数理生物学演習

第12回 機械学習と深層学習

岩政公平
iwamasa@morphometrics.jp
九州大学大学院システム生命科学府
数理生物学研究室

第12回 機械学習と深層学習

本日の目標

- 機械学習(教師なし学習/教師あり学習)
- 深層学習の応用例

機械学習(Machine Learning)

“Machine Learning is the study of computer algorithms that improve automatically through experience”. (Mitchell, Hill, 1997)

どんなことができるのか？

回帰分析
・広告宣伝費の額によってどれだけ来店者数が増えるか？

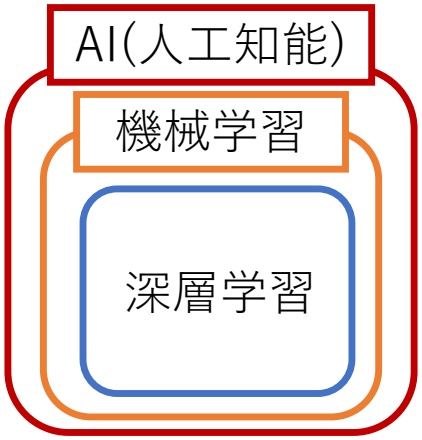
クラス分類
・この画像は犬か猫か？

時系列予測
・グー → チョキ → パー と出したとき次の手は何が出やすいか？

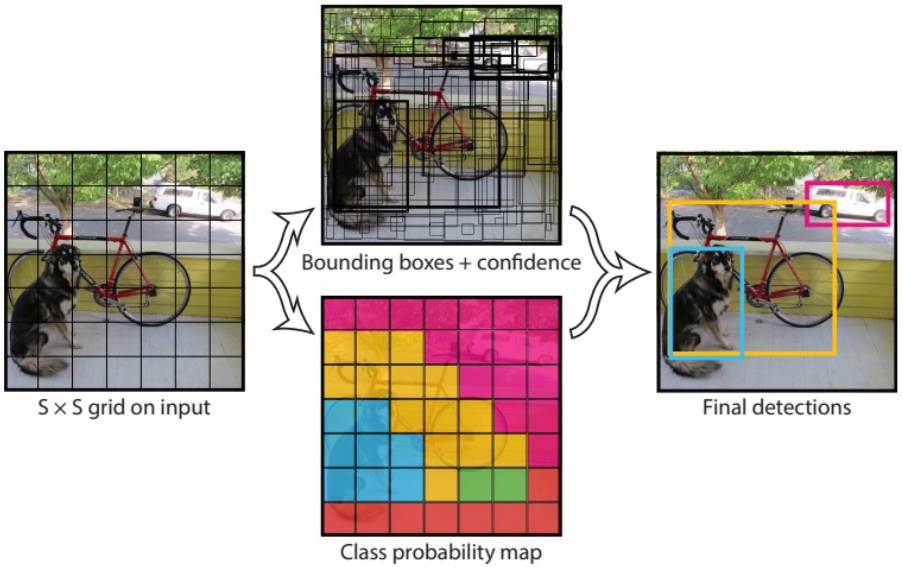
データをもとにどのような傾向があるか
機械が自動的に学習してデータのパターンを見つける手法

深層学習(Deep Learning)

機械学習では人間がデータの特徴を抽出して，
機械学習モデルがその特徴を用いてタスクを解く．
一方，深層学習では自動的に特徴を抽出し，
その特徴を用いてタスクを解く，
「**end-to-endな学習**」を行うことができる．



画像認識



(Redmon et al, 2016)

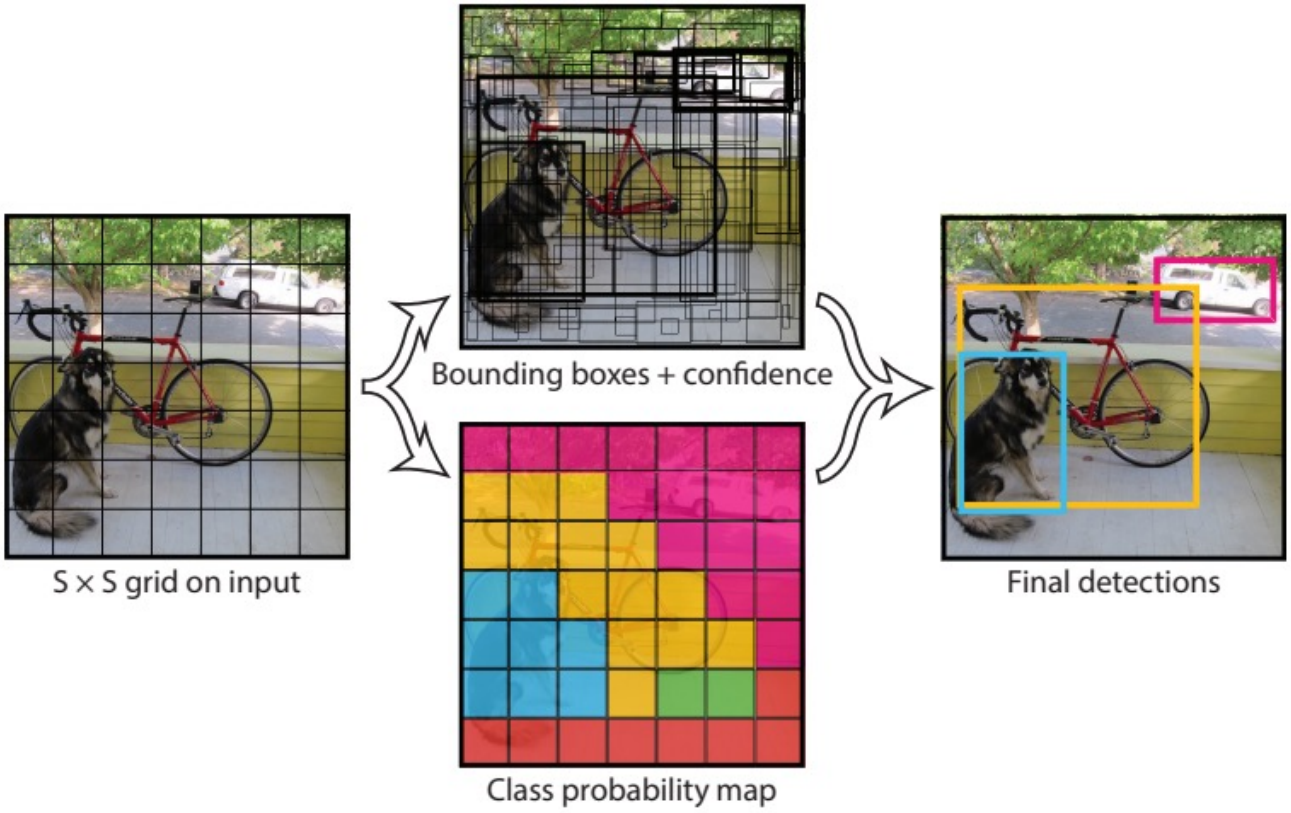
自然言語処理

https://huggingface.co/

深層学習での画像解析

物体検出（Object Detection）

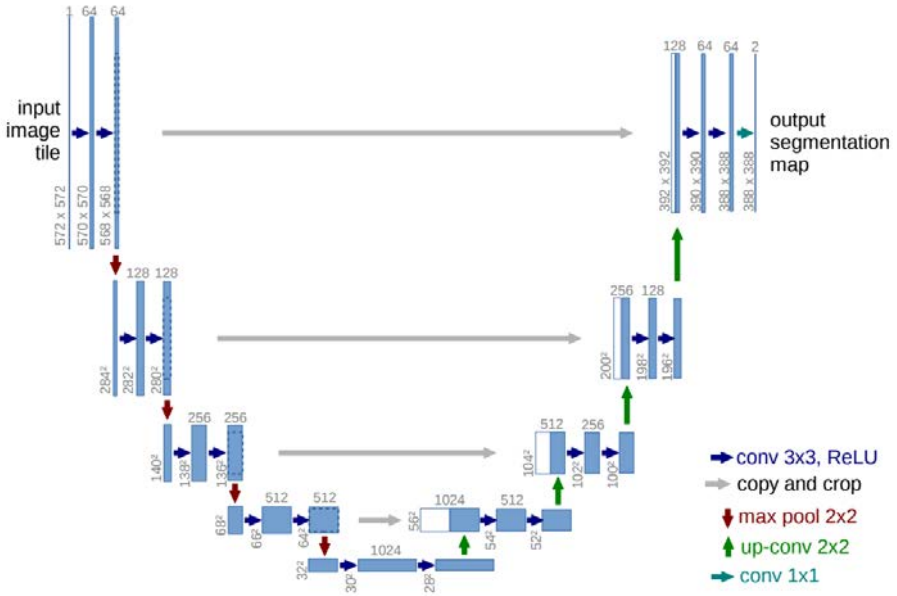
YOLO Redmon et al (2016)
…オブジェクトがどこにあるか矩形で答える

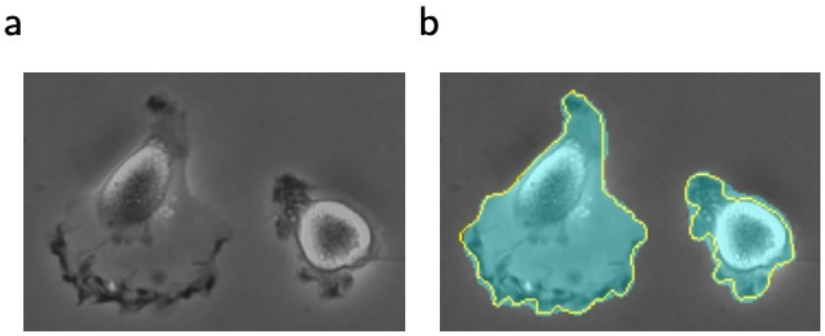


深層学習での画像解析

領域分割 (Semantic Segmentation)

U-Net Ronneberger et al (2015)
…ピクセルレベルでどの物体か認識

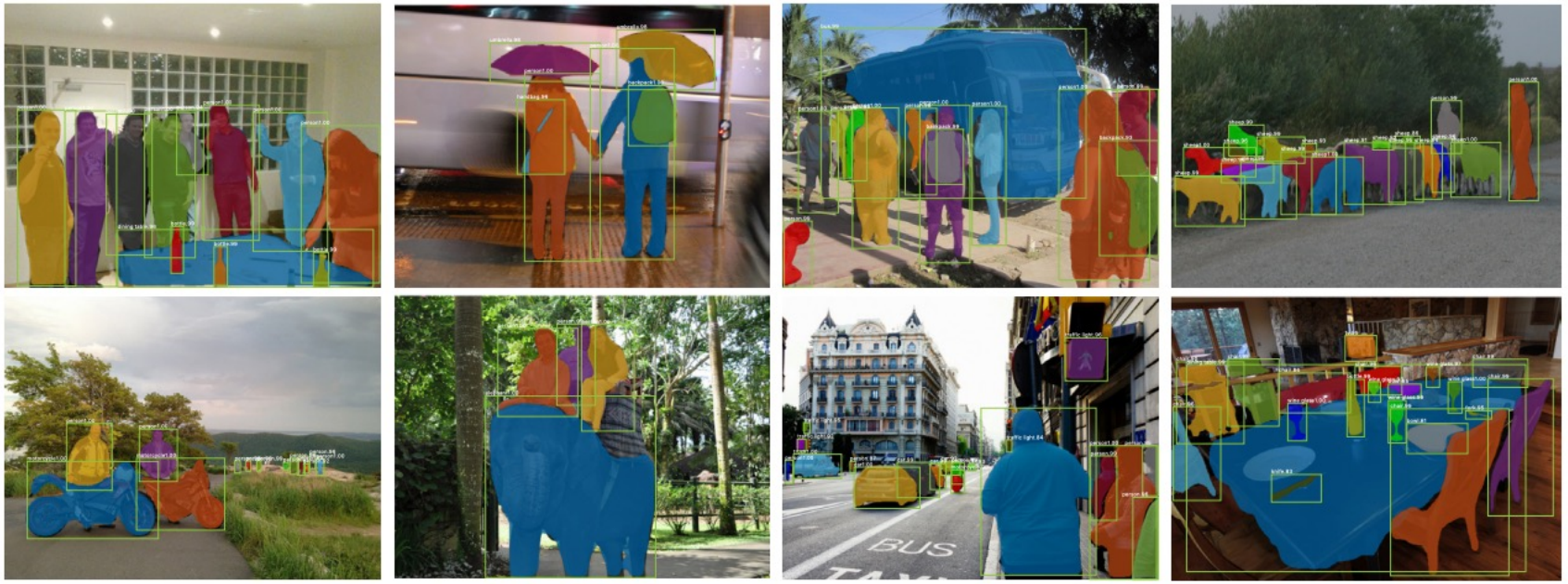




深層学習での画像解析

領域分割 (Instance Segmentation)

Mask R-CNN He et al (2018)
…個々のオブジェクトを個別のエンティティとして領域分割



深層学習での画像解析

画像生成

DALL･E 2 Ramesh et al (2022)

input: **“A photo of an astronaut riding a horse”**

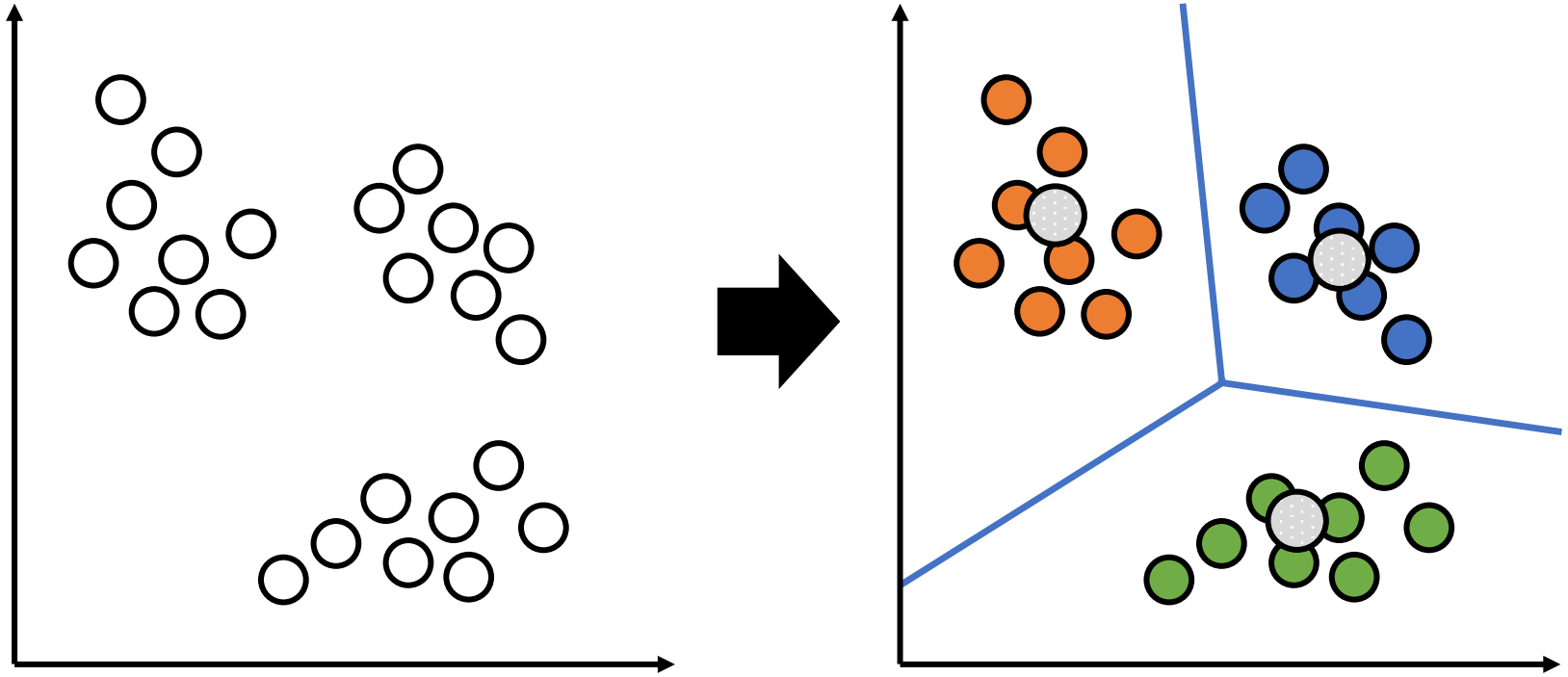
output:

https://twitter.com/OpenAI/status/1511714545529614338?s=20&t=wbHGVtVaHdWi1H0hFPkylw

教師なし学習：クラスタリング

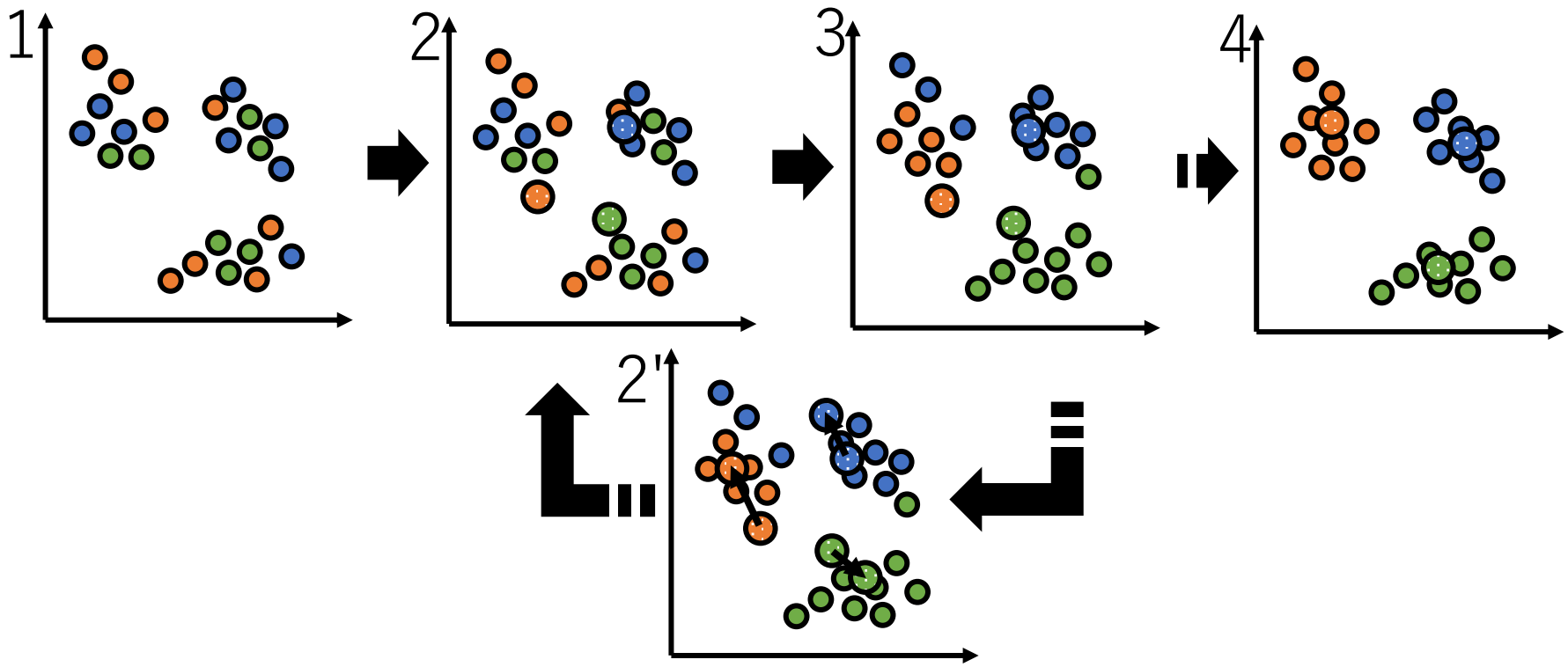
教師データなしで入力データ間の類似度をもとに
クラスタにグループ分けをする手法

N個の入力データをK個の部分集合に分割して各部分集合の重心を代表点とする
→ 以下のデータならN=23個をK=3の部分集合に分割し　　が代表点



クラスタリングの一例：k-means法

1. N個のデータをランダムにK個のクラスタのどれかに振り分ける
2. 各クラスタの代表点を求める
3. 各データを各クラスタとの距離を求めて最短のクラスタに割り当てる
4. 2-3を繰り返してクラスタの割り当てが変化しなかった場合に終了する



ペンギンのクラスタリング

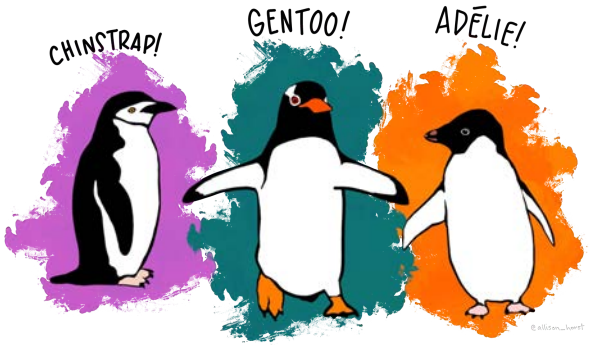
palmerpenguins[1]のデータセットを使う

```
# 01-01. パッケージのインストール
! pip install palmerpenguins

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from palmerpenguins import load_penguins
```

Colabにインストールされていないパッケージは
! pip install ${パッケージ名}
でインストールできる.




```
# 01-02. ペンギンデータセットの読み込み
penguins = load_penguins()
penguins.head()
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	year
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	male	2007
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	female	2007
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	female	2007
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN	2007
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	female	2007

[1] Horst, Allison. Hill, Alison. Gorman, Kristen. “palmerpenguins”. https://allisonhorst.github.io/palmerpenguins/

pandas：数表操作ライブラリ

数表操作に特化したライブラリのpandasを扱ってみる

```
# 01-03. データの行数・列数の出力
print(penguins.shape)
```

```
# 出力 (行数(index), 列数(column))
(344, 8)
```

```
# 01-04. データ構造の情報を出力
penguins.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   species            344 non-null    object
 1   island             344 non-null    object
 2   bill_length_mm     342 non-null    float64
 3   bill_depth_mm      342 non-null    float64
 4   flipper_length_mm  342 non-null    float64
 5   body_mass_g        342 non-null    float64
 6   sex                333 non-null    object
 7   year               344 non-null    int64
dtypes: float64(4), int64(1), object(3)
memory usage: 21.6+ KB
```

列名を出力

その列のデータの型を出力

その列の欠損していないデータ数を出力

pandas：数表操作ライブラリ

データを見る

```
# 01-05. 特定の列のデータをSeriesで出力
penguins['species']
```

```
# 01-06. 特定の列のデータをDataFrameで出力
penguins[['bill_length_mm', 'bill_depth_mm']]
```

	bill_length_mm	bill_depth_mm
0	39.1	18.7
1	39.5	17.4
2	40.3	18.0
3	NaN	NaN
4	36.7	19.3
...	...	...
339	55.8	19.8
340	43.5	18.1
341	49.6	18.2
342	50.8	19.0
343	50.2	18.7

344 rows × 2 columns

欠損値

データには**欠損値**(NaN)が含まれる場合があり
・その列の代表値や欠損値とわかる値で補間する
・そのデータを含む行(もしくは列)を削除する　などをする

今回は欠損値を含む行を削除する

```
# 01-07. 欠損データの削除
penguins = penguins.dropna(axis=0)
penguins[['bill_length_mm', 'bill_depth_mm']]
```

	bill_length_mm	bill_depth_mm
0	39.1	18.7
1	39.5	17.4
2	40.3	18.0
4	36.7	19.3
5	39.3	20.6
...	...	...
339	55.8	19.8
340	43.5	18.1
341	49.6	18.2
342	50.8	19.0
343	50.2	18.7

333 rows × 2 columns

```
# 01-08. index番号を振り直す
penguins = penguins.reset_index(drop=True)
```

pandas：数表操作ライブラリ

データを見る

```
# 01-09. ユニークな要素の出現回数
penguins['species'].value_counts()
```

```
Adelie     146
Gentoo     119
Chinstrap   68
Name: species, dtype: int64
```

```
# 01-10. 特定の要素を含むデータを抽出
penguins[penguins['species']=='Adelie']
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	year
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	male	2007
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	female	2007
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	female	2007
3	Adelie	Torgersen	36.7	19.3	193.0	3450.0	female	2007
4	Adelie	Torgersen	39.3	20.6	190.0	3650.0	male	2007
...	...	...	...	...	...	...	...	...
141	Adelie	Dream	36.6	18.4	184.0	3475.0	female	2009
142	Adelie	Dream	36.0	17.8	195.0	3450.0	female	2009
143	Adelie	Dream	37.8	18.1	193.0	3750.0	male	2009
144	Adelie	Dream	36.0	17.1	187.0	3700.0	female	2009
145	Adelie	Dream	41.5	18.5	201.0	4000.0	male	2009

146 rows × 8 columns

複数条件を指定するには，丸括弧で条件を指定する[1]

```
penguins[(penguins['species']=='Adelie') & (penguins['sex']=='male')]
```

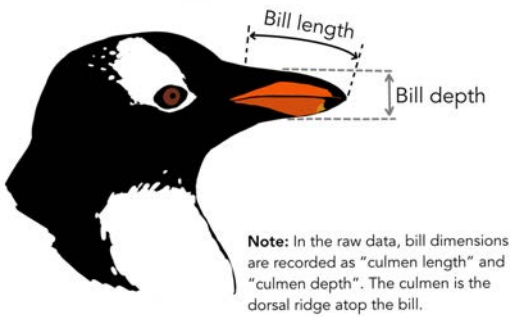
AND ： &
OR ： |
NOT ： ~

[1] pandasで複数条件のAND, OR, NOTから行を抽出（選択） https://note.nkmk.me/python-pandas-multiple-conditions/

pandas：数表操作ライブラリ

pandasでの四則演算や新しい列の追加

```
# 01-11. pandasでの四則演算
penguins['bill_length_mm'] + 1
penguins['bill_length_mm'] - 2
penguins['bill_length_mm'] * 3
penguins['bill_length_mm'] / 4
```




```
# 01-12. 他の列との四則演算
penguins['bill_length_per_body_mass_mm/g'] =¥
        penguins['bill_length_mm'] /
penguins['body_mass_g']
penguins.head()
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	year	bill_length_per_body_mass_mm/g
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	male	2007	0.010427
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	female	2007	0.010395
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	female	2007	0.012400
3	Adelie	Torgersen	36.7	19.3	193.0	3450.0	female	2007	0.010638
4	Adelie	Torgersen	39.3	20.6	190.0	3650.0	male	2007	0.010767

pandas：数表操作ライブラリ

matplotlibを用いた可視化

```
# 01-13. matplotlibを用いた散布図での可視化
plt.scatter(penguins['bill_length_mm'], penguins['bill_depth_mm'])
```

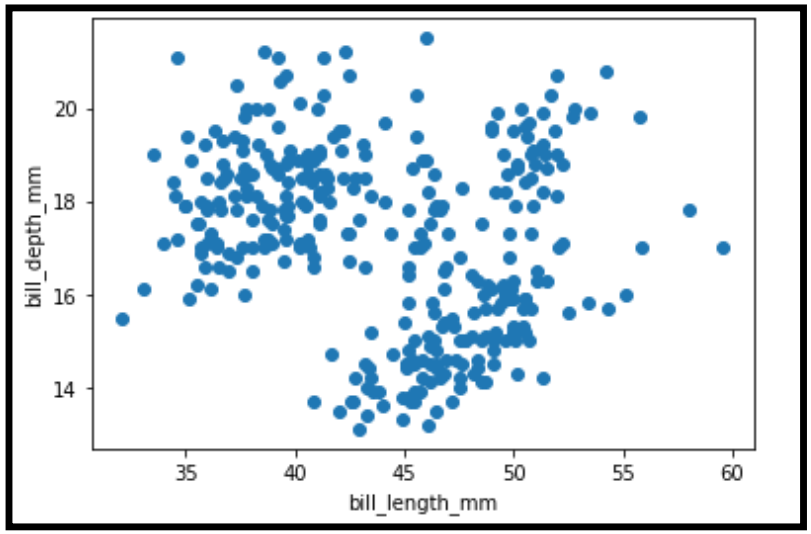
(x, y)

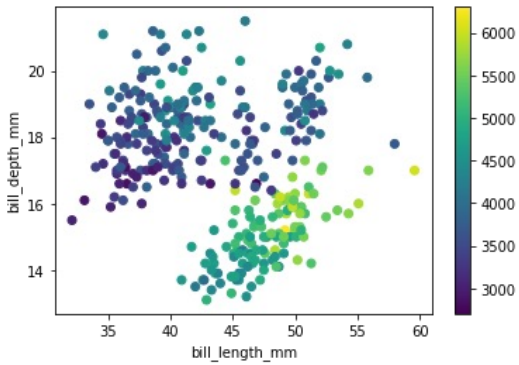
```
# 01-13-1. 散布図での可視化
x_col = 'bill_length_mm'
y_col = 'bill_depth_mm'
plt.scatter(penguins[x_col], penguins[y_col])
plt.xlabel(x_col)
plt.ylabel(y_col)
plt.show()
```

scatterのx・yの配列以外の引数

s　：点のサイズ(デフォルト値: 20)
marker：マーカーの形(例: “o”, “x”, “^”)
alpha ：透明度 0(透明)~1(不透明)
c　：色または連続した色の配列

```
plt.scatter(
    penguins[x_col],
    penguins[y_col],
    c=penguins['body_mass_g'],
)
plt.colorbar()
```

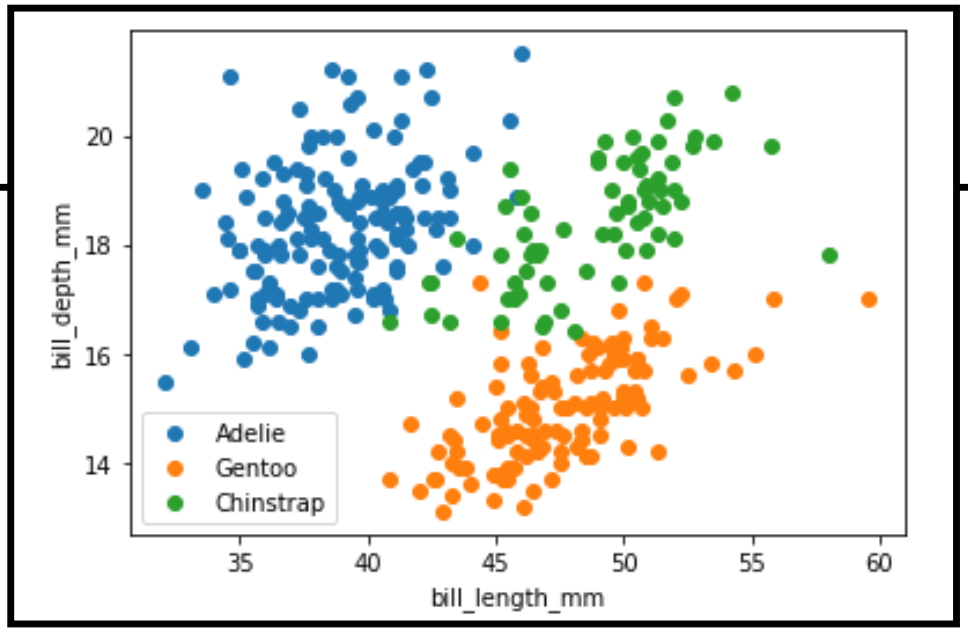




pandas：数表操作ライブラリ

matplotlibを用いた可視化

```
# 01-14. 種ごとの散布図
x_col = 'bill_length_mm'
y_col = 'bill_depth_mm'
adelie_df = penguins[penguins['species']=='Adelie']
plt.scatter(adelie_df[x_col], adelie_df[y_col], label='Adelie')
gentoo_df = penguins[penguins['species']=='Gentoo']
plt.scatter(gentoo_df[x_col], gentoo_df[y_col], label='Gentoo')
chinstrap_df = penguins[penguins['species']=='Chinstrap']
plt.scatter(chinstrap_df[x_col], chinstrap_df[y_col], label='Chinstrap')
plt.xlabel(x_col)
plt.ylabel(y_col)
plt.legend()
plt.show()
```



14:50から再開

pandas：数表操作ライブラリ

機械学習の前処理 –標準化–

機会学習モデルを構築する前に学習できる・しやすいように**前処理**を行う
今回は各列データの**平均値**(=0)と**分散**(=1)を揃える**標準化**を行う

→測定単位に影響がないを特徴ベクトルを構成することができる

$$Z = \frac{X - \mu}{\sigma}$$

μ: Xの平均値
σ: Xの標準偏差

```
# 01-15. 標準化
mean_bill_depth = penguins['bill_depth_mm'].mean() # 平均値
std_bill_depth = penguins['bill_depth_mm'].std() # 標準偏差
(penguins['bill_depth_mm'] - mean_bill_depth) / std_bill_depth
```

scikit-learn：機械学習ライブラリ

sklearn.preprocessing.StandardScalerを用いた標準化

scikit-learnは機械学習で用いるアルゴリズムやモデルが入ったライブラリ

```
# 01-16. scikit-learnでの標準化
from sklearn.preprocessing import StandardScaler

standard = StandardScaler()
standard.fit(penguins[['bill_depth_mm']])
standard.transform(penguins[['bill_depth_mm']])
```

二重括弧にしてDataFrame型で与える必要あり

```
# 01-17. 標準化したデータの追加
columns = [
    'bill_length_mm', 'bill_depth_mm', 'flipper_length_mm', 'body_mass_g'
]
for col in columns:
    new_col = 'standard_' + col
    standard = StandardScaler()
    standard.fit(penguins[[col]])
    penguins[[new_col]] = standard.transform(penguins[[col]])
```

scikit-learn：機械学習ライブラリ

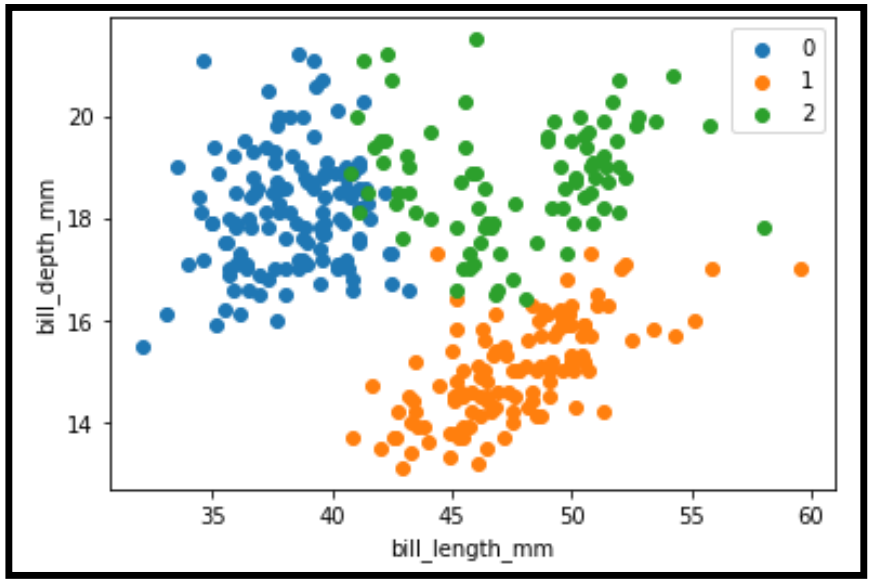
sklearn.cluster.KMeansを用いたクラスタリング

```
# 01-18. scikit-learnのKMeansでクラスタリング
from sklearn.cluster import KMeans

columns = [
'standard_bill_length_mm', 'standard_bill_depth_mm',
'standard_flipper_length_mm', 'standard_body_mass_g'
]
km = KMeans(n_clusters=3, random_state=42)
km.fit(penguins[columns])
print(km.labels_)
```

sklearn.cluster.KMeansの主要な引数

n_clusters　：クラスター数(重心数)
random_state：seed値
（決めないと毎回結果が変わる）



教師あり学習：クラス分類

ロジスティック回帰を用いたクラス分類

2クラスのロジスティック回帰

予測したい事象が起こる確率をp，その起こりやすさ(オッズ)を$\frac{p}{1-p}$とする．このオッズの対数を入力ベクトルx_iの線形和で表すと，

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n = \sum_{i=0}^{n} \beta_i x_i$$

β_i:係数, n:入力ベクトルの次元数

この確率pで整理すると，

$$p = \frac{1}{1 + \exp\left(-\sum_{i=0}^{n} \beta_i x_i\right)}$$

となり，この係数β_iを目的変数を用いて
最尤推定法によって解を求め確率値pを求める．

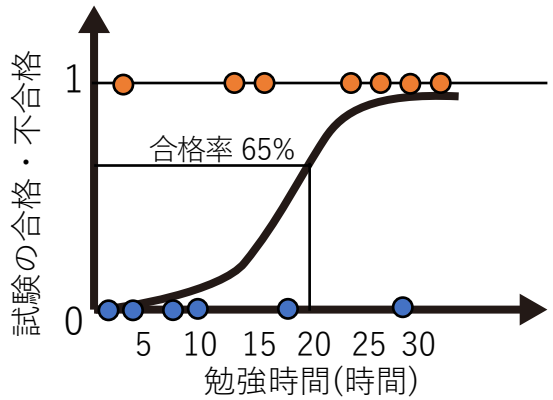
多クラスの場合はクラスkごとの線形和
$a_k = \sum_{i=0}^{n} \beta_{ki} x_i$とおくと，

$$p_k = \frac{\exp(a_k)}{\sum_{j}^{K} \exp(a_j)}$$

とおける(softmax)．
これを最尤推定法によって係数βを求める．

ロジスティック回帰の例

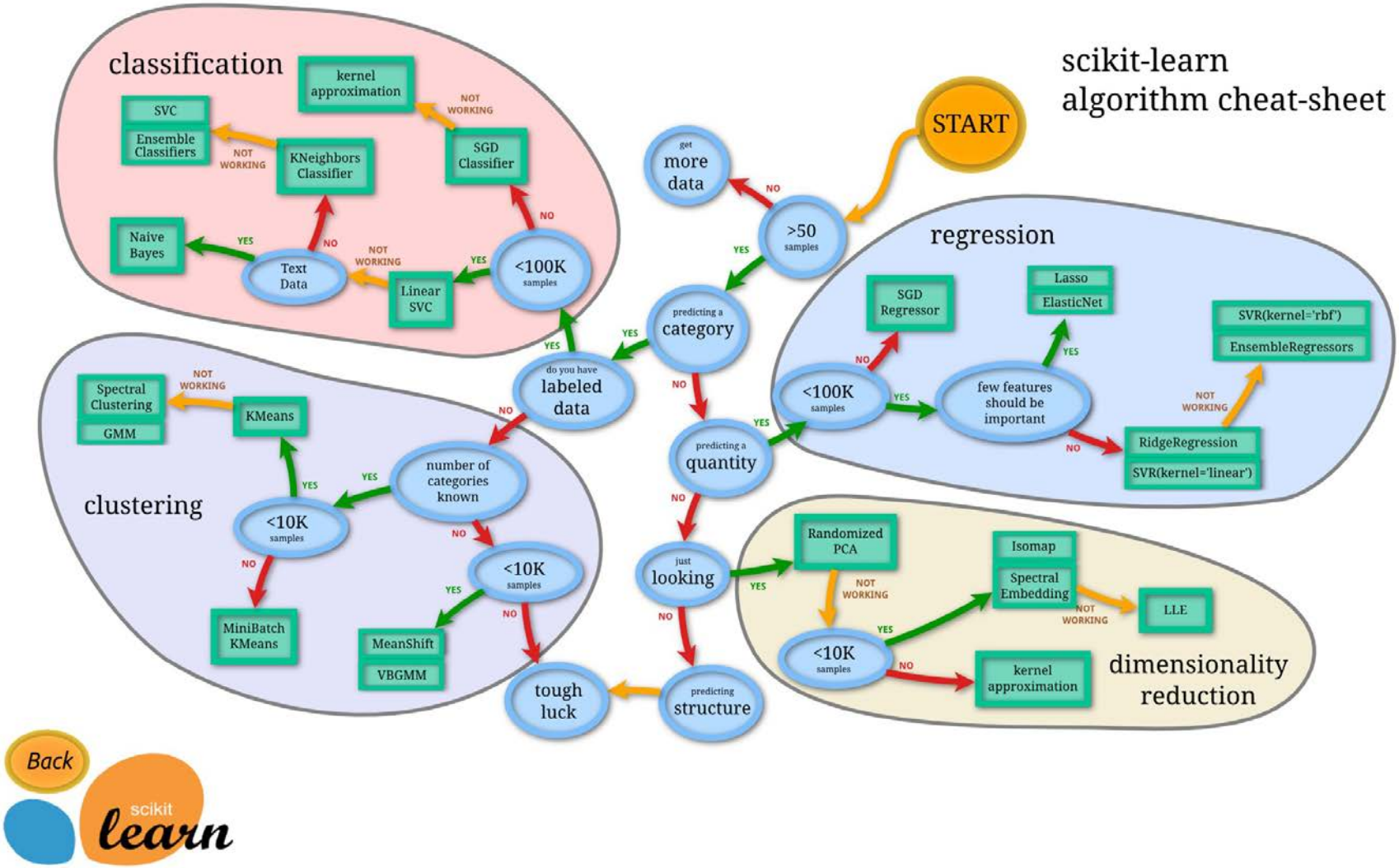
ある試験の勉強時間に対する合格・不合格の事象の起こる確率を求めたい．
このときにロジスティック回帰を用いてこれまでの勉強時間と合格・不合格のデータを用いてパラメータ推定して可視化した図が以下である．



=> **<u>20時間勉強すれば65%の確率で合格</u>**

scikit-learn：機械学習ライブラリ

機械学習アルゴリズムのフローチャート[1]



[1] Pedregosa, Fabian et al. 2011. “Scikit-Learn: Machine Learning in Python.” *Journal of Machine Learning Research*

教師あり学習：クラス分類

カテゴリ変数を整数値に変換

```
# 02-01. speciesを整数値に変換
from sklearn.preprocessing import LabelEncoder

labelencoder = LabelEncoder()
labelencoder.fit(penguins['species'])
penguins['label_species'] =¥
    labelencoder.transform(penguins['species'])
```

labelencoder.classes_
で整数値の順番に元の変数が入ったlistを取得

```
# 02-02. 整数値からspeciesに逆変換
labelencoder.inverse_transform(penguins['label_species'])
```

教師あり学習：クラス分類

Hold-Out検証

機械学習モデルの汎化性能を検証するためにパラメータ推定を行うデータ(train),
パラメータ推定を行わないデータ(test)として，パラメータを推定したモデルの
性能評価をtestデータで行う検証方法.

```
# 02-03 学習用データと評価用データに分ける
from sklearn.model_selection import train_test_split

columns = [
    'standard_bill_length_mm', 'standard_bill_depth_mm',
    'standard_flipper_length_mm', 'standard_body_mass_g'
]
penguins_X = penguins[columns]
penguins_y = penguins['label_species']

train_X, test_X, train_y, test_y = train_test_split(
    penguins_X, penguins_y, test_size=0.2, random_state=42
)
```

20%のデータを
testデータとする

教師あり学習：クラス分類

ロジスティック回帰を用いたクラス分類と精度評価

```
# 02-04 ロジスティック回帰での多クラス分類
from sklearn.linear_model import LogisticRegression

model = LogisticRegression(random_state=42)
model.fit(train_X, train_y)
test_predictions = model.predict(test_X)
```

model.predict_proba
で予測確率値を取得

評価指標として今回は元のクラスと予測クラスが
何割正解しているかのaccuracy(正解率)で確認

```
# 02-05 精度評価(正解率)
from sklearn.metrics import accuracy_score

print(accuracy_score(test_y, test_predictions))
```

常に”正解率”で精度評価すればいいとは限らない.
例えば，1%がクラス1，99%がクラス2の場合は，全てのデータでクラス2と予測すれば99%の精度が出るが，クラス1の感度が0になる問題がある.
そのため正解率以外に，クラス1と予測したもののうち実際にクラス1の割合を適合率(precision)，全体のクラス1の中でクラス1と予測できた割合を再現率(recall)とする指標がある.

本日の課題　ノーマル

1. palmerpenguinsのデータを用いて任意のクラスタ数でクラスタリングを行い，その結果を2つの列データを用いて散布図で出力せよ．また，その過程をコードに記載すること．

2. 機械学習や深層学習を用いて解きたい課題を見つけて，その課題とどう解決できるかを簡潔に記せ．

3. 質問，意見，要望等どうぞ．

課題をノートブック（.ipynbファイル）にまとめて，Moodleにて提出すること

ファイル名は[回数，01~15]_[難易度，ノーマル nかハード h].ipynb．例．12_n.ipynb

本日の課題　ハード

1. ある調査隊が新しく10体のペンギンを調査した.
   このペンギンはAdelie，Gentoo，Chinstrapのどれかだが,
   見分けることができず計測データのみ持って帰ってきた.
   この10体のペンギンがどれか，そう考えた手法と共に説明せよ.
   (データはmoodleにアップロードしています)

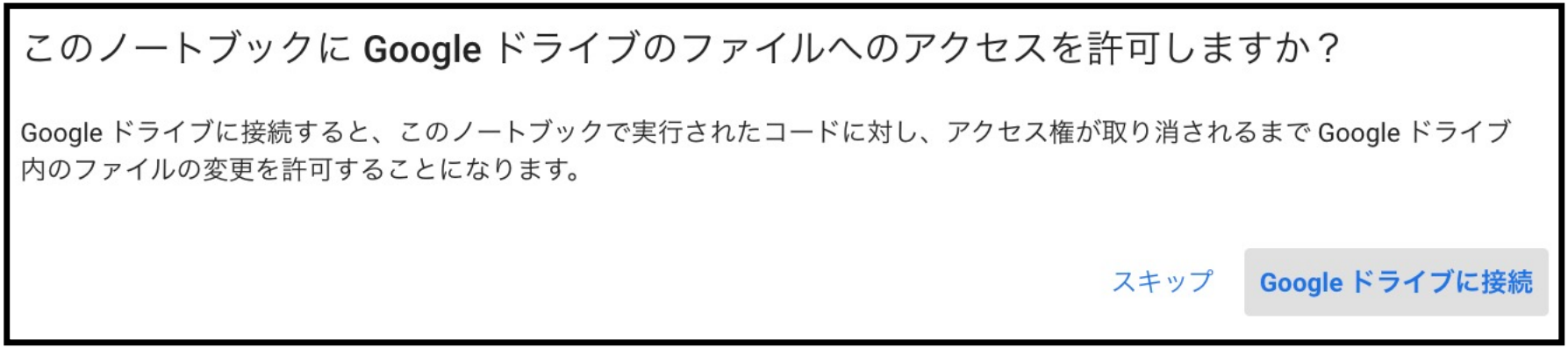
課題をノートブック（.ipynbファイル）にまとめて，Moodleにて提出すること

ファイル名は[回数，01~15]_[難易度，ノーマル nかハード h].ipynb．例．12_h.ipynb

補足 colabでドライブをマウント

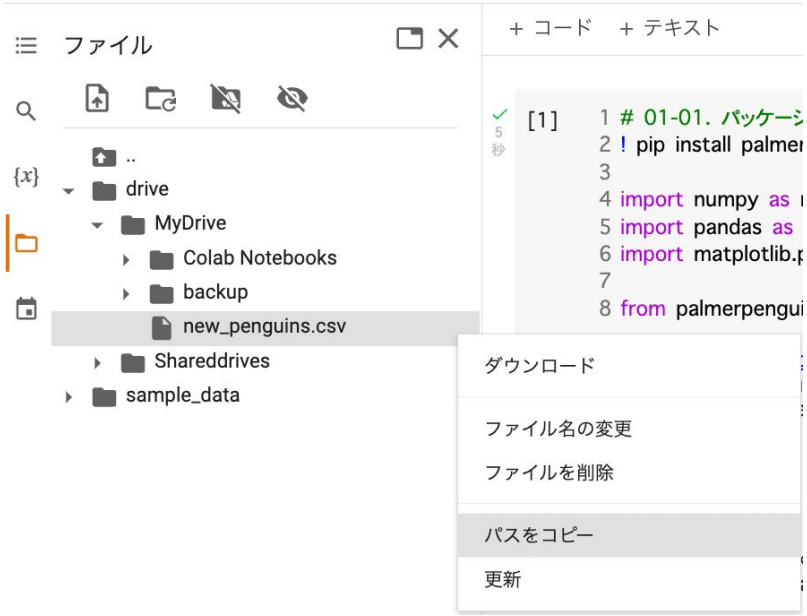
1.
colabの画面左側にフォルダアイコンを
クリックして，左から3番目の
”ドライブをマウント”をクリック

2.
確認画面が出るので”Google ドライブに接続”をクリック



補足 colabでドライブをマウント

3.
マウントすると”drive”フォルダが
追加されてデータを保存したところまで
開いて，”パスをコピー”を行う



4.
pandas.read_csvを使ってデータを読み込み

```
new_penguins = pd.read_csv('/content/drive/MyDrive/new_penguins.csv')
```