

# GCI 2024 Summer

## Week5 教師あり学習

---



松尾・岩澤研究室

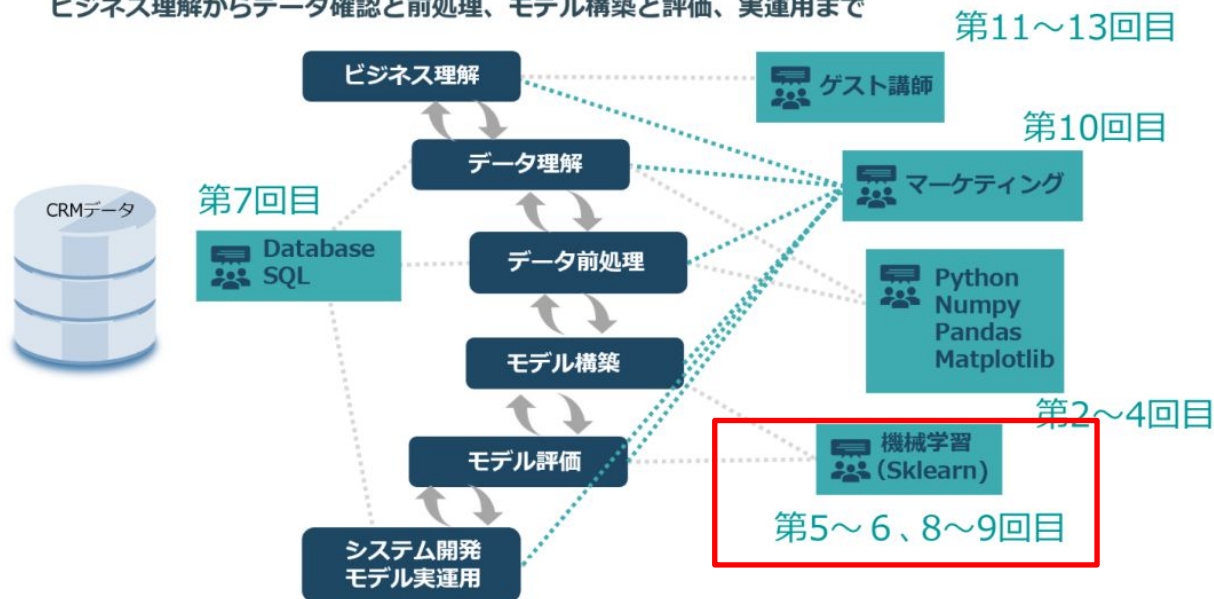
MATSUO-IWASAWA LAB UTOKYO

作成者・2022年講師: 近藤 佑樹  
編集・講師: 石田 将貴 / 福地 清康  
発表者: 福地 清康

- 今日からモデル構築・評価について学びます

## 実データサイエンスのプロジェクトと本講義の関係性

ビジネス理解からデータ確認と前処理、モデル構築と評価、実運用まで



GCI 2023 Summer week1より引用

- 機械学習の概要を理解できる
  - 機械学習は何をやっているのか理解する
  - 機械学習の大分類を理解できる
  -
- 教師あり機械学習モデルの概略が理解できる
  - 回帰と分類について理解する
  - 5つのモデルがどのように動いているのか
  - 5つのモデルにはどのような特徴があるか?
  - 5つの機械学習モデルをSckit-learnで実装できる

機械学習:

データから知識やパターンを理解させるアルゴリズム

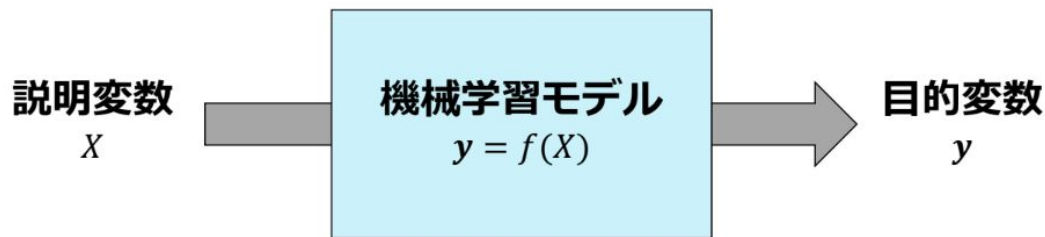
## 教師あり学習の用語と概観

○説明変数(特徴量):

機械学習モデルへの入力に用いられる変数

○目的変数:

機械学習モデルの出力として定められる変数



## 教師あり学習

- ・ 正解データが定められた学習法
- ・ 入力データと正解データの関係性を関数として近似



• Age  
• Sex  
• Ticket

→

Survived  
or  
Died

タイタニック号の生存者予測



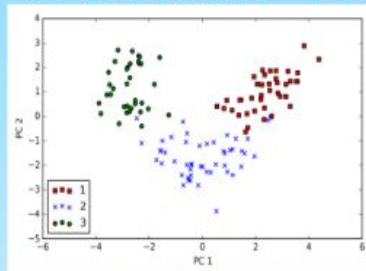
• Kite  
• Sea snake  
• Siberian husky  
• Drake  
•  
•

画像分類 (ImageNet)

引用: J. Deng+ CVPR2009

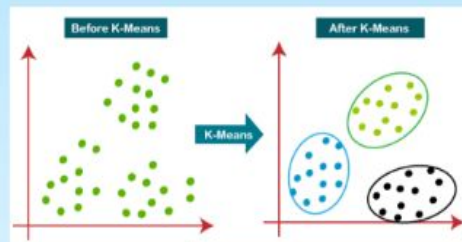
## 教師なし学習

- ・ 正解データが定められていない学習法
- ・ データの潜在的なパターンを学習



主成分分析

引用: [https://github.com/rasbt/python-machine-learning-book/blob/master/code/ch05/images/05\\_03.png](https://github.com/rasbt/python-machine-learning-book/blob/master/code/ch05/images/05_03.png)

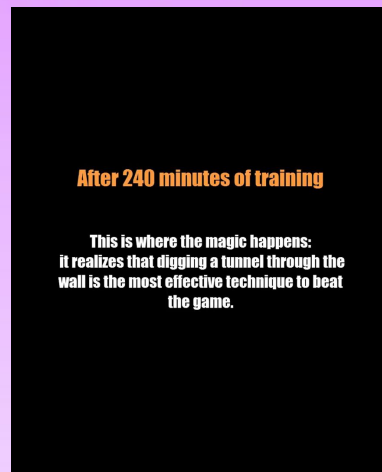


K-means

引用: <https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning>

## 強化学習

- ・ 報酬を最適化させる行動・知識を学習する方法
- ・ ロボット制御やゲーム等で応用されている



DQNを用いたAtariのプレイ動画

引用: <https://www.youtube.com/watch?v=V1eYnU0Rnk>

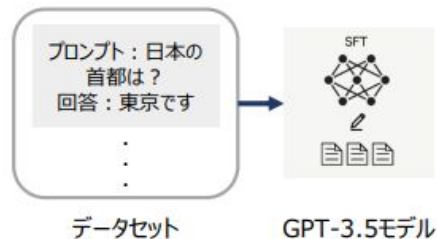
\* GCIでは割愛

## ChatGPTの学習方法

ChatGPTの学習は以下の3つのステップで構成されている

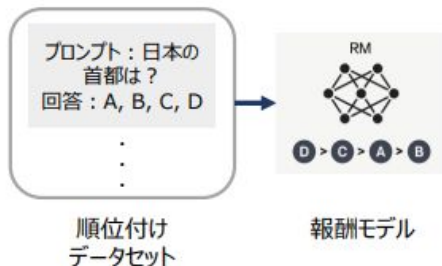
### Step 1: 教師あり学習

- プロンプトとそれに対する適切な回答のペアをアノテーター（人間）が考案し、データセットを作成する
- このデータセットを用いてGPT-3.5モデルをファインチューニングする



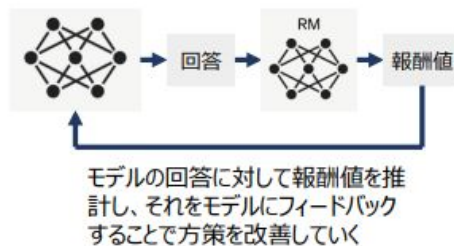
### Step 2: 報酬モデルの学習

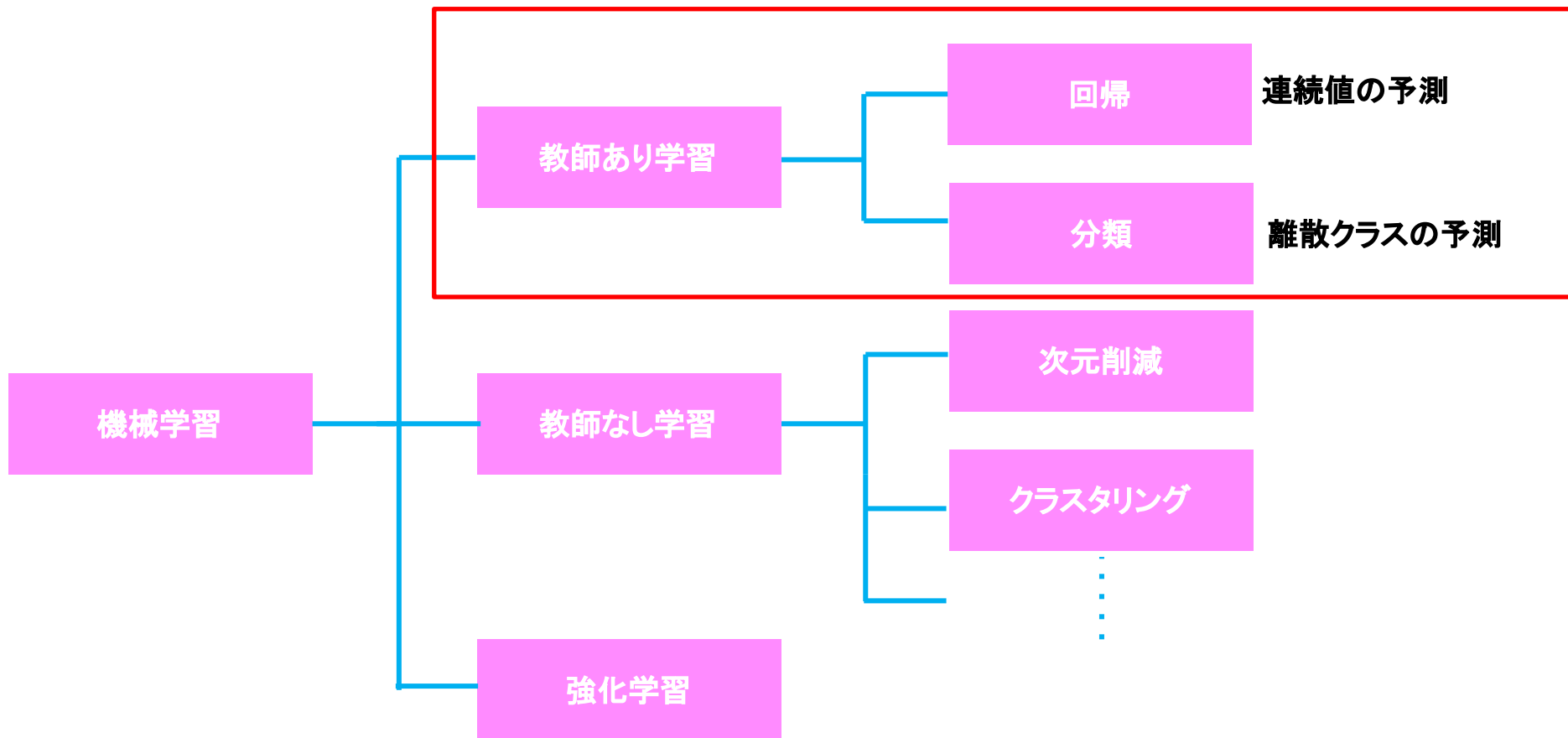
- プロンプトに対するstep1で学習させたモデルの回答を複数パターン用意し、アノテーターにその中で良いものはどれかの順位付けをしてもらう
- 順位付けデータセットを用いて報酬モデルを学習させる
  - 回答の順位付けを予測するタスクを解かせる



### Step 3: 強化学習

- Step1/2で学習させたGPT-3.5モデルと報酬モデルを用いて、強化学習を実施する
  - 報酬が最大になるような方策を探索し、最適な回答を生成する

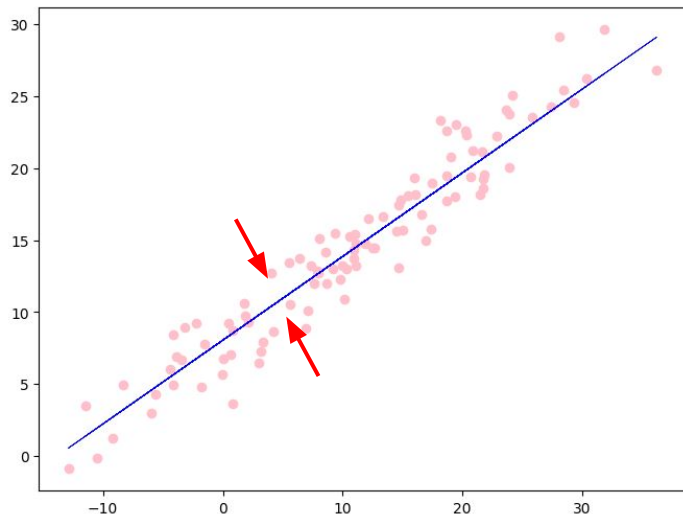




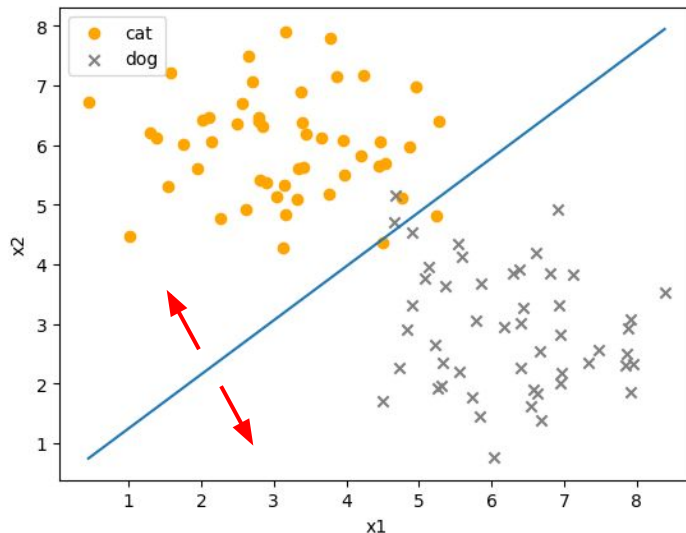
教師あり学習には回帰と分類の二種類がある

- 回帰: 学習データから連続値の予測を行う
- 分類: 設計した分類にデータを振り分ける

回帰の例:  
気温から水温を予測する



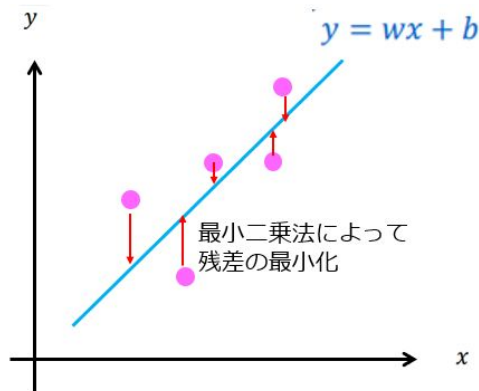
分類の例:  
犬と猫の画像を分類



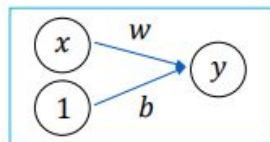


- モデル概要: 目的変数を複数の説明変数の重み付き和で予測する
- 用途: 回帰
- 統計に基づくシンプルなモデルなので根拠が明確
- 外れ値に弱い

## 単回帰分析



一つの説明変数と誤差

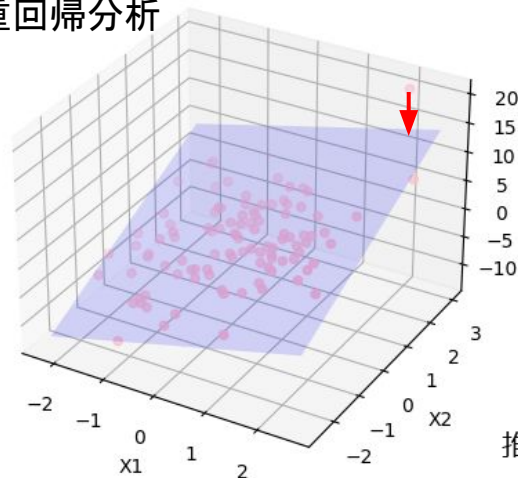


推定パラメータ  $w, b$

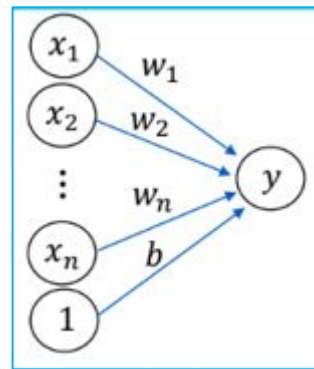
最小二乗法で残差を最小化するような **直線**を求める

$$f(\mathbf{x}) = w_1x_1 + b$$

## 重回帰分析



複数の説明変数と誤差

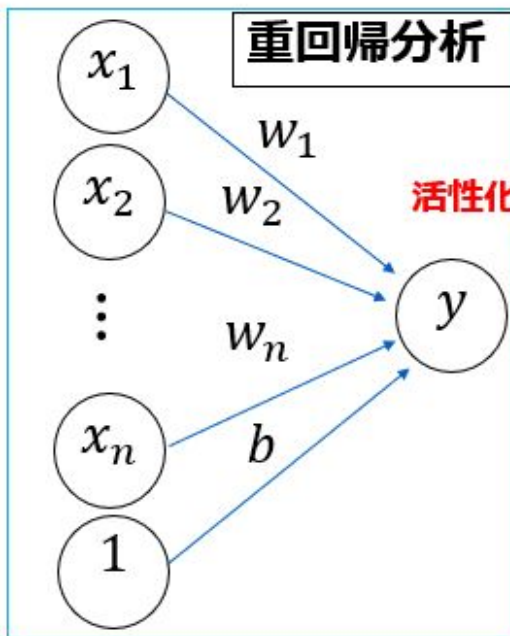


推定パラメータ  $w_1, w_2, \dots, w_n, b$

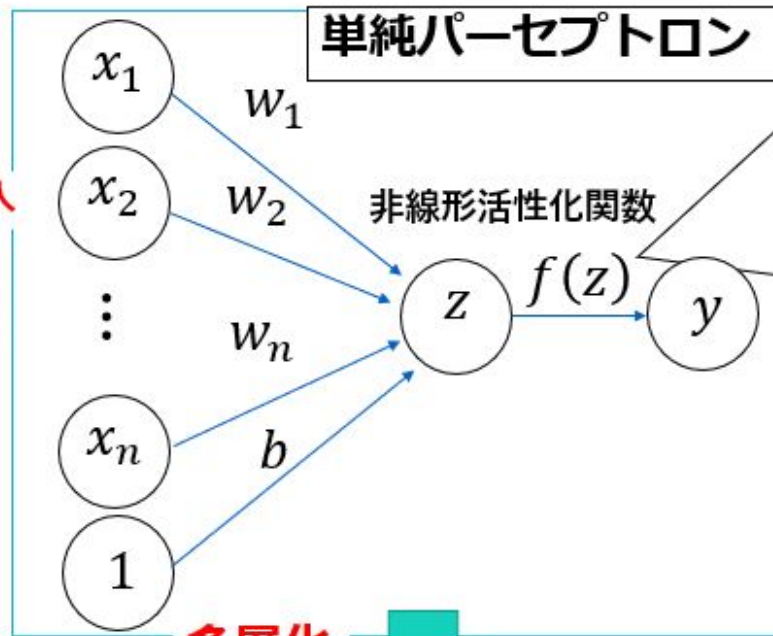
最小二乗法で残差を最小化するような **平面**を求める

$$f(\mathbf{x}) = w_1x_1 + w_2x_2 + w_3x_3 + \dots + b$$

# 重回帰分析と深層学習



活性化関数を導入

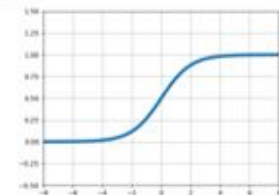


非線形活性化関数

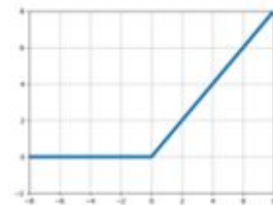
多層化



**多層パーセプトロン  
(深層学習モデル)**



シグモイド関数  
(このモデルを  
ロジスティック回帰と呼ぶ)



ReLU

etc. ...

- ・目的変数と説明変数に分ける
- ・訓練データとテストデータに分ける

データ収集

EDA・前処理

```
# 目的変数にpriceを指定、説明変数にそれ以外を指定
```

```
X = auto.drop('price', axis=1)
```

```
y = auto['price']
```

```
# 訓練データとテストデータに分ける
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
```

訓練データ

テストデータ

```
# 重回帰クラスの初期化と学習  
model = LinearRegression()
```

モデル選択・構築

モデル学習

推論

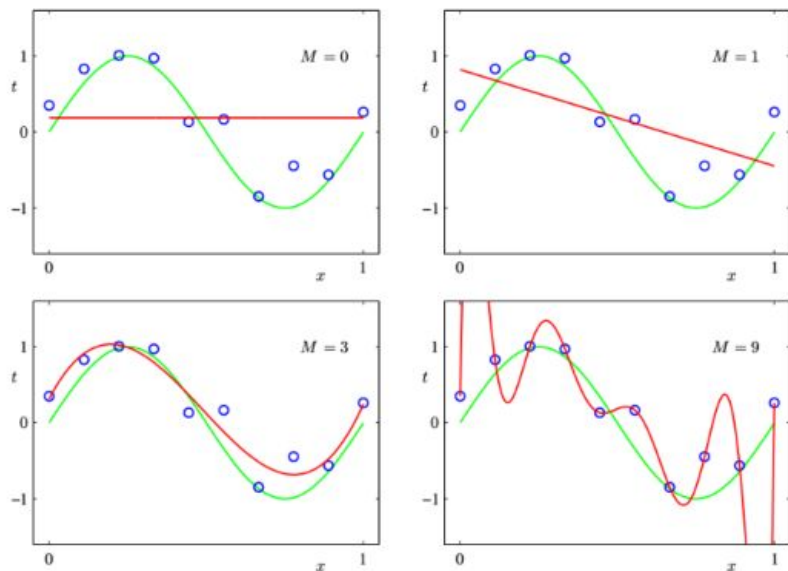
```
model.fit(X_train, y_train)
```

```
y_pred = model.predict(X_test)
```



notebook 

学習データに過剰にフィットすることで、  
モデル化した関数が真のデータの関数から  
離れてしまう状態

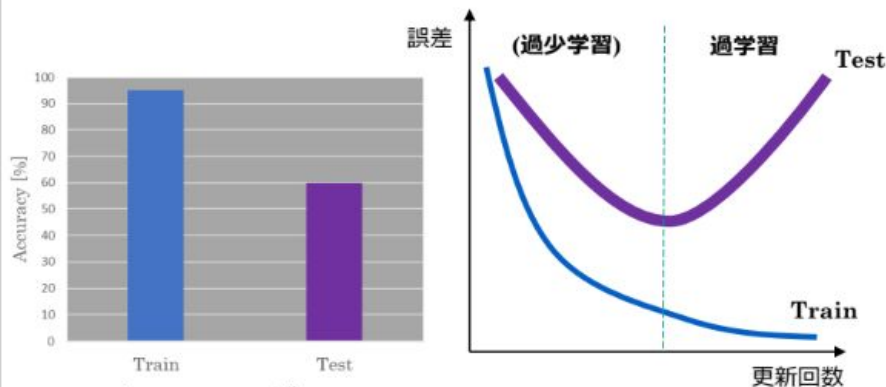


$M=0, 1$ ではデータの関数の表現力が不足しており、 $M=3$ ではうまく表現できている。 $M=9$ では過学習が発生している。

引用 : C. Bishop "Pattern Recognition and Machine Learning".

確認方法 :

学習データのスコアに対し、テストデータのスコアが大きく低下する場合、  
過学習しているとみなす。



テストでスコアが35%も低下しているため、過学習していると言える。

逐次パラメータを更新する場合、  
モデルが更新するほど、学習  
データにフィットするため、過  
度な学習は過学習につながる。

- 正則化とは？
  - 複雑になったモデルをシンプルにすることで過学習を
  - 解決するという手法
  - 目的変数に寄与しない変数の係数を小さくして実質的な
  - 説明変数の数を減らし、過学習を抑制する効果

【**正則化項**が加えられた損失関数  $L$ 】

$$L = \underbrace{\sum_{i=1}^n (y_i - f(x_i))^2}_{\text{二乗誤差関数}} + \underbrace{R(\mathbf{w})}_{\text{正則化項}}$$

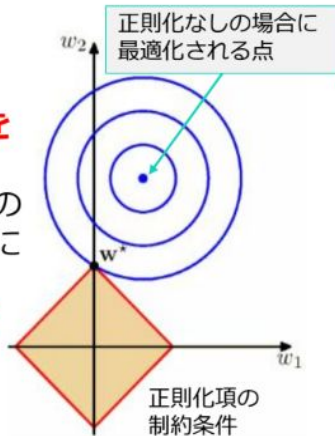
# 正則化: ラッソ回帰・リッジ回帰

- ラッソ回帰(L1正則化): 不要な特徴量の削除
- リッジ回帰(L2正則化): 大きな特徴量にペナルティ→小さくする

## ラッソ(LASSO)回帰

$$L = \sum_{i=1}^n (y_i - f(x_i))^2 + \lambda \sum_{j=1}^m |w_j|$$

- ・ 正則化項によって,  
 **$w^*$ に最適化.**
- ・ **いくつかのパラメータを0にする作用**  
➡ 実質的に元のデータの説明変数を除くことに相当  
(**スパースモデリング**という)

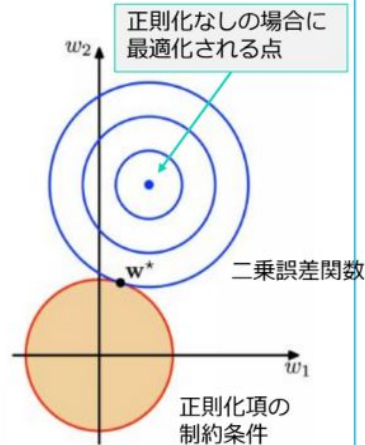


引用: C. Bishop "Pattern Recognition and Machine Learning".

## リッジ(Ridge)回帰

$$L = \sum_{i=1}^n (y_i - f(x_i))^2 + \lambda \sum_{j=1}^m w_j^2$$

- ・ 正則化項によって,  
 **$w^*$ に最適化.**
- ・ パラメータを**全体的に小さくする**作用

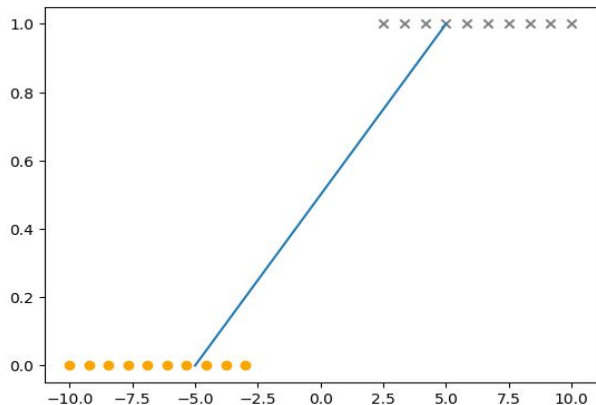


- 余談 -  
リッジ回帰とラッソ回帰の正則化項両方を加えたモデルを **Elastic Net** という。

引用: C. Bishop "Pattern Recognition and Machine Learning".

- モデル概要 : 二値分類を行う回帰分析
- 用途: **分類** (回帰とついているが注意!)
- 統計に基づくシンプルなモデルなので説明性が高い
- 精度は後述のモデルよりも劣るケースが多い

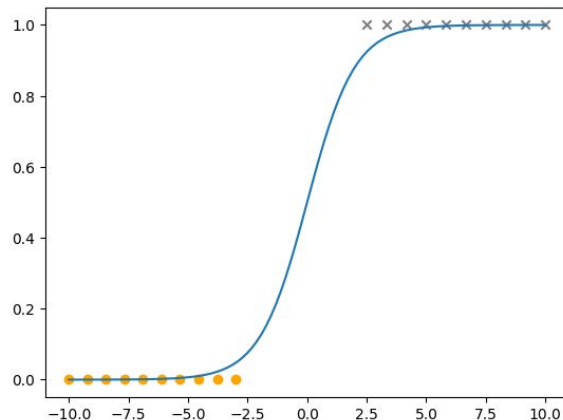
2クラス分類の場合、  
単純に線形回帰では予測できない



シグモイド関数を導入



0,1のどちらを取るかという確率にすること  
でうまく予測できる



重回帰の式をシグモイド関数に入れる

$$f(\mathbf{x}) = w_1x_1 + w_2x_2 + w_3x_3 + \dots + b$$



$$f(\mathbf{x}) = \frac{1}{1 + e^{-(w_1x_{i1} + w_2x_{i2} + \dots + w_kx_{ik})}}$$





notebook 

- ・目的変数と説明変数に分ける
- ・訓練データとテストデータに分ける

データ収集

EDA・前処理

```
# 目的変数にpriceを指定、説明変数にそれ以外を指定
```

```
X = auto.drop('price', axis=1)
```

```
y = auto['price']
```

```
# 訓練データとテストデータに分ける
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
```

訓練データ

テストデータ

```
# 重回帰クラスの初期化と学習  
model = LinearRegression()
```

モデル選択・構築

モデル学習

推論

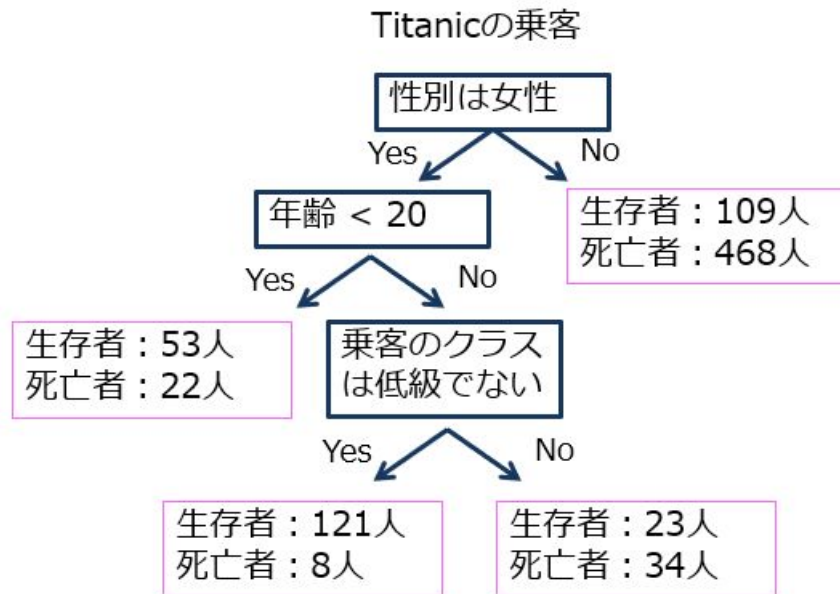
```
model.fit(X_train, y_train)
```

```
y_pred = model.predict(X_test)
```

- モデル概要: ノード・枝・葉からなる木構造で予測する
- 用途: 回帰・分類
- 分類の仕方がわかりやすく説明性が高い
- 過学習になりやすい

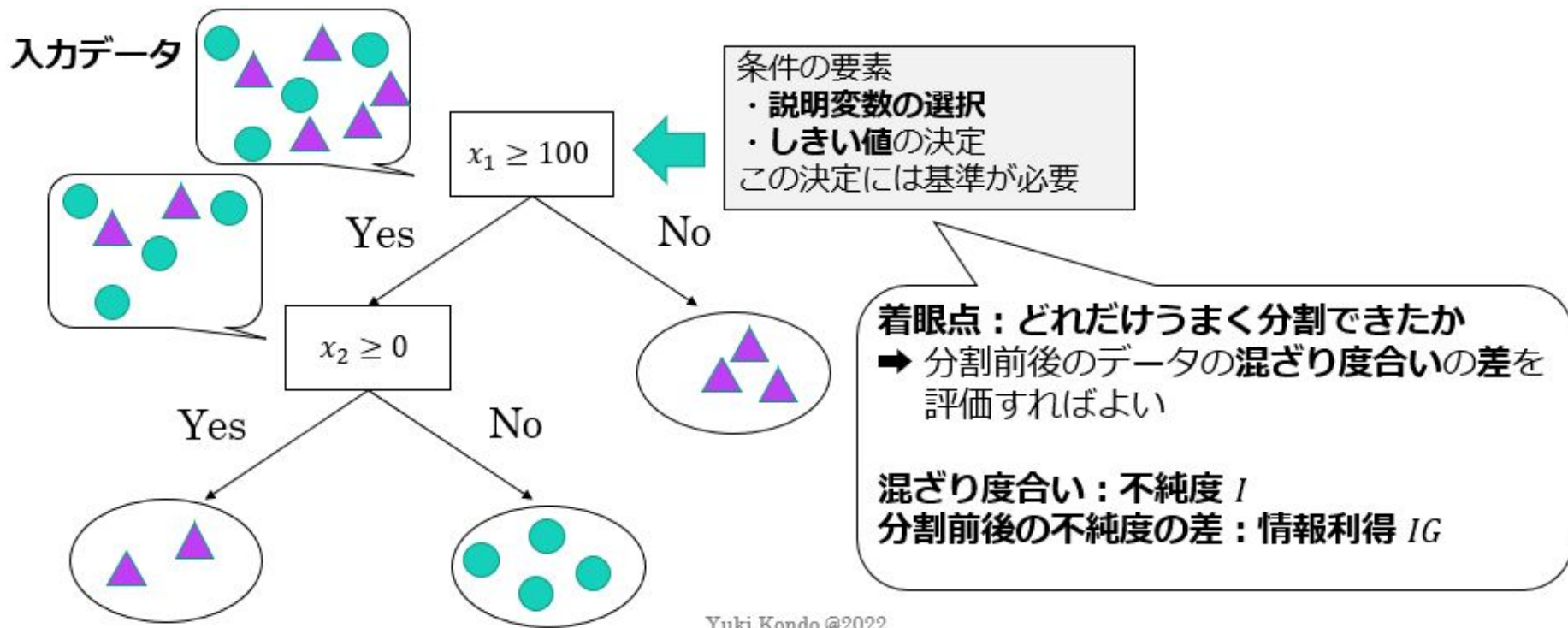
## Titanicでの分類(例)

1. 生存・死亡を分ける上で重要な特徴量は性別である
2. 女の子は生き残りやすい
3. 大人の女性の場合、乗客のクラスが高いと生き残りやすいなどの情報が視覚的に分かる



# 決定木の動き

- ノードごとに一つの説明変数に注目し、データに質問して分ける
- 不純度と情報利得でデータをうまく分割できたか評価する

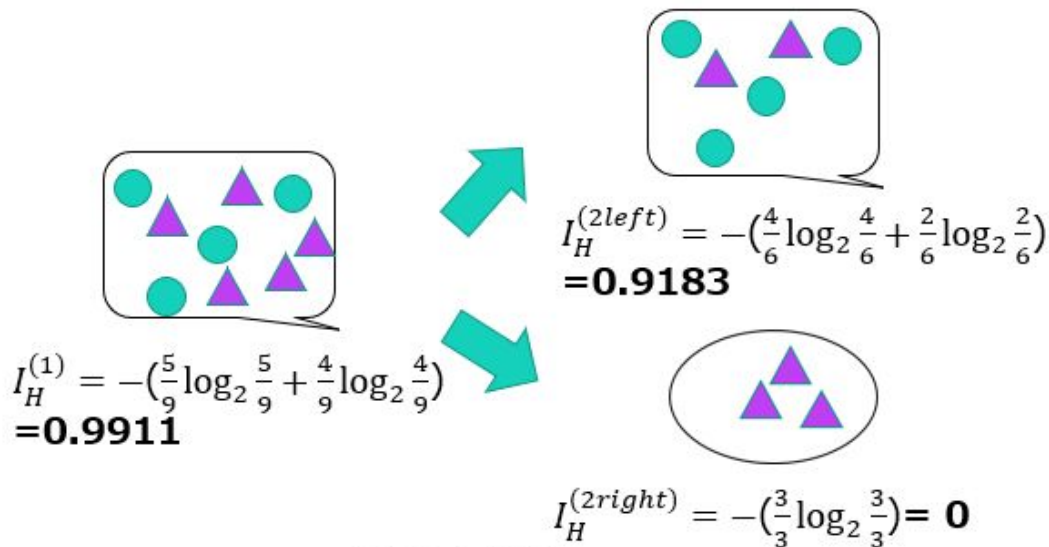


# 不純度の計算

## 【不純度】

- 複数種類ある
  - (シャノン)エントロピー:  $I_H = -\sum_{i=1}^C p_i \log_2 p_i$
  - ジ二不純度:  $I_G = 1 - \sum_{i=1}^C p_i^2$
  - 分類誤差:  $I_E = 1 - \max_{i=1, \dots, C} (p_i)$

## 【計算例】



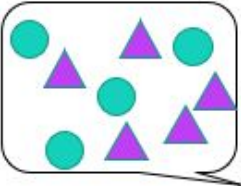
# 情報利得の計算

## 【情報利得】

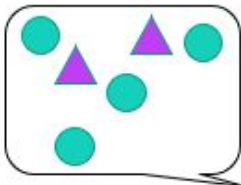
- 2分決定木の場合


- $$IG = I(p) - \frac{N_{left}}{N_p} I^{(left)} - \frac{N_{right}}{N_p} I^{(right)}$$

## 【計算例】


$$I_H^{(1)} = -\left(\frac{5}{9} \log_2 \frac{5}{9} + \frac{4}{9} \log_2 \frac{4}{9}\right) = \mathbf{0.9911}$$




$$I_H^{(2left)} = -\left(\frac{4}{6} \log_2 \frac{4}{6} + \frac{2}{6} \log_2 \frac{2}{6}\right) = \mathbf{0.9183}$$


$$I_H^{(2right)} = -\left(\frac{3}{3} \log_2 \frac{3}{3}\right) = \mathbf{0}$$

左図より

$$\begin{aligned} IG &= I_H^{(1)} - \frac{N_{left}}{N_p} I_H^{(2left)} - \frac{N_{right}}{N_p} I_H^{(2right)} \\ &= 0.9911 - \frac{6}{9} \times 0.9183 - \frac{3}{9} \times 0 \\ &= \mathbf{0.3789} \end{aligned}$$

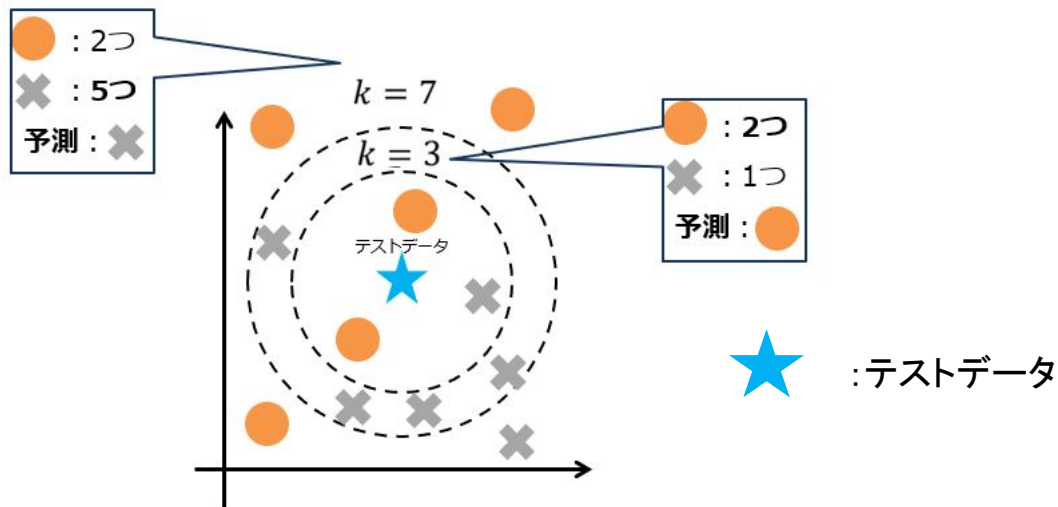
この情報利得を最大化する  
ルールを求める



notebook 

# k-NN (k近傍法)

- モデル概要: 特徴量空間のテストデータ近傍のk個の学習データの正解ラベルから推定する
- 用途: 回帰・分類
- 可視化に優れ解釈が容易
- 学習データが多い場合、学習時間が長い

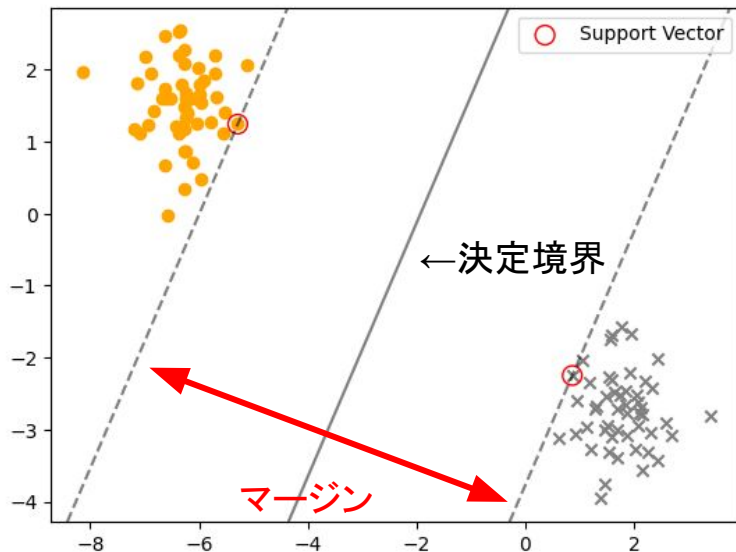




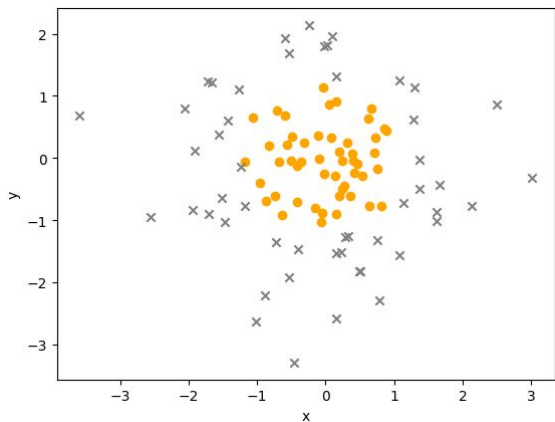


notebook 

- モデル概要: クラス間のマージンを最大化する条件下で決定境界を定めるモデル
- 用途: 回帰・分類
- 精度が高い(あくまで今回紹介した方法の中での傾向)
- 学習データが多い場合、学習時間が長い



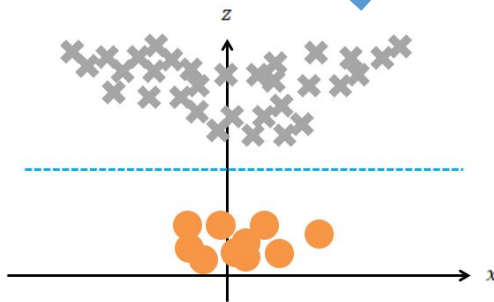
## SVMはカーネルトリックを用いて非線形な決定境界を得ることができる



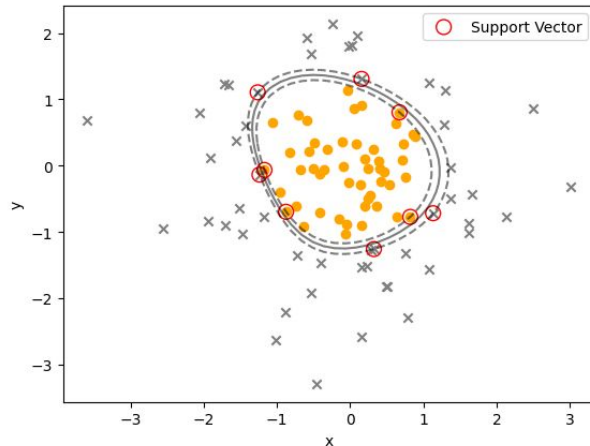
線形分離不可能なケース

カーネルトリックで  
変換

元の空間に決定  
境界を写像



線形分離可能な空間に写像



線形分離不可能なケースでも決定境界を得ることができた

notebook 

# 今日扱った5つのモデルと特徴

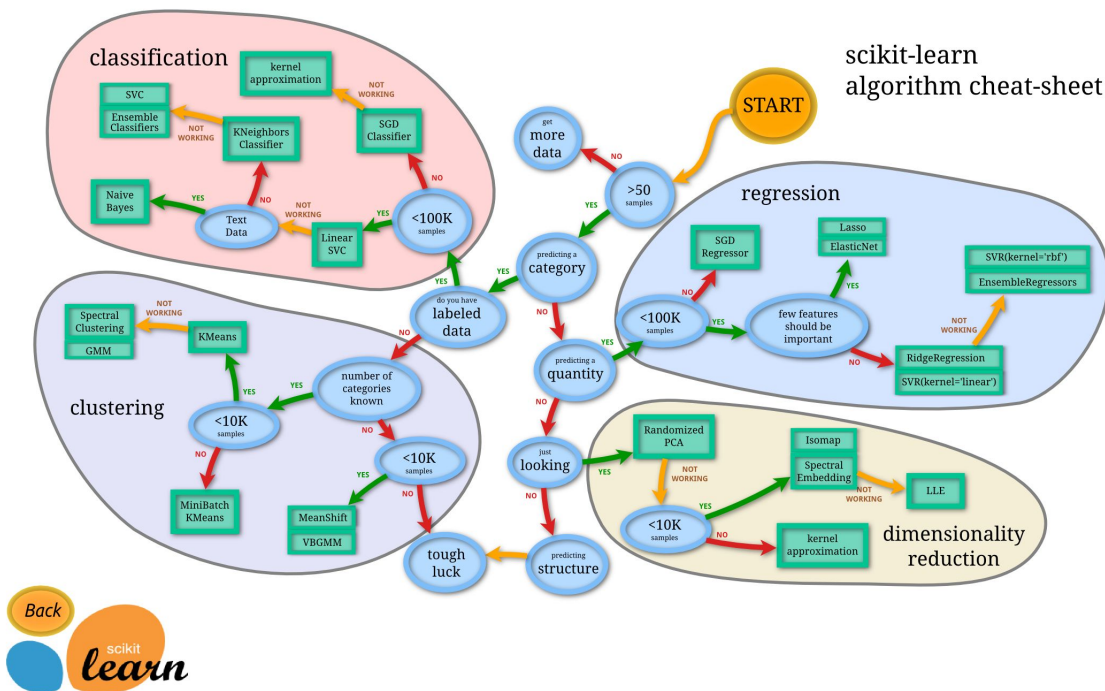


手法	用途	メリット	デメリット
重回帰分析	回帰	シンプルな実装 学習時間が早い	精度がそこまで高くない 場合が多い
ロジスティック回帰	分類	シンプルな実装 学習時間が早い	精度がそこまで高くない 場合が多い
決定木	分類/回帰	モデルの解釈が非常に容易	過学習しやすい 精度は低め
k-NN法	分類/回帰	解釈が容易	データが多いと学習時間がかかる
SVM	分類/回帰	非線形でも分類可能	学習速度が遅い

# どうやってモデルを選択したらいいか？

- データの特性を理解してモデル選択
- チートシートなどを参考にするといいです

[https://scikit-learn.org/stable/tutorial/machine\\_learning\\_map/](https://scikit-learn.org/stable/tutorial/machine_learning_map/)



- ・目的変数と説明変数に分ける
- ・訓練データとテストデータに分ける

データ収集

EDA・前処理

```
# 目的変数にpriceを指定、説明変数にそれ以外を指定
```

```
X = auto.drop('price', axis=1)
```

```
y = auto['price']
```

```
# 訓練データとテストデータに分ける
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)
```

訓練データ

テストデータ

```
# 重回帰クラスの初期化と学習  
model = LinearRegression()
```

モデル選択・構築

モデル学習

推論

```
model.fit(X_train, y_train)
```

```
y_pred = model.predict(X_test)
```