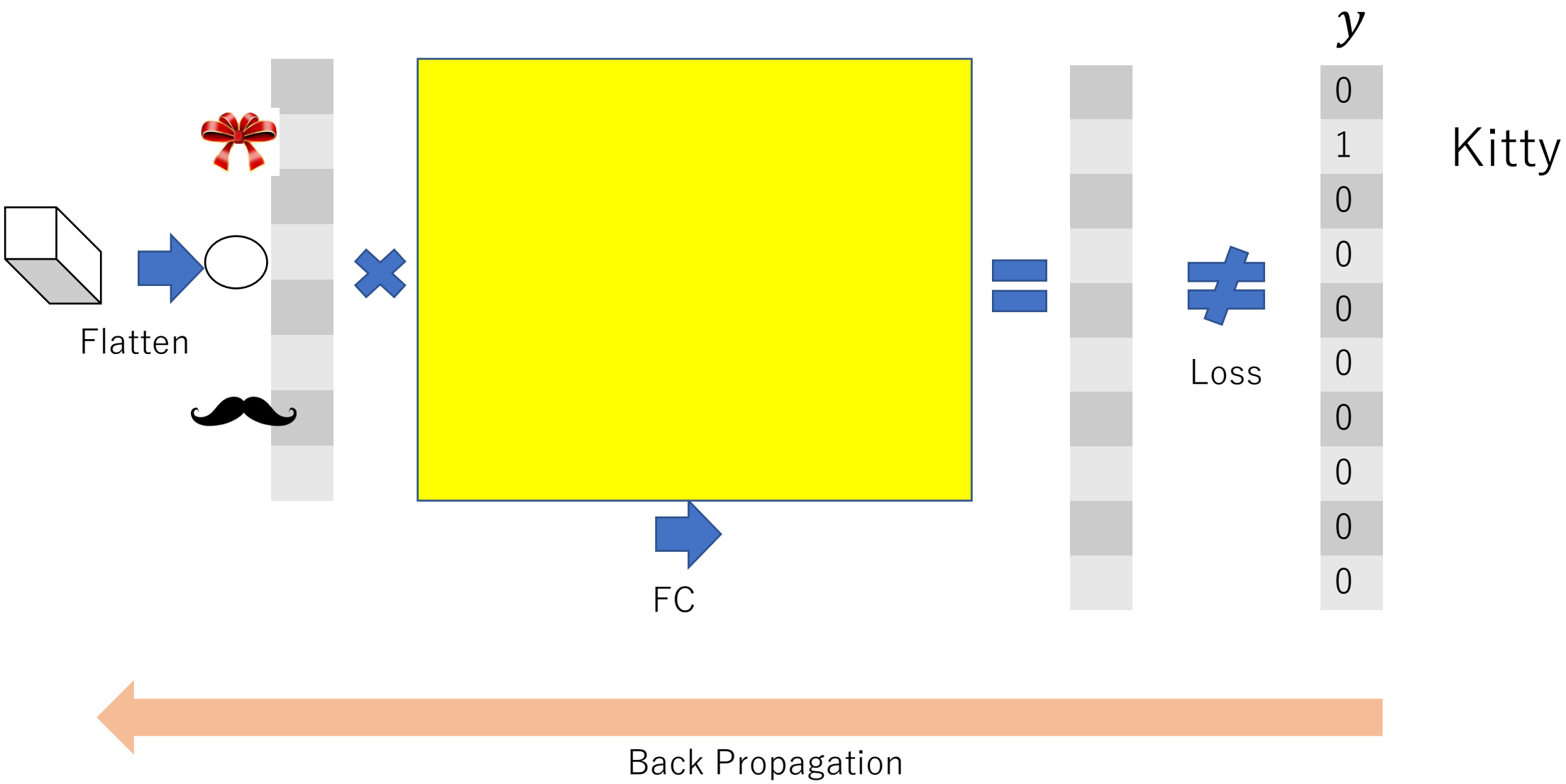
- たくさんの機械学習の手法を同時に行い良いものを選ぶ.
- 良いものをアンサンブル

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
et	Extra Trees Regressor	0.4636	0.4248	0.6334	0.9837	0.0705	0.0523	0.0440
gbr	Gradient Boosting Regressor	0.6601	0.7945	0.8653	0.9704	0.0910	0.0709	0.0100
xgboost	Extreme Gradient Boosting	0.6984	0.8608	0.9045	0.9675	0.0893	0.0698	0.0360
rf	Random Forest Regressor	0.7123	0.8845	0.9210	0.9667	0.0945	0.0750	0.0500
catboost	CatBoost Regressor	0.5846	0.9041	0.8906	0.9655	0.1063	0.0776	1.4960
ada	AdaBoost Regressor	0.9116	1.3416	1.1366	0.9485	0.1020	0.0903	0.0160
dt	Decision Tree Regressor	0.9474	1.4913	1.2042	0.9418	0.1032	0.0836	0.0060
lightgbm	Light Gradient Boosting Machine	1.1357	2.4086	1.5262	0.9037	0.1636	0.1345	0.0120

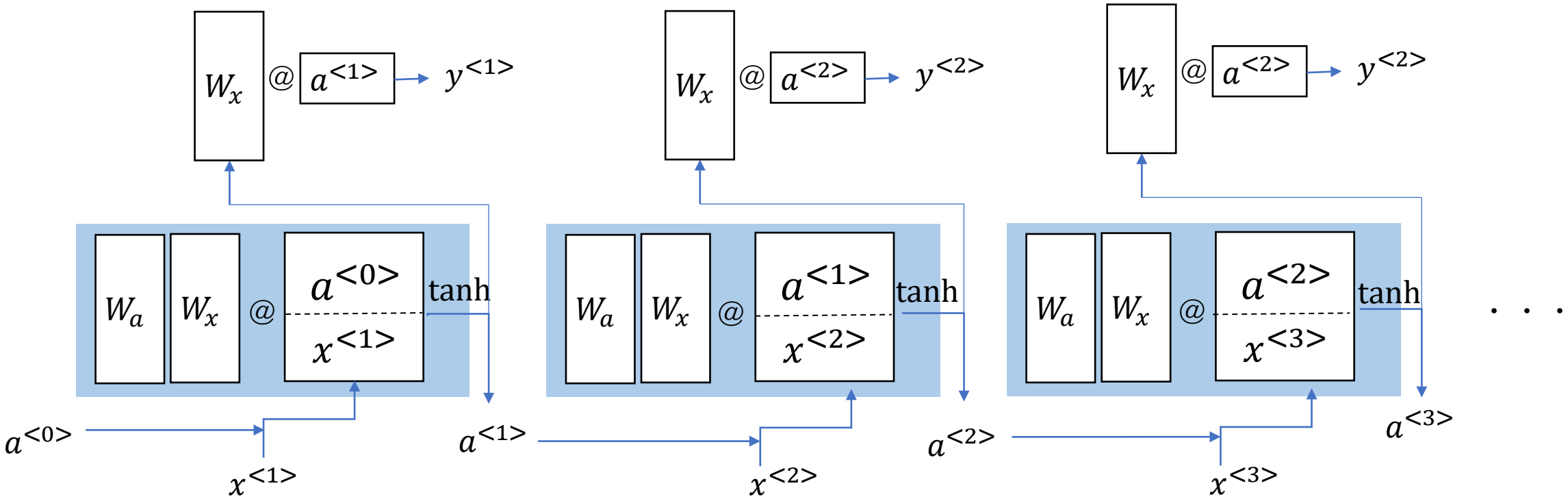
畳み込みニューラルネット (1)



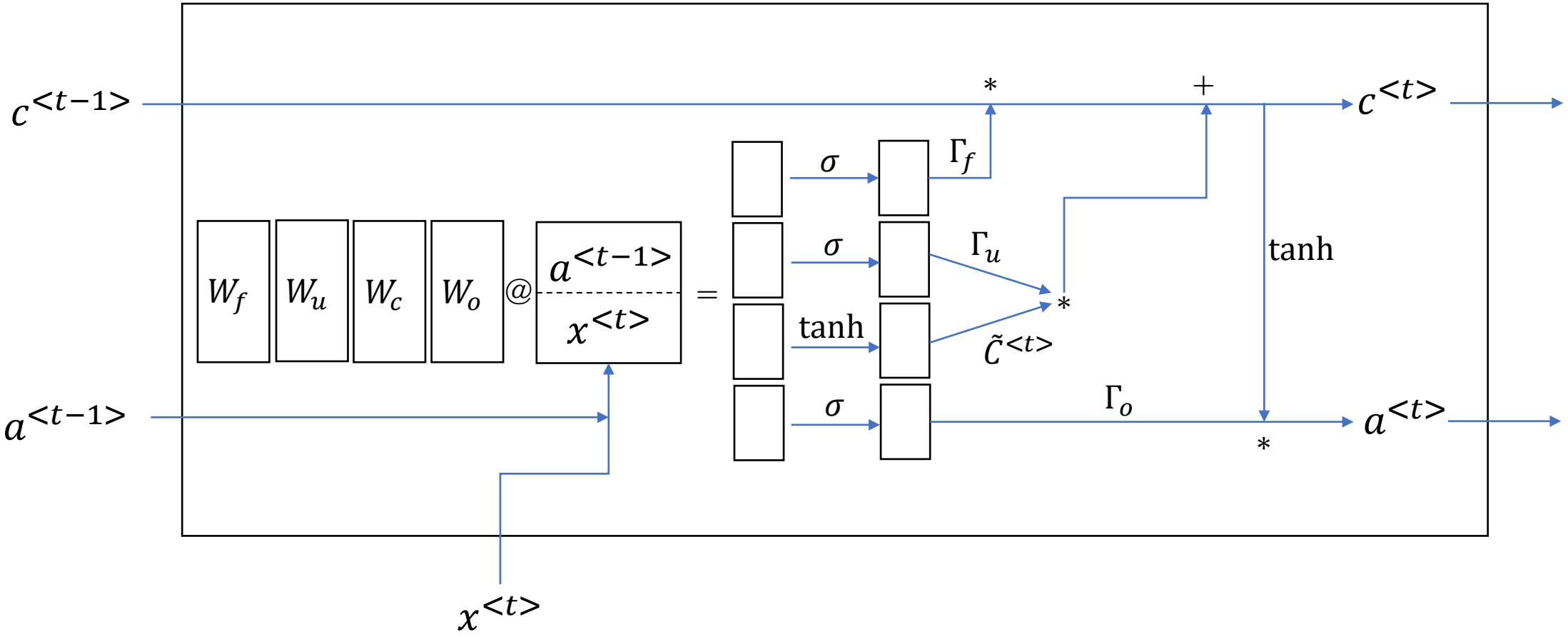
畳み込みニューラルネット (2)



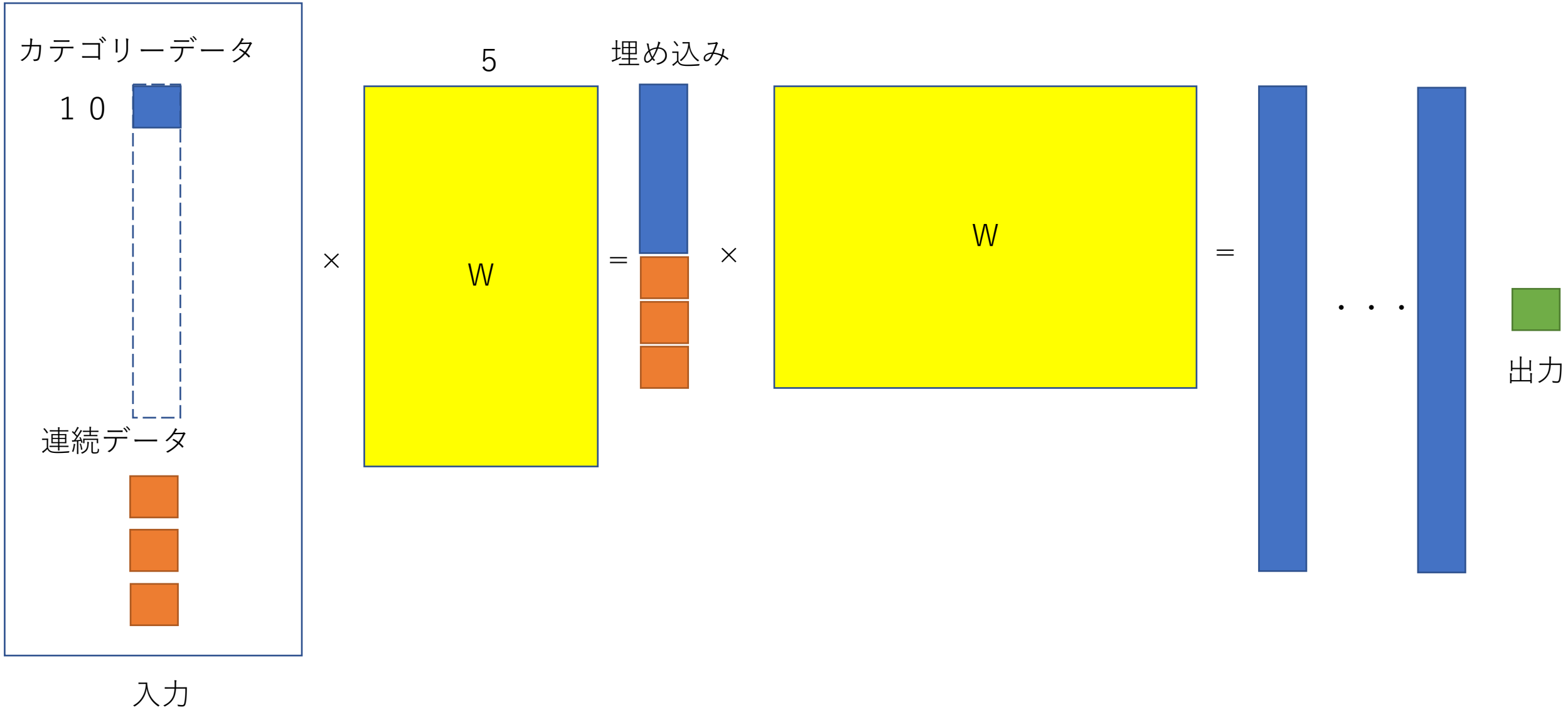
回帰型ニューラルネット



LSTM（時系列データに対する予測）

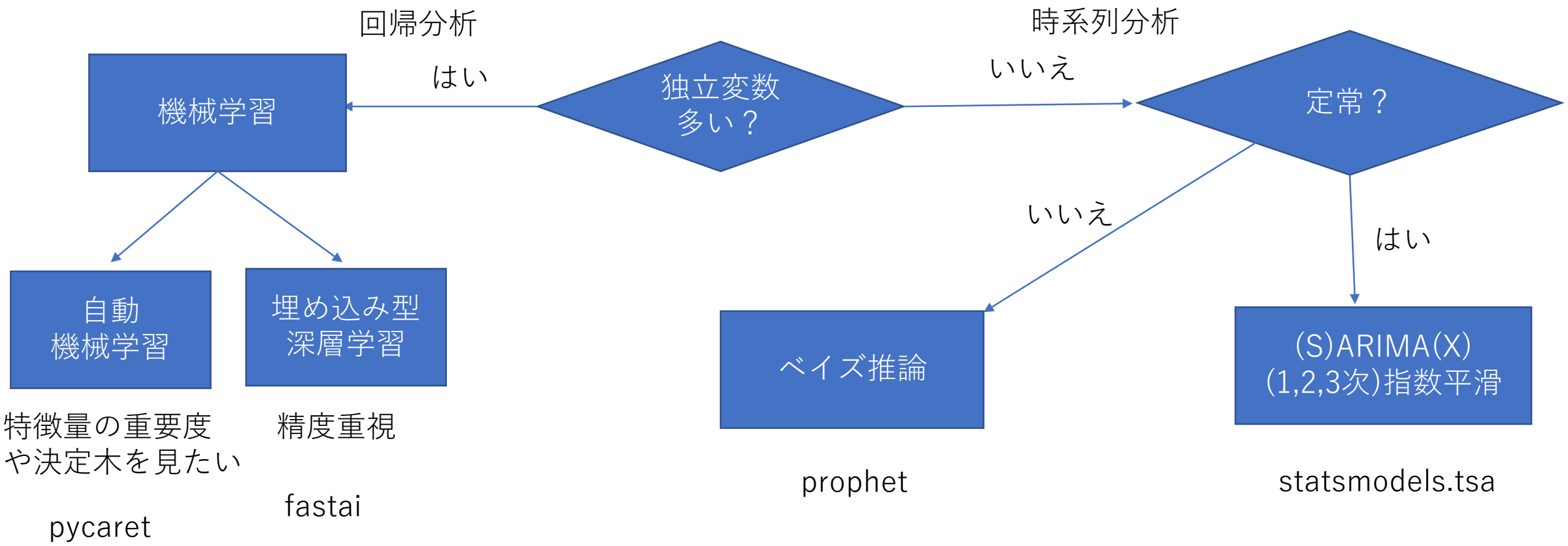


埋め込み層を用いた深層学習



使い分け

menu


回帰分析
時系列分析
はい
いいえ
機械学習
独立変数
多い？
定常？
いいえ
はい
自動
機械学習
埋め込み型
深層学習
ベイズ推論
(S)ARIMA(X)
(1,2,3次)指数平滑
特徴量の重要度
や決定木を見たい
精度重視
prophet
statsmodels.tsa
pycaret
fastai

Gurobi

- https://www.gurobi.com/
- 混合整数最適化ソルバー
- 作成者: Zonghao Gu, Edward Rothberg, Robert Bixby
- 大規模な混合整数二次(錘)最適化をマルチコアで求解
- 商用(でもアカデミックフリー)
- (おそらく)最速
- Pythonインターフェィスあり
- (ほぼ)同じインターフェィスのオープンソース・パッケージ
  mypulp (pulpのラッパ;ソルバーはCBC)

数理最適化ソルバーの性能向上

- CPLEX 1.2 (1991) -> CPLEX 11 (2007) : 29000倍
- Gurobi 1.0 (2009) -> Gurobi 9.0 (2019) : 59倍
- 合わせると．．．　170万倍
- 計算機の速度向上
  59.7 Gflops/s (1993) -> 93.0 Pflops/s (2016)
- 合わせると．．．　2.2兆倍

以前は 7 万年かかっていた計算が 1 秒以下！

最適化の効果

menu

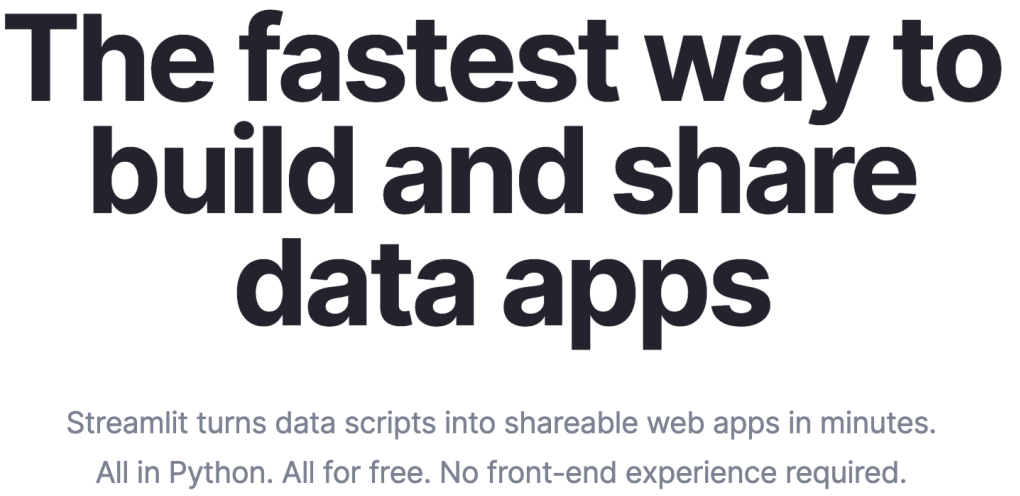
- 人間（ベテラン）の作った解の10%程度の削減
- 物流費用はGDPの8%程度（途上国ではその倍以上）
- 例えば日本だとGDPは560兆円程度

最適化を使えば，物流費だけで年間4兆円削減可能！

サプライ・チェインの費用はさらに大きい！

Streamlit

- https://www.streamlit.io/
- Pythonだけで簡単にWebアプリ
- プログラムを上から実行していくだけ
  （コールバック不要）
- 関数をキャッシュすることによる高速化（デコレータ使用）
- github経由で簡単にデプロイ（無料）



こういうのが簡単に作れる（DLで予測）

アプリ選択 (Select an App)

Deep Forecast

Select Language

日本語

予測する顧客名と製品名を選択

製品選択

A

顧客選択

仙台市

予測パラメータ

期間

1D

予測期間数

1 − +

追加データ選択

promo_0 × promo_1 ×

検証パラメータ

検証開始期

2020/10/19

☑ 予測データ生成

深層学習

☑ 学習器生成

Deep Forecast

Sample excel file

Excelファイルアップロード

Drag and drop file here
Limit 200MB per file • XLSX
Browse files

需要データ

需要データアップロード

Drag and drop file here
Limit 200MB per file • CSV, TXT
Browse files

date	cust	prod	promo_0	promo_1	demand
2019-01-01	札幌市	A	0	0	0
2019-01-01	札幌市	B	0	0	1
2019-01-01	札幌市	C	0	0	0
2019-01-01	青森市	A	0	0	2
2019-01-01	青森市	B	0	0	14

1 to 5 of 21,900 Page 1 of 4,380

Download csv file

☐ データ探索

プロモーションデータ

プロモーションデータアップロード

Drag and drop file here
Limit 200MB per file • CSV, TXT
Browse files

date	promo_0	promo_1
2019-01-01	0	0
2019-01-02	1	0
2019-01-03	0	0
2019-01-04	0	0
2019-01-05	0	0

1 to 5 of 1,095 Page 1 of 219

Download csv file

☐ データ探索

demand	promo_0	Week	promo_1	Year	Month	Day	Dayofweek	Dayofyear	Is_month_end	Is_month_start	Is_quarter_end	Is_quarter_start	Is_year_end	Is_year_start	El
13	0	1	0	2019	1	1	1	1	false	true	false	true	false	true	15
19	1	1	0	2019	1	2	2	2	false	false	false	false	false	false	15
17	0	1	0	2019	1	3	3	3	false	false	false	false	false	false	15
12	0	1	0	2019	1	4	4	4	false	false	false	false	false	false	15
19	0	1	0	2019	1	5	5	5	false	false	false	false	false	false	15

1 to 5 of 730 Page 1 of 146

Download csv file

☐ データ探索

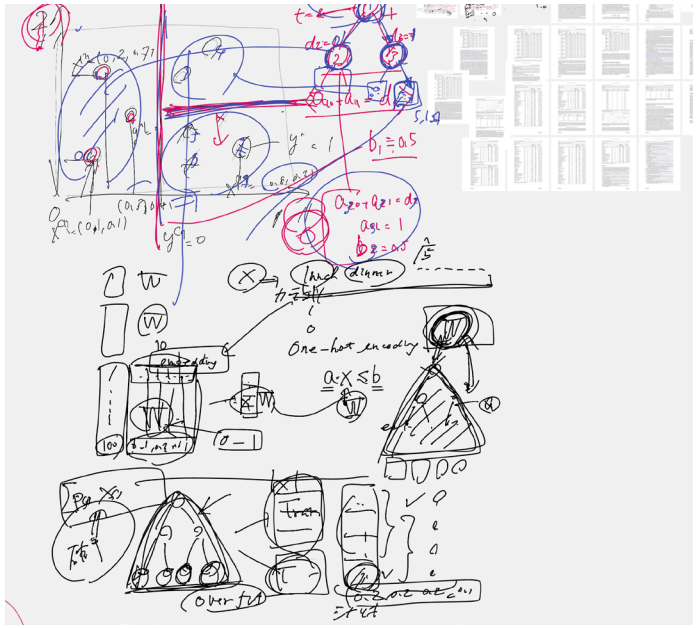
MIRO

The online collaborative whiteboard platform to bring teams together, anytime, anywhere.

Start a whiteboard →

Free forever — no credit card required

- https://miro.com/
- オンライン・ホワイトボード（画面共有）
- コロナ禍でSPRINTを行うためのツール（zoomと併用）
- 授業にも便利（学生参加型の演習）
- アカデミック登録で無制限



menu

Notion

- https://www.notion.so/
- ゼミ学生との研究共有
- 講義の簡易HP

データサイエンス演習

データサイエンス演習

ゼミ生個別ページ

- 💬 コロナ禍の避難計画
- 🏨 深層・機械学習を用いた ホテル予約件数の予測
- 👮 サプライ・チェイン途絶条件下でのリスク分析
- AGV 最適パス設計のための最適化手法
- 時間枠付きTSPの実験

研究室ゼミ用

- SCMOPT-zero間の連携システム
- 実務における大規模配車計画問題
- SCIPとPuLPの比較実験
- 賞味期限を考慮した動的価格・在庫最適化
- ガスボンベの配送計画問題
- pyvisを用いたOptSeqの時間制約の可視化
- 勤務表作成アプリ
- 回転を考慮した配送最適化
- scmopt0 on GCP
- ✏️ SCMOPTにValidationを追加し，fastapiでREST APIを作る
- サービスネットワーク設計をSCOPで解く
- 予測システムの比較
- 複数の製品・顧客に対して予測するML Ops
- 大規模グラフに対する時刻依存最短路
- 二次割当問題
- 時間枠付き巡回セールスマン問題
- 🚕 OptSeqの実験的解析
- 階層的巡回セールスマン問題
- 動的確率的ビンパッキング問題
- 分割配送問題に対する近似解法
- 🔍 Interpretable AI

データサイエンス演習

アナリティクス（データサイエンス）練習問題集

データを入手したとき，最初に行うことはデータの概要を把握することである．これをデータの探索(exploration

https://mikiokubo.github.io/analytics/

環境整備

【初心者向け】PythonとJupyterの環境構築手順2021年版(Windows編) - Qiita

本記事はWindowsユーザーが対象となります デフォルトの状態から環境構築していきます。コマンドプロンプトをインストールしてJupyter ...

https://qiita.com/Nate0928/items/d4b8cc091f7f1510f3c8

【初心者向け】Pythonとjupyterの環境構築手順2021年版（Mac編） - Qiita

本記事はMacユーザーが対象となりますが、M1搭載のMacについては未だ未知数なことも多いため対象外です。す。 Mac上にPythonでのデータ分析環境を整えることを目的とします。 ...

https://qiita.com/Nate0928/items/40bc828fa4ef28fa560e

Streamlit + github

[簡単爆速]Pythonだけでデプロイ。Streamlit1.0の使い方を紹介 - Qiita

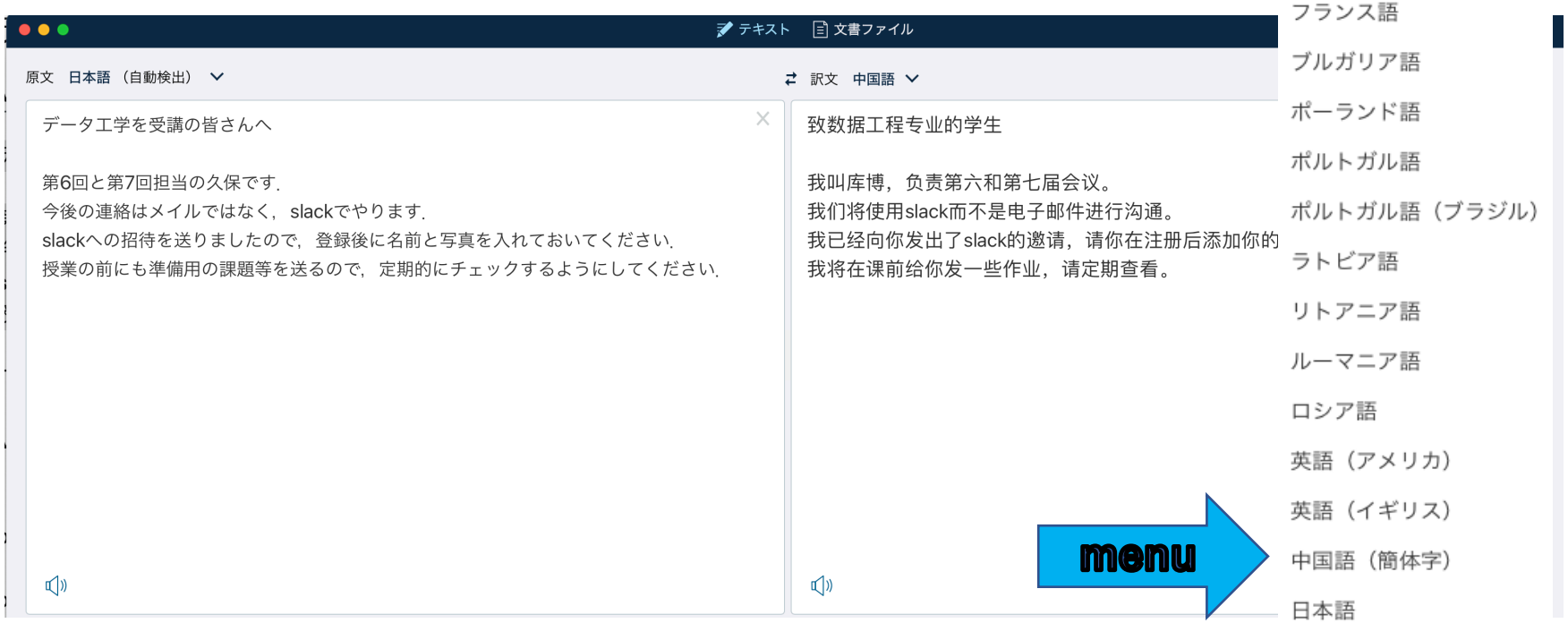
データ分析の結果や研究成果を公開するとき、多くの方に使ってもらえるように、何かしらツール化したほうがいムワークです。 ...

https://qiita.com/Nate0928/items/0b192c903a41d75cb7c6

menu

DeepL

- https://www.deepl.com/
- 深層学習による翻訳
- 読み上げ機能
- 無料アプリあり



menu

Slack

- https://slack.com/
- 授業やゼミの連絡や質問（チャット）
- リアルタイムに関係者全員に（頻繁に）連絡可能

SPRINT

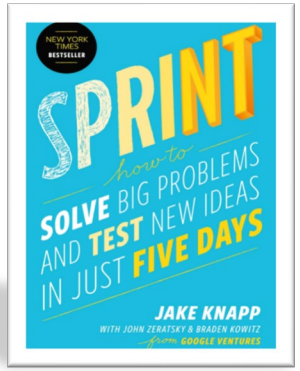
Noriko Yoshida

Tokyo university of marine science and technology

SPRINTとは

今までにない革新的な問題発見と解決の方法

SPRINTの始まりと現在



2010年

- **Googleでスタート**
  - GVデザインパートナー Jake Knappが開発

2012年

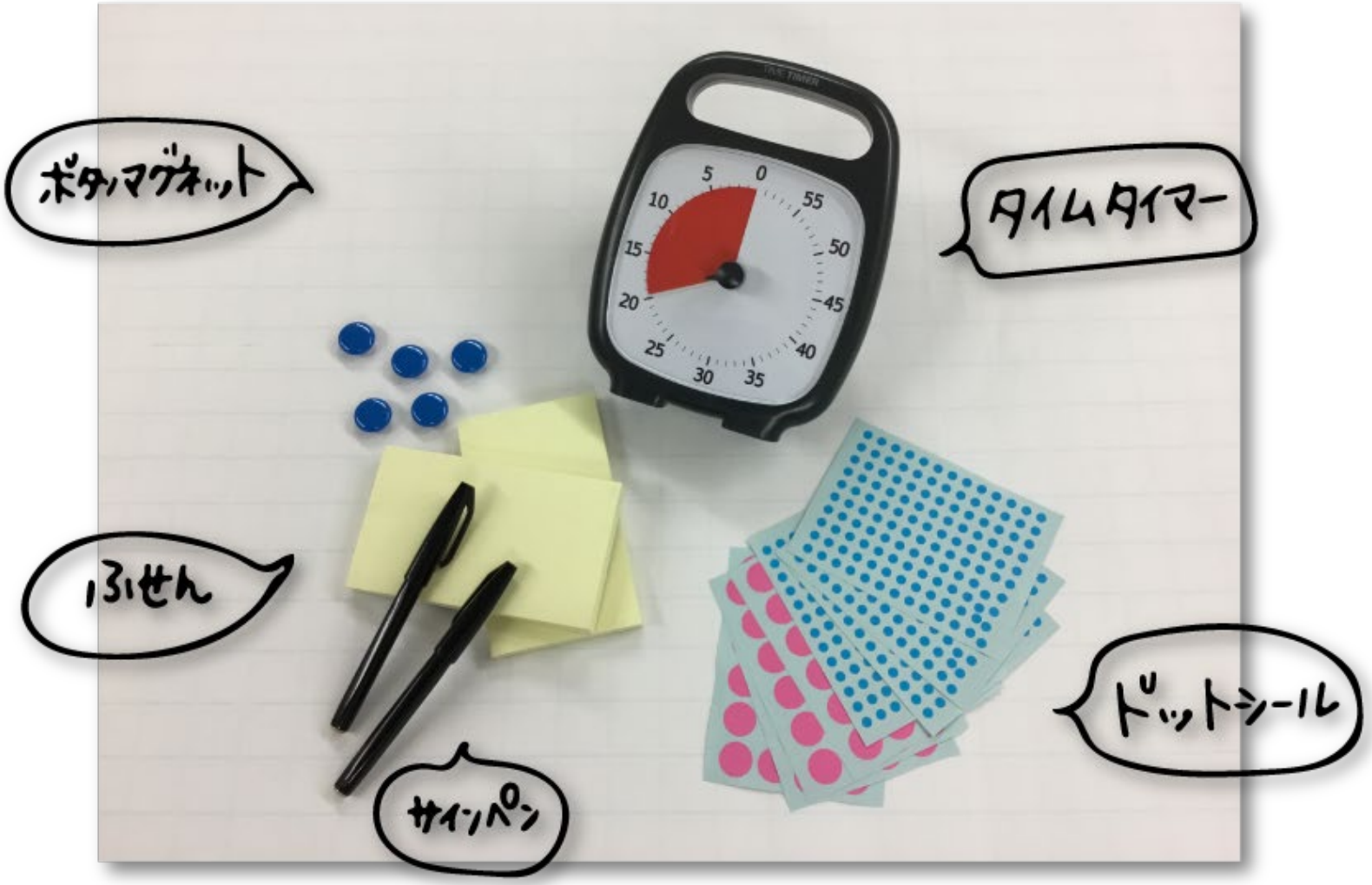
- GVで用いられる
  - スタートアップ企業が活用

2017年

- 日本で出版
  - 2017年4月

使う道具が決まっている

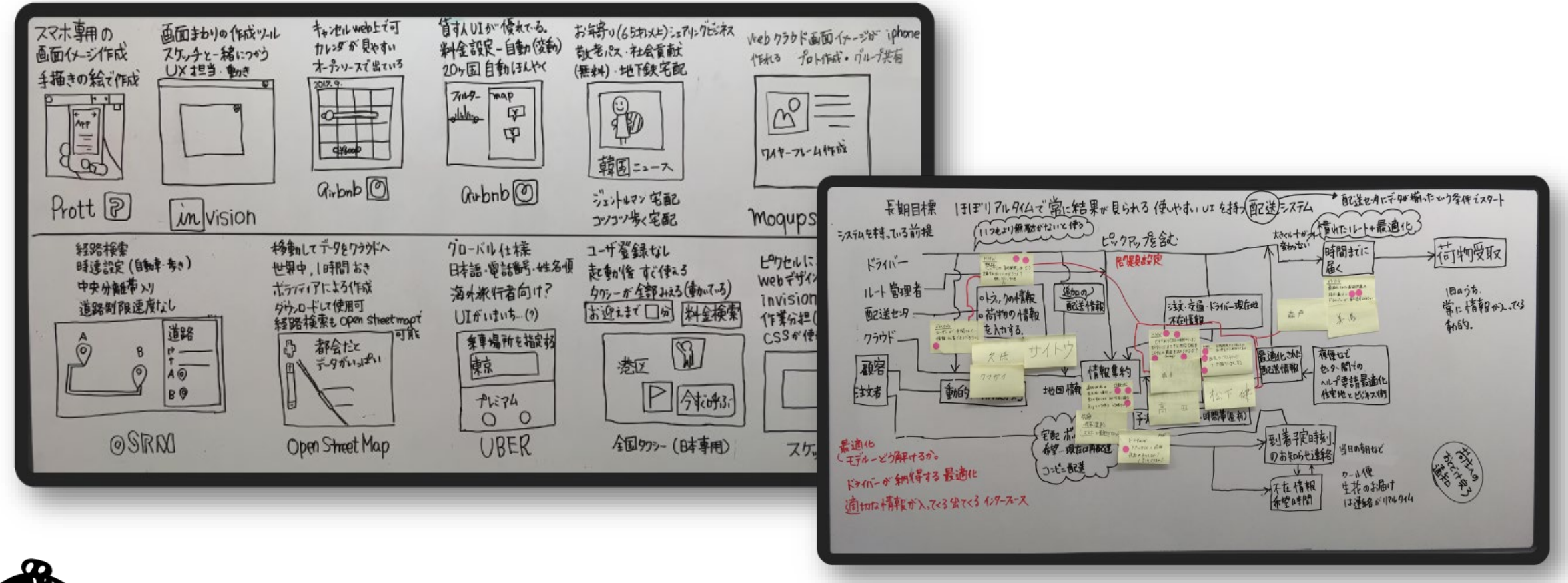
SPRINTの工夫 その③



あとはコーヒー  とヘルシーなスナック  も！

ホワイトボードを2台使用する

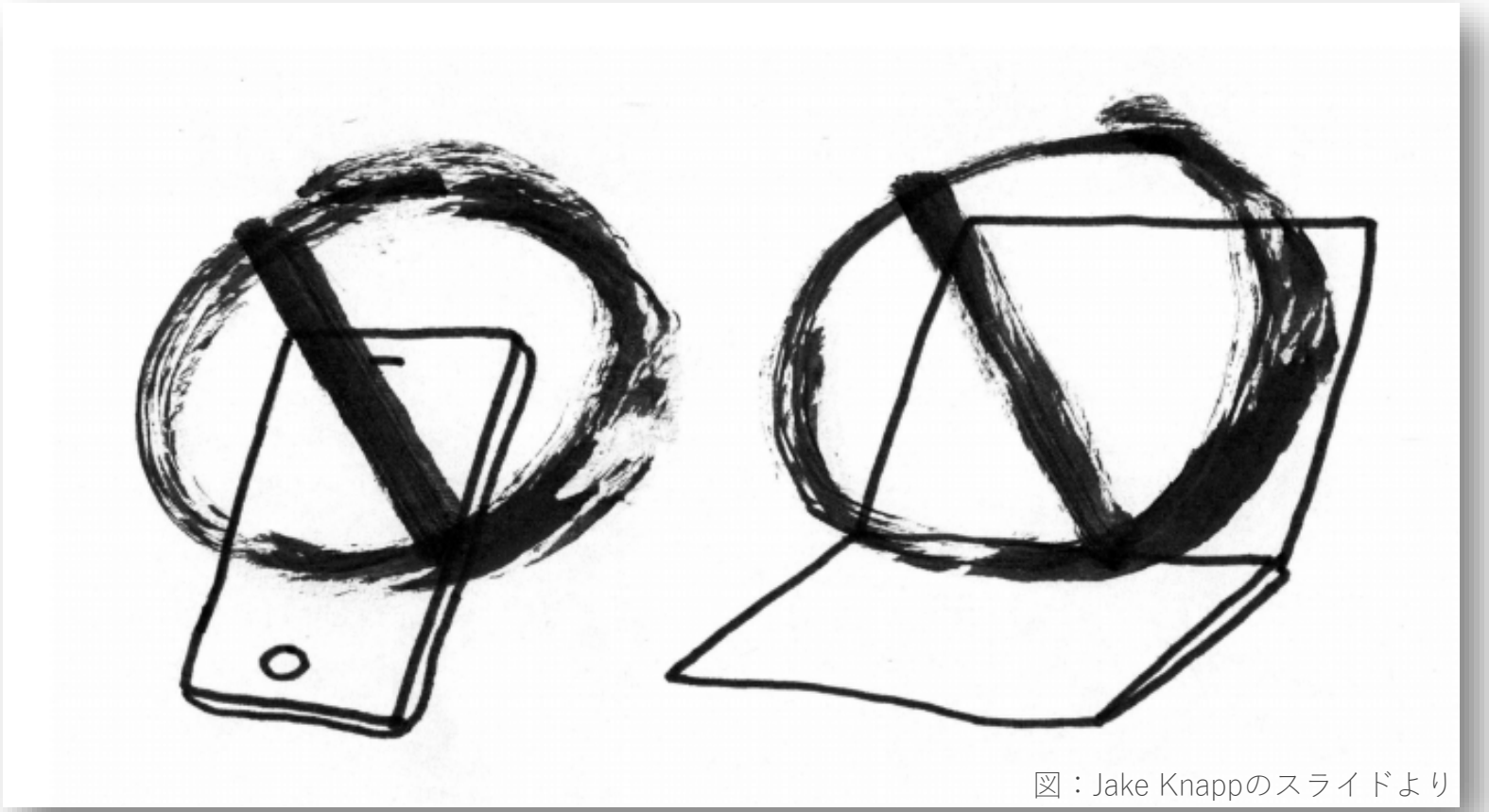
SPRINTの工夫 その④



メンバーの知識と知恵は全部ここに記録し、ホワイトボードが共通の頭脳となる

SPRINT中はデバイス使用禁止

SPRINTの工夫 その⑤



図：Jake Knappのスライドより

気が散るものは部屋の外で使ってね。休憩時間は自由に使用できます！

SPRINTのプロセス

1. 問題の神髄を見極める
2. 出来るだけ多くの解決策を考えてみる
3. いくつかのアイデアを選び、研ぎ澄ます
4. 最終的なアイデアに絞り込む
5. プロトタイプの作成
6. 結果を知って、学習する

5日間のプロセスが決まっている

menu

図：日本語版「SPRINT」（ダイヤモンド社）より

MIROのSPRINTテンプレート