

第2回

線形回帰～ロジスティック回帰

本教材を使用した際にはお手数ですが、
下記アンケートフォームにご協力下さい。

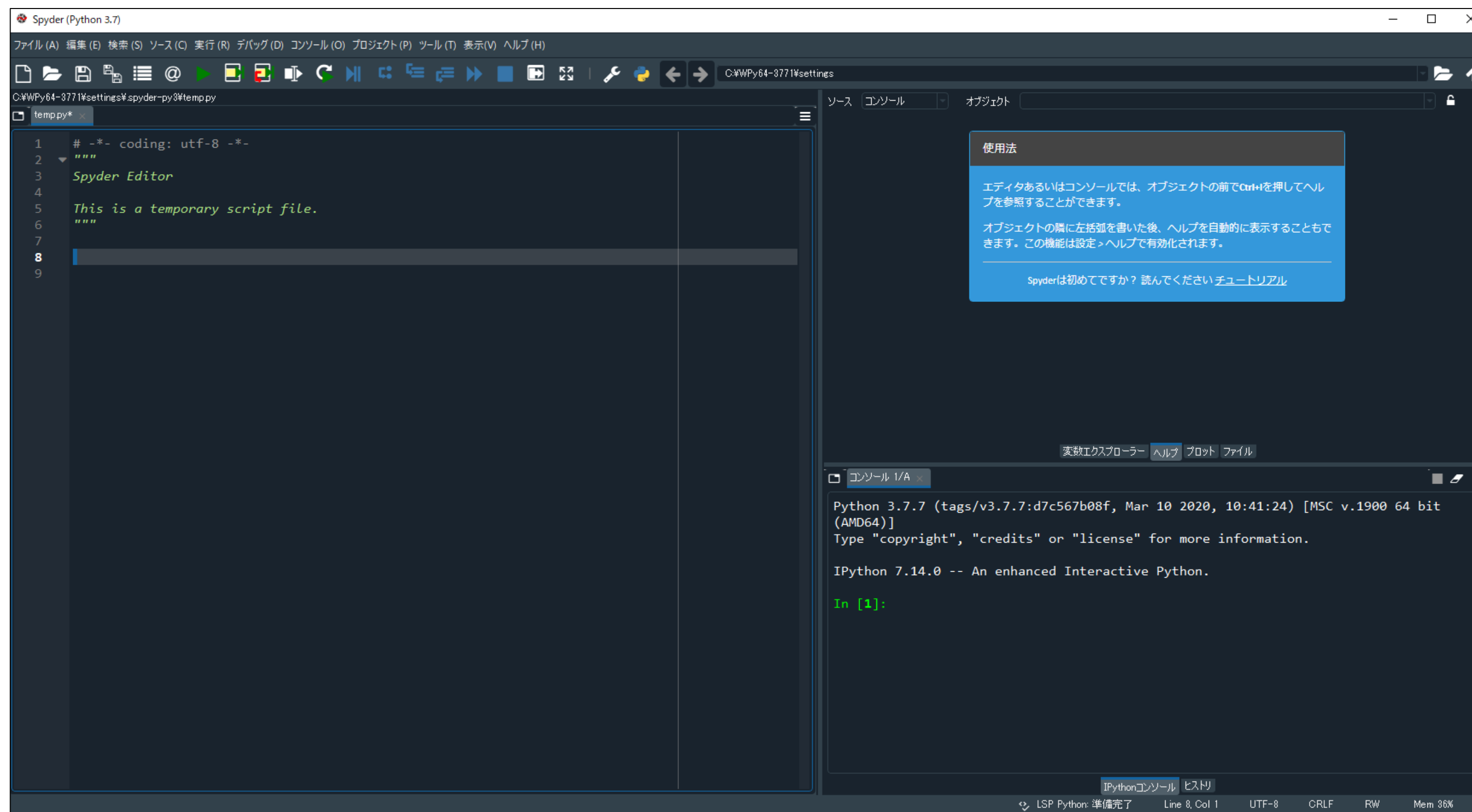
<https://forms.gle/cgej2DL5PvneRhCp8>

統合教育機構
須藤毅顕

Spyderの準備

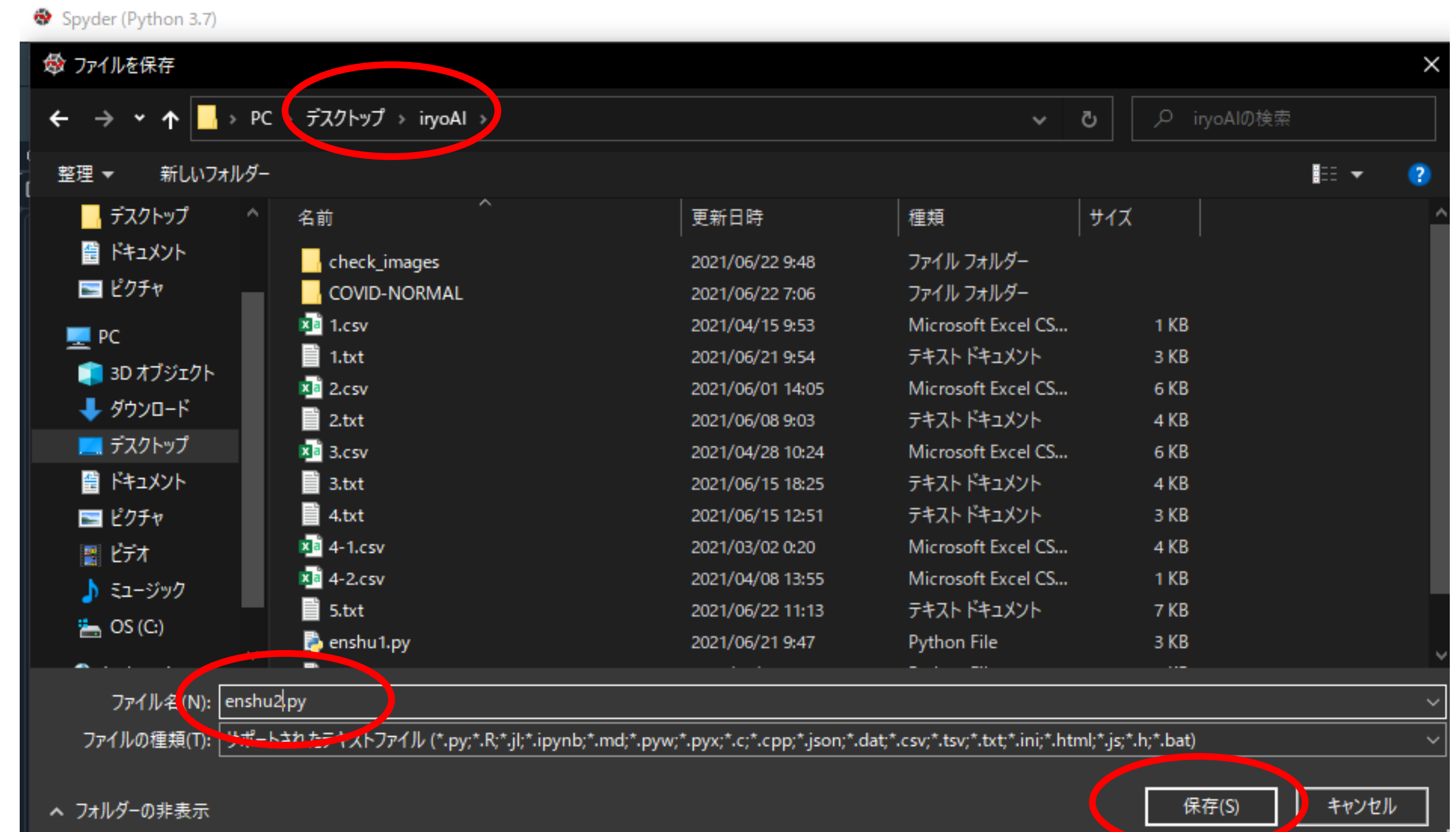
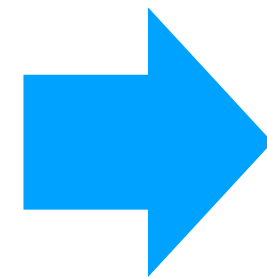
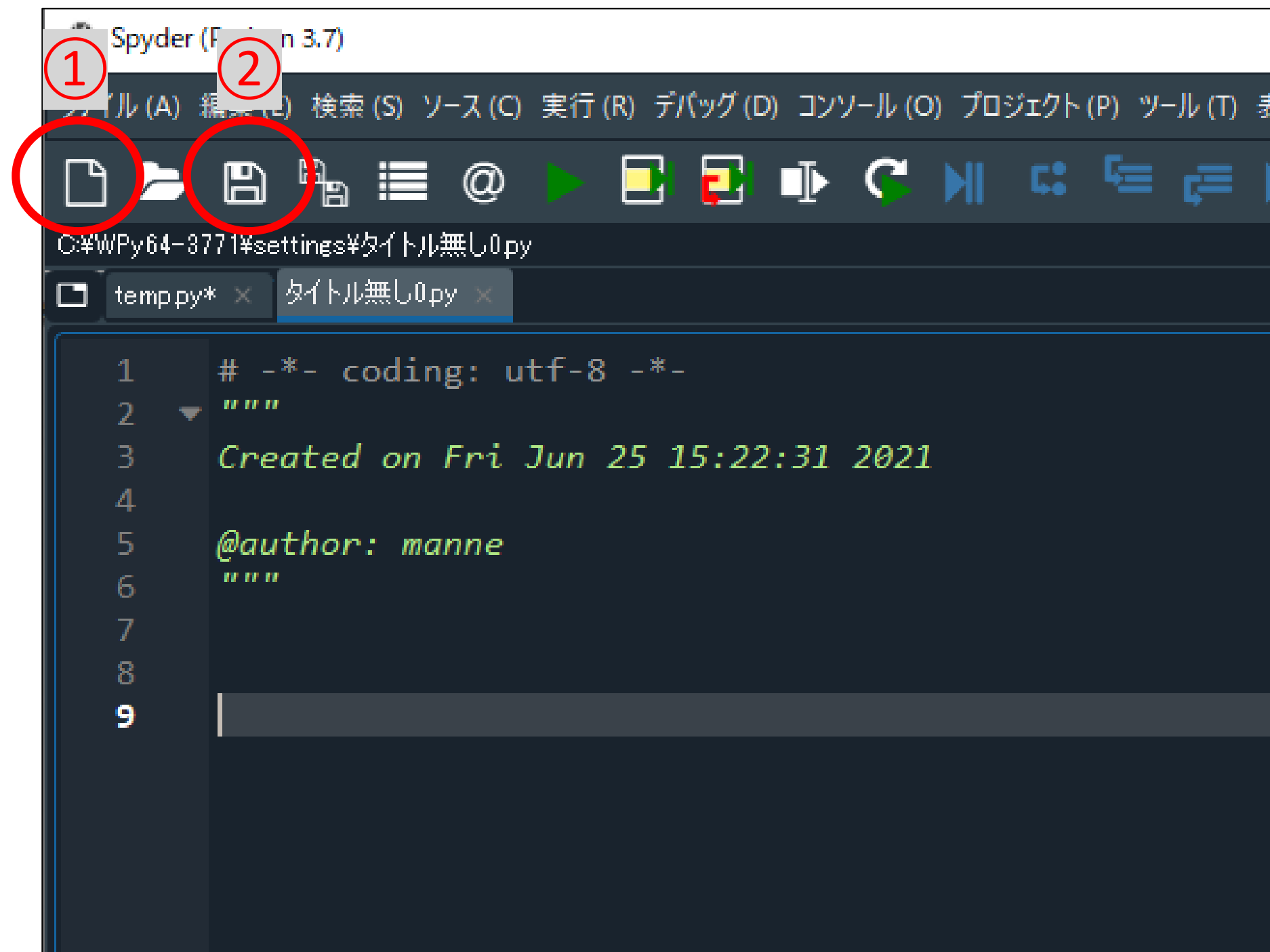
Spyderを開きましょう(anacondaならiryoyAIの仮想環境)
Webclassの2.txtを開きましょう

Webclassの2.csvがiryoyAIのフォルダに保存されていることを確認しましょう



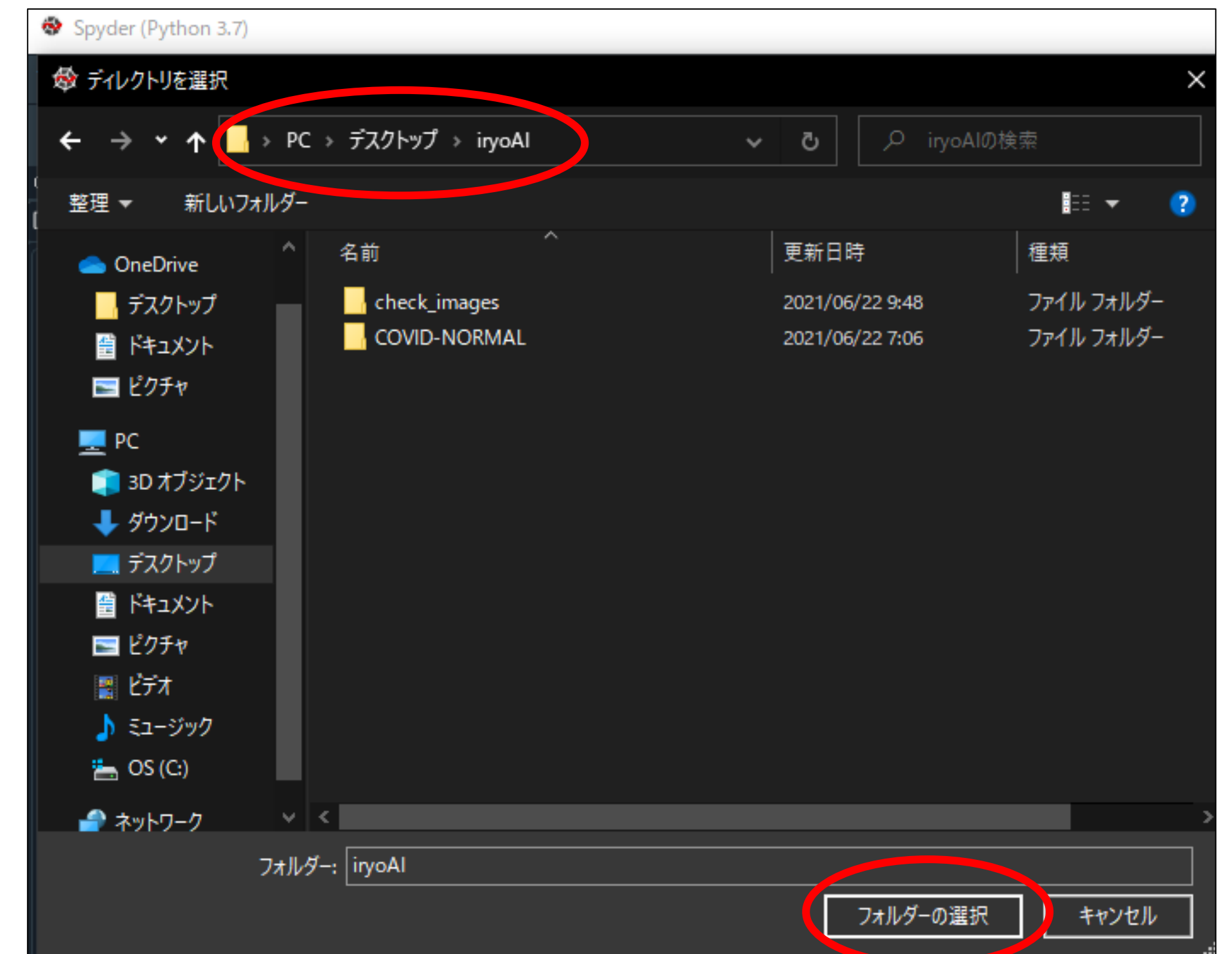
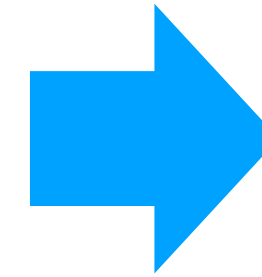
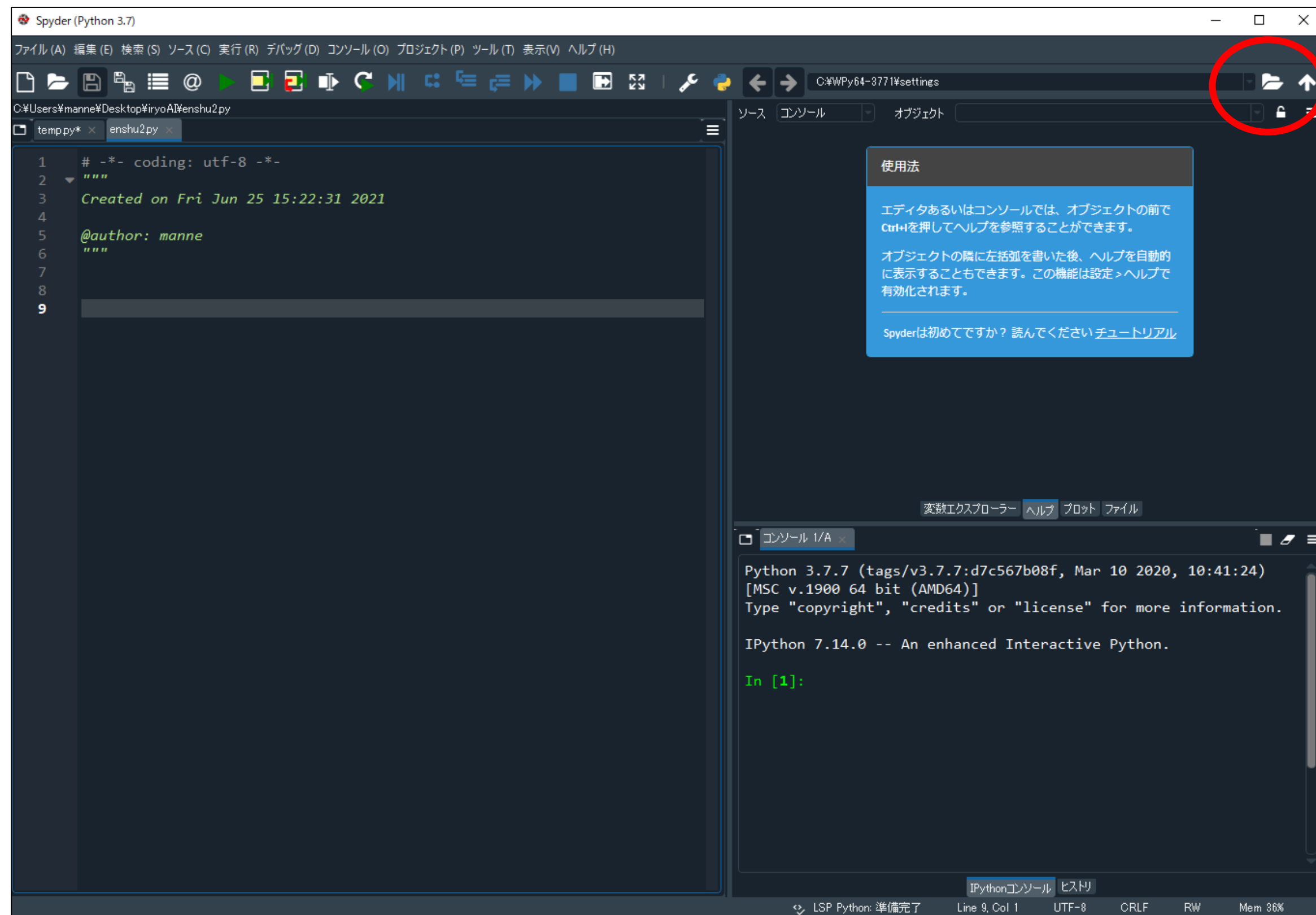
Spyderの準備

新規ファイルを作成して、enshu2.pyとしてiryoAIに保存しましょう



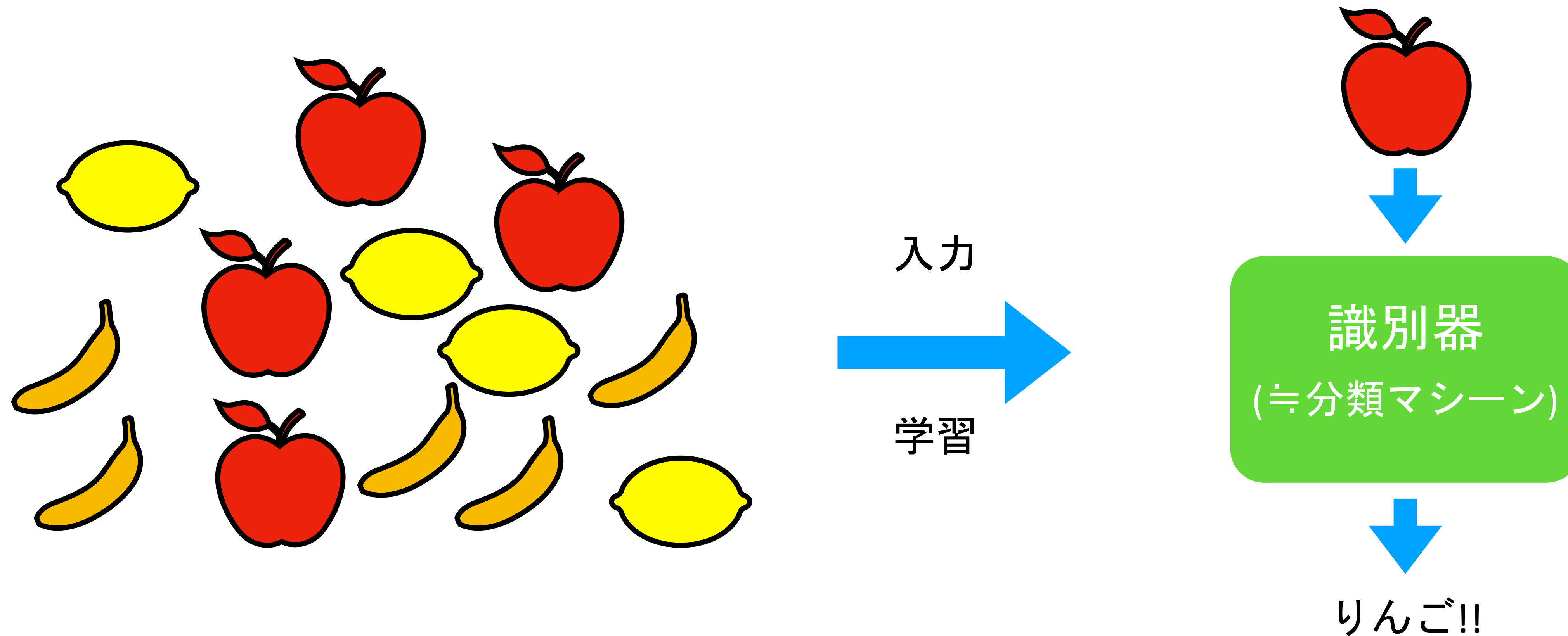
Spyderの準備

Spyderの右上の  をクリックして、作業場所をiryoAIに設定しましょう



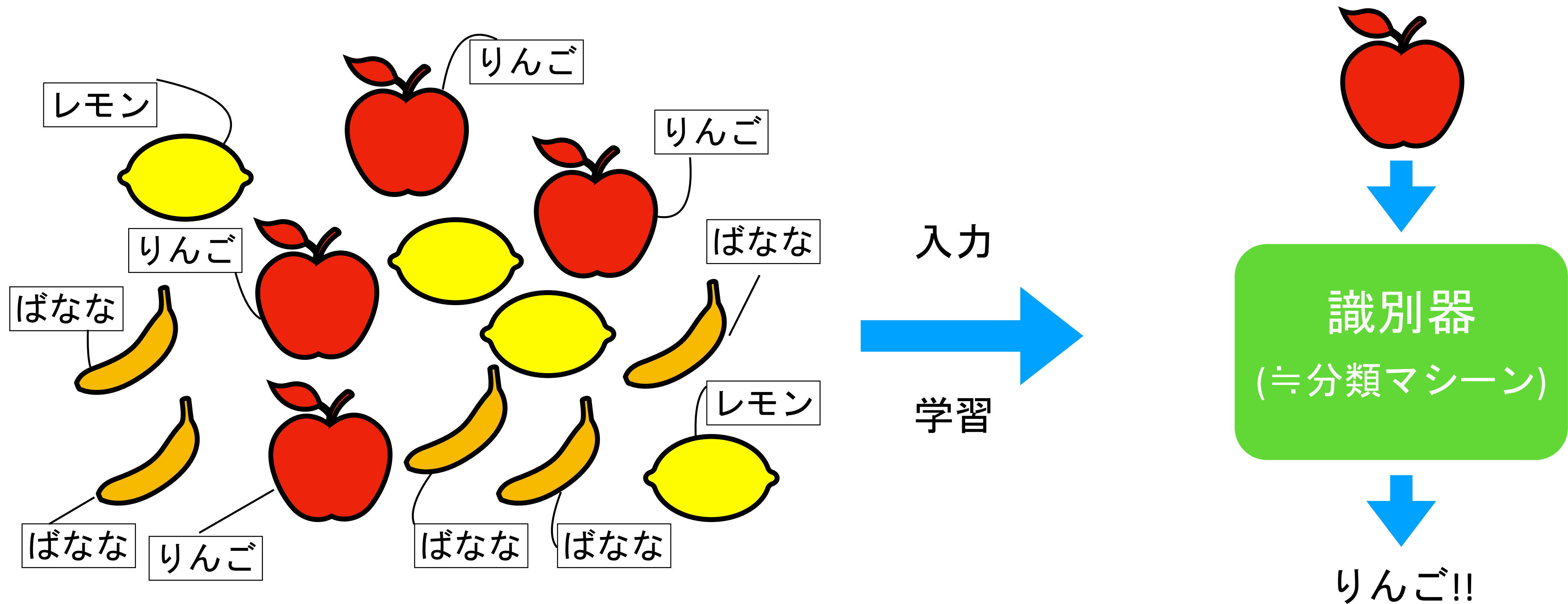
機械学習を実践してみよう！！

- 機械学習はコンピュータにデータを学習させて分類、予測などを行う手法

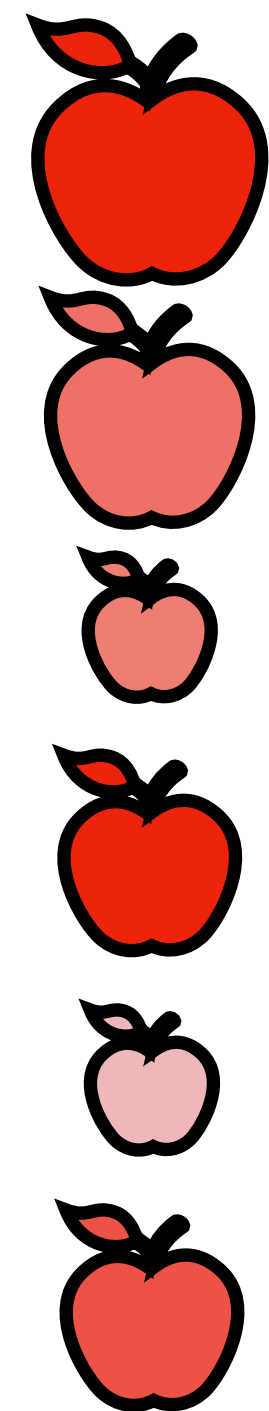


機械学習を実践してみよう！！

教師あり機械学習は正解をセットで学習させて識別器を作る

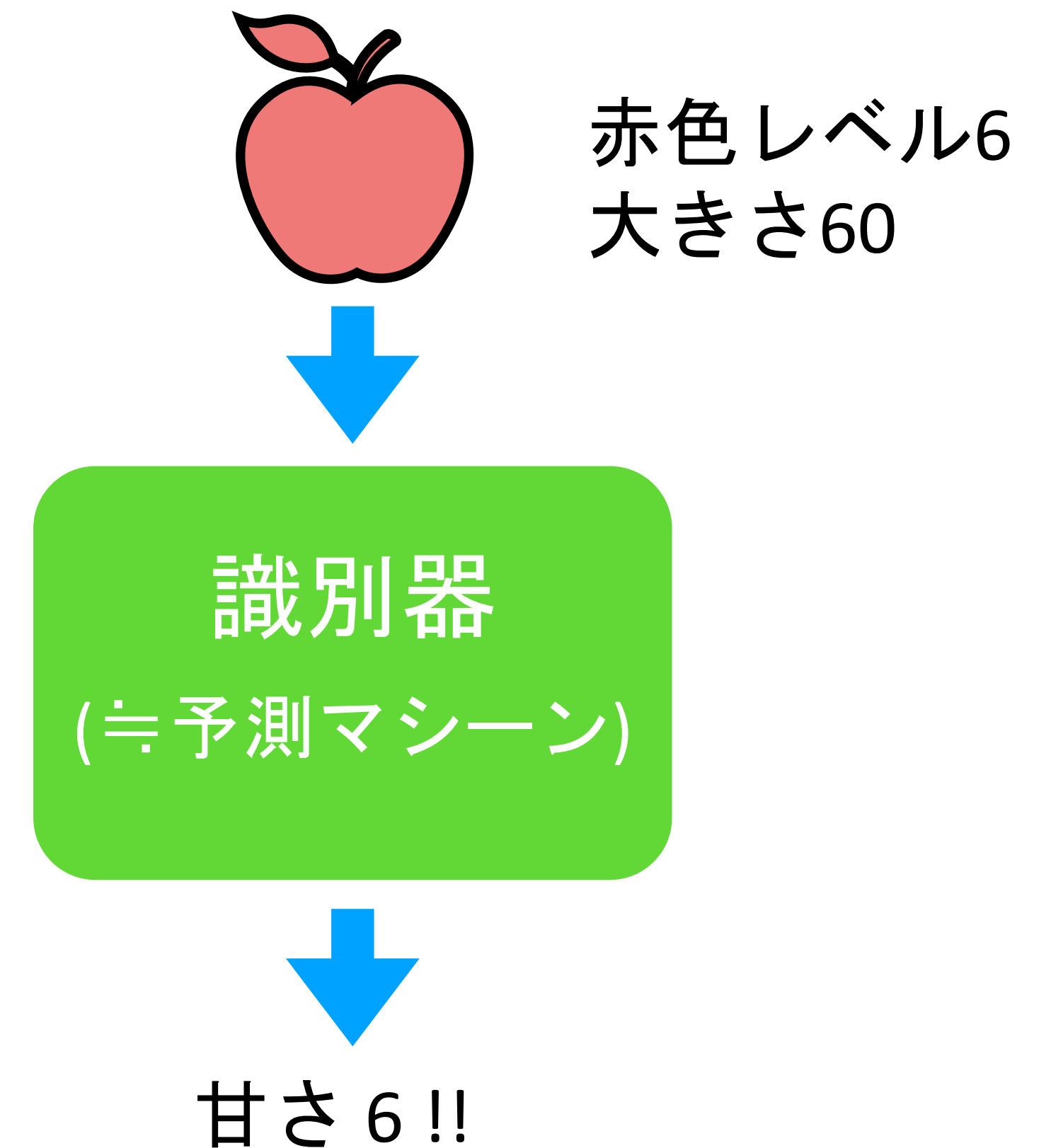


教師あり機械学習の中の「回帰」を実践してみよう！！



	特徴1	特徴2	正解
	赤色レベル	大きさ	甘さレベル
りんご1	10	100	5
りんご2	8	88	8
りんご3	7	44	9
りんご4	10	73	7
りんご5	5	50	4
りんご6	8	98	10

入力
→
学習



まずは教師あり学習の1つである「回帰」で予測をしてみよう

回帰分析

回帰分析は与えられたデータが当てはまるような関数を考える

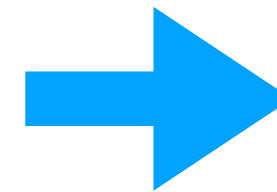
単回帰分析は直線で表すことが出来る(線形単回帰分析) → 回帰直線

特徴 1 つ

特徴＝説明変数

正解＝目的変数

赤色レベル



甘さレベル

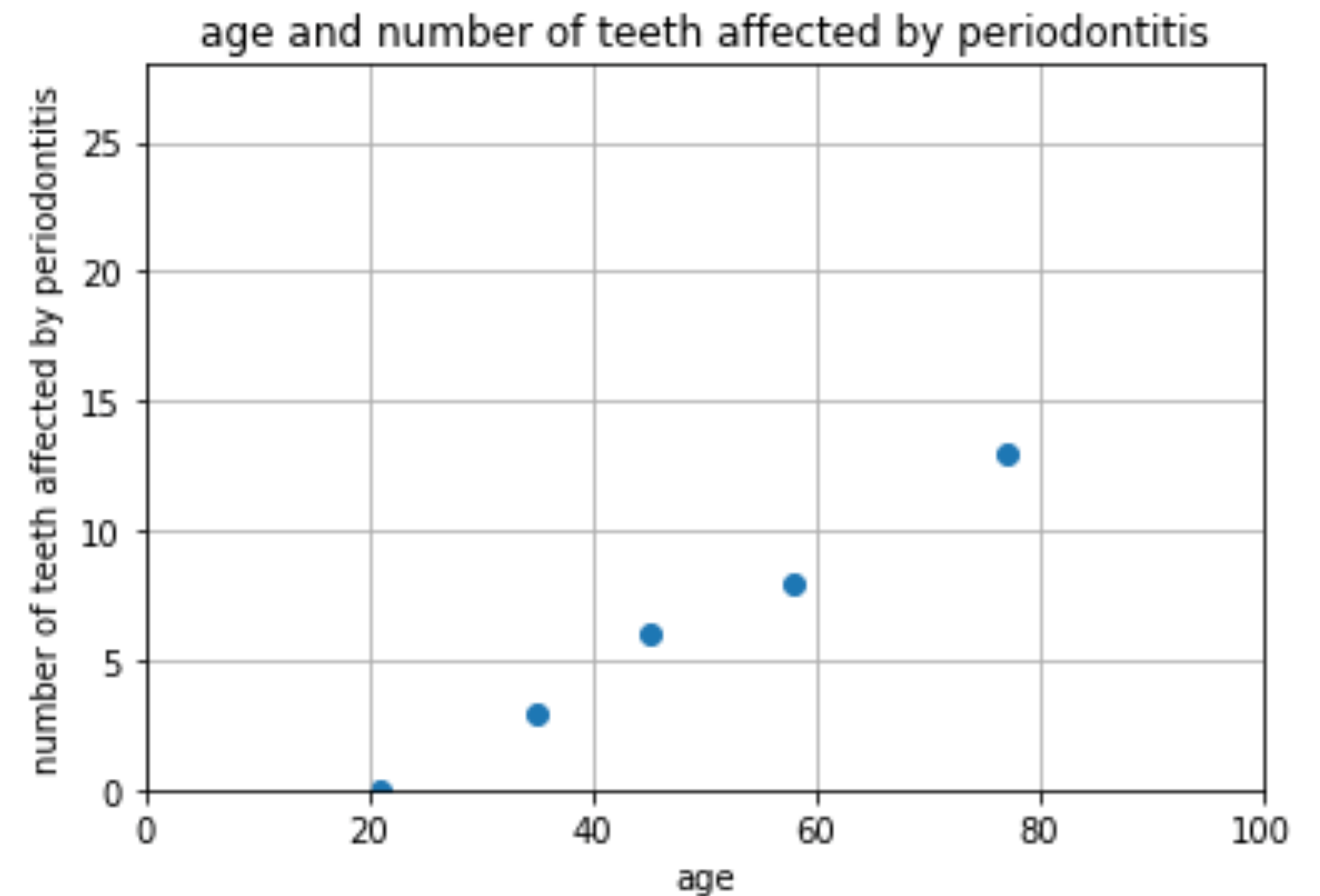
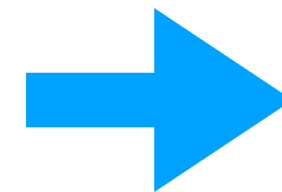
回帰式

$$y = b_0 + b_1x \quad (y : \text{目的変数、} x : \text{説明変数、} b_0 : \text{切片、} b_1 : \text{傾き})$$

前回のデータ

5人の年齢と歯周病の歯の本数を作図するところまで行いました

被験者	年齢	歯周病の歯の本数
1	35	3
2	21	0
3	45	6
4	58	8
5	77	13

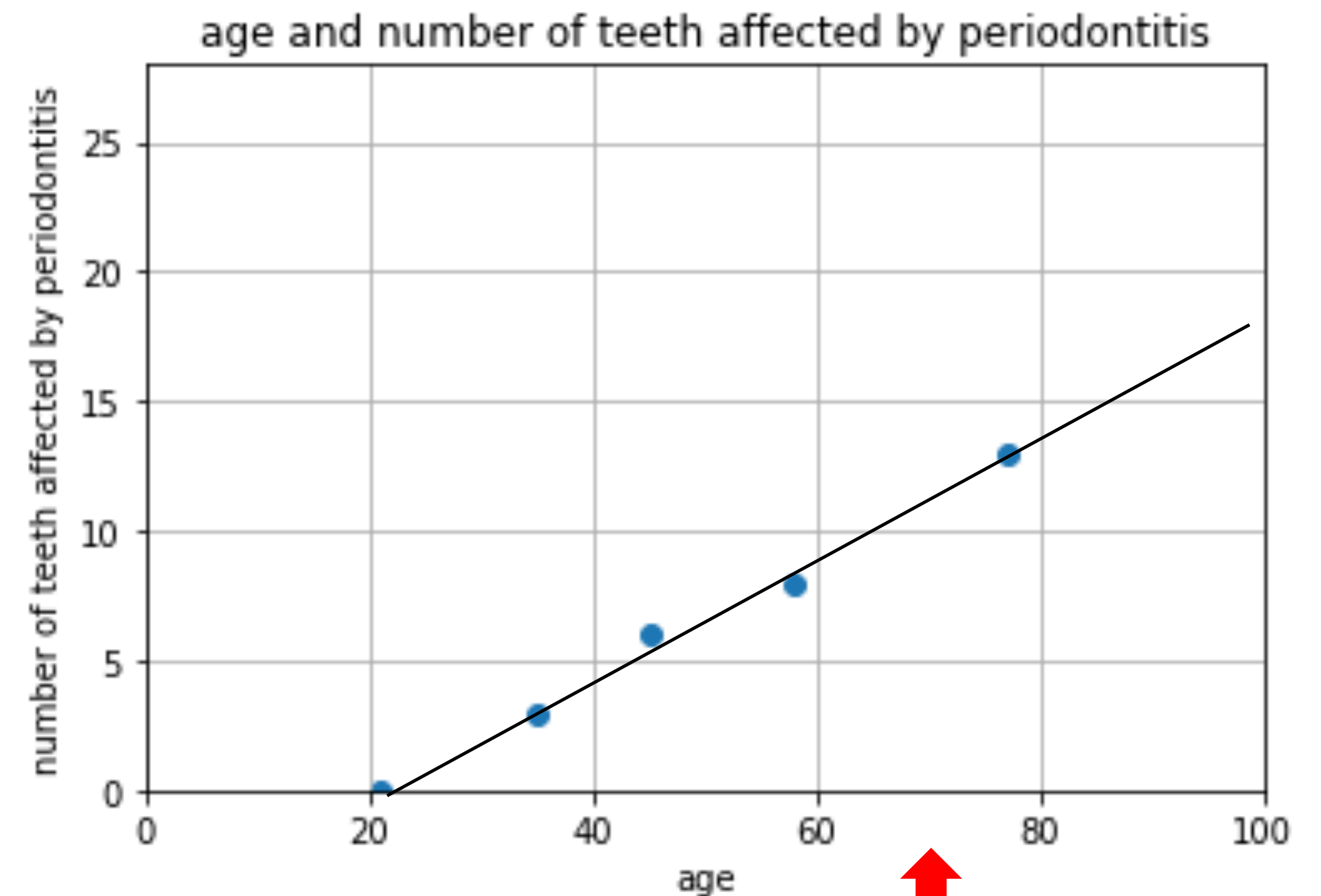
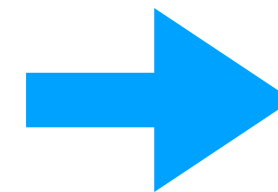


前回のデータ

5人の年齢と歯周病の歯の本数を作図するところまで行いました

このデータは直線に近似出来そう！？

被験者	年齢	歯周病の歯の本数
1	35	3
2	21	0
3	45	6
4	58	8
5	77	13



線形回帰で70歳の歯周病の歯の本数を予測する

線形回帰で88歳の歯周病の歯の本数を予測する

scikit-learnを用いた機械学習の書き方

- ① 学習モデルの選択
- ② データを入れて学習させる
- ③ 傾き(偏回帰係数)と切片(定数項)を求める
- ④ 予測を行う

線形回帰で88歳の歯周病の歯の本数を予測する

scikit-learnを用いた機械学習の書き方

- ① 学習モデルの選択(今回は線形回帰)
(モデル名) = `LinearRegression()`
- ② データを入れて学習させる
(モデル名).`fit`(説明変数, 目的変数)
- ③ 傾き(偏回帰係数)と切片(定数項)を求める
(モデル名).`coef_` #傾き
(モデル名).`intercept_` #切片
- ④ 予測を行う
(モデル名).`predict`(新たな説明変数)

1) 線形回帰分析をコピーしよう

```
from sklearn.linear_model import LinearRegression
```

```
x = [[35],[21],[45],[58],[77]]  
y = [3,0,6,8,13]
```

```
model = LinearRegression()
```

```
model.fit(x,y)
```

```
print(model.coef_)  
print(model.intercept_)
```

```
test = [[70]]  
num_teeth = model.predict(test)  
print("70歳の時の本数は",num_teeth,"本")
```

① 学習モデルの選択(今回は線形回帰)
(モデル名) = `LinearRegression()`

② データを入れて学習させる
(モデル名).`fit`(説明変数,目的変数)

③ 傾き(偏回帰係数)と切片(定数項)を求める
(モデル名).`coef_` #傾き
(モデル名).`intercept_` #切片

④ 予測を行う
(モデル名).`predict`(新たな説明変数)

1) 線形回帰分析をコピーしよう

```
from sklearn.linear_model import LinearRegression
```

```
x = [[35],[21],[45],[58],[77]]  
y = [3,0,6,8,13]
```

```
model = LinearRegression()
```

```
model.fit(x,y)
```

```
print(model.coef_)  
print(model.intercept_)
```

```
test = [[70]]  
num_teeth = model.predict(test)  
print("70歳の時の本数は",num_teeth,"本")
```

① 学習モデルの選択(今回は線形回帰)
(モデル名) = `LinearRegression()`

② データを入れて学習させる
(モデル名).`fit`(説明変数,目的変数)

③ 傾き(偏回帰係数)と切片(定数項)を求める
(モデル名).`coef_` #傾き
(モデル名).`intercept_` #切片

④ 予測を行う
(モデル名).`predict`(新たな説明変数)

x = [[35],[21],[45],[58],[77]], y = [3,0,6,8,13]として、
説明変数をx (年齢)、目的変数をy(歯の本数) に代入

(Scikit-learnを使うときは説明変数のデータを2次元配列にする)

線形回帰で88歳の歯周病の歯の本数を予測する

scikit-learnを用いた機械学習の書き方

①学習モデルの選択(今回は線形回帰)
(モデル名) = `LinearRegression()`

`LinearRegression()`をモデル名(変数)に代入することで
scikit-learnの`LinearRegression()`という
機能を使うことが出来る

モデル名は何でも良い

```
model = LinearRegression()
```


線形回帰で88歳の歯周病の歯の本数を予測する

scikit-learnを用いた機械学習の書き方

② データを入れて学習させる
(モデル名).fit(説明変数, 目的変数)

今回は、モデル名をmodel、説明変数をx(年齢)、
目的変数をy(歯周病の歯の本数)としたい

```
model.fit(x,y)
```

modelは線形回帰を選んでいるので、
これでxとyを用いて線形回帰による学習を行う

線形回帰で88歳の歯周病の歯の本数を予測する

scikit-learnを用いた機械学習の書き方

③傾き(偏回帰係数)と切片(定数項)を求める
(モデル名).coef_ #傾き
(モデル名).intercept_ #切片

線形回帰での傾きと切片を求める。
中身を出力したいので、print()を用いる

```
print(model.coef_)
```

```
print(model.intercept_)
```

線形回帰で88歳の歯周病の歯の本数を予測する

scikit-learnを用いた機械学習の書き方

④ 予測を行う
(モデル名).**predict**(新たな説明変数)

70才の時の歯の本数を知りたいので、

```
test = [[70]]  
num_teeth = model.predict(test)  
print("70才の時の本数は", num_teeth, "本")
```

線形回帰で88歳の歯周病の歯の本数を予測する

scikit-learnを用いた機械学習の書き方

④ 予測を行う
(モデル名).**predict**(新たな説明変数)

70才の時の歯の本数を知りたいので、

```
test = [[70]]  
num_teeth = model.predict([[70]])  
print("70才の時の本数は", model.predict([[70]]), "本")
```

線形回帰で88歳の歯周病の歯の本数を予測する

```
from sklearn.linear_model import LinearRegression
```

```
x = [[35],[21],[45],[58],[77]]
```

```
y = [3,0,6,8,13]
```

```
model = LinearRegression()
```

```
model.fit(x,y)
```

```
print(model.coef_)
```

```
print(model.intercept_)
```

```
test = [[70]]
```

```
num_teeth = model.predict(test)
```

```
print("70歳の時の本数は",num_teeth,"本")
```

① 学習モデルの選択(今回は線形回帰)
(モデル名) = `LinearRegression()`

② データを入れて学習させる
(モデル名).`fit`(説明変数,目的変数)

③ 傾き(偏回帰係数)と切片(定数項)を求める
(モデル名).`coef_` #傾き
(モデル名).`intercept_` #切片

④ 予測を行う
(モデル名).`predict`(新たな説明変数)

線形回帰で88歳の歯周病の歯の本数を予測する

```
from sklearn.linear_model import LinearRegression
```

```
x = [[35],[21],[45],[58],[77]]
```

```
y = [3,0,6,8,13]
```

```
model = LinearRegression()
```

```
model.fit(x,y)
```

```
print(model.coef_)
```

```
print(model.intercept_)
```

```
test = [[70]]
```

```
num_teeth = model.predict(test)
```

```
print("70歳の時の本数は",num_teeth,"本")
```

```
[[0.22983521]]
```

```
[-4.84822203]
```

```
70歳の時の本数は [[11.24024284]] 本
```

①学習モデルの選択(今回は線形回帰)
(モデル名) = LinearRegression()

②データを入れて学習させる
(モデル名).fit(説明変数,目的変数)

③傾き(偏回帰係数)と切片(定数項)を求める
(モデル名).coef_ #傾き
(モデル名).intercept_ #切片

④予測を行う
(モデル名).predict(新たな説明変数)

補足 Python3.7以降での一般的な書き方
print(f"70歳の時の本数は{num_teeth}本")
(フォーマット済み文字リテラルと言います。)

線形回帰で88歳の歯周病の歯の本数を予測する

```
[[0.22983521]]  
[-4.84822203]  
88歳の時の本数は [[15.37727667]] 本
```

$$y = b_0 + b_1x \quad (y : \text{目的変数、} x : \text{説明変数、} b_0 : \text{切片、} b_1 : \text{傾き})$$

$$y = (-4.84822203) + (0.22983521) x$$

線形回帰分析を行い、学習によってこの式が算出された

この式をもとに、`model.predict()`で70歳の時は11.24本と予測された

2)回帰直線の作図 をコピーしよう

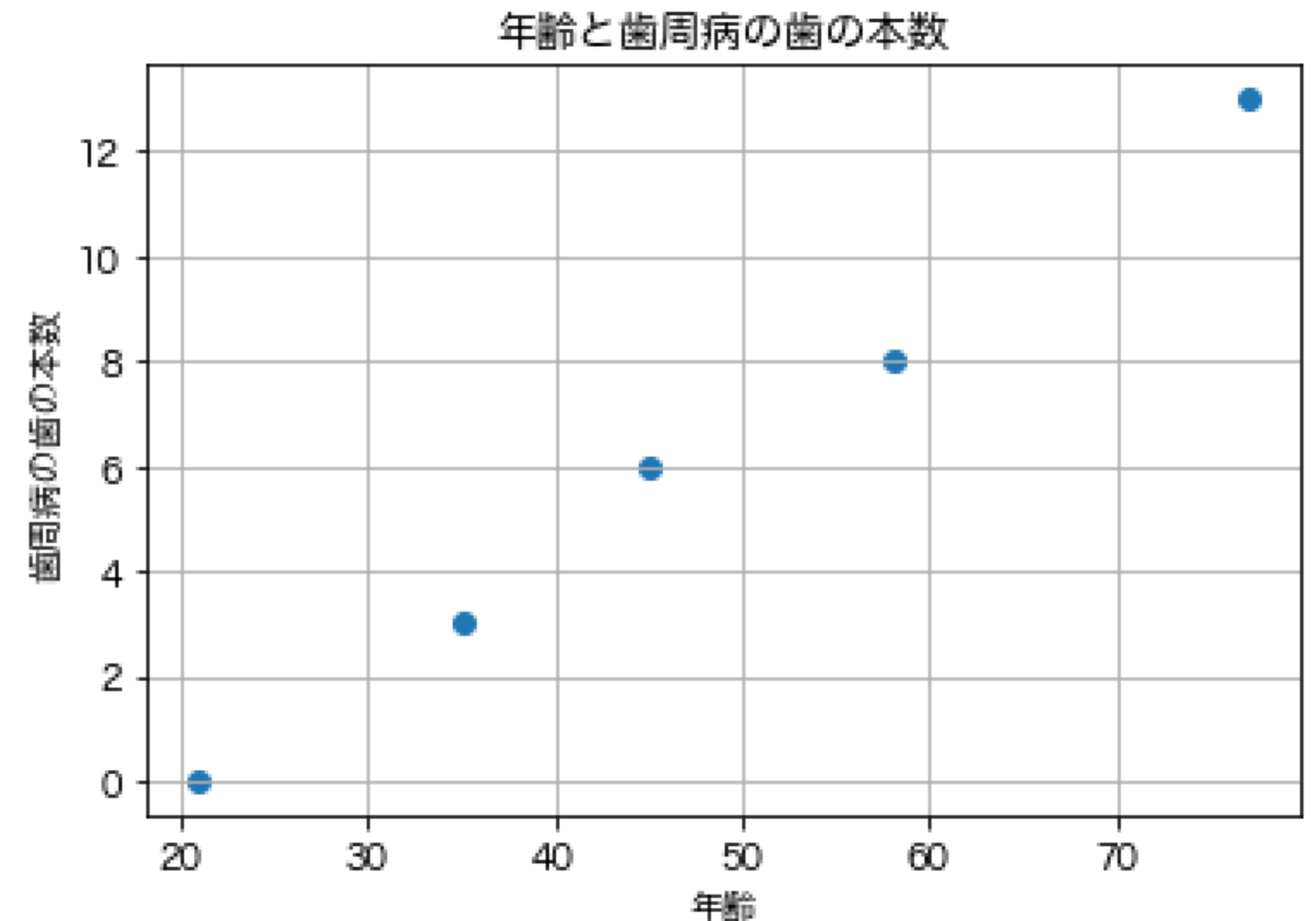
```
import matplotlib.pyplot as plt
from matplotlib import rcParams
rcParams['font.family'] = 'sans-serif'
rcParams['font.sans-serif'] = ['Hiragino Maru Gothic Pro', 'Yu Gothic', 'Meirio']
```

```
x = [[35],[21],[45],[58],[77]]
y = [3,0,6,8,13]
```

```
model = LinearRegression()
model.fit(x,y)
print(model.coef_)
print(model.intercept_)
test = [[70]]
num_teeth = model.predict(test)
print("70歳の時の本数は",num_teeth,"本")
```

```
plt.figure()
plt.title('年齢と歯周病の歯の本数')
plt.xlabel('年齢')
plt.ylabel('歯周病の歯の本数')
plt.grid(True)
plt.scatter(x,y)
plt.plot(x,model.predict(x))
plt.show()
```

赤字は前とほぼ一緒
(xが2次元配列になっているが図は変わらない)



2)回帰直線の作図 をコピーしよう

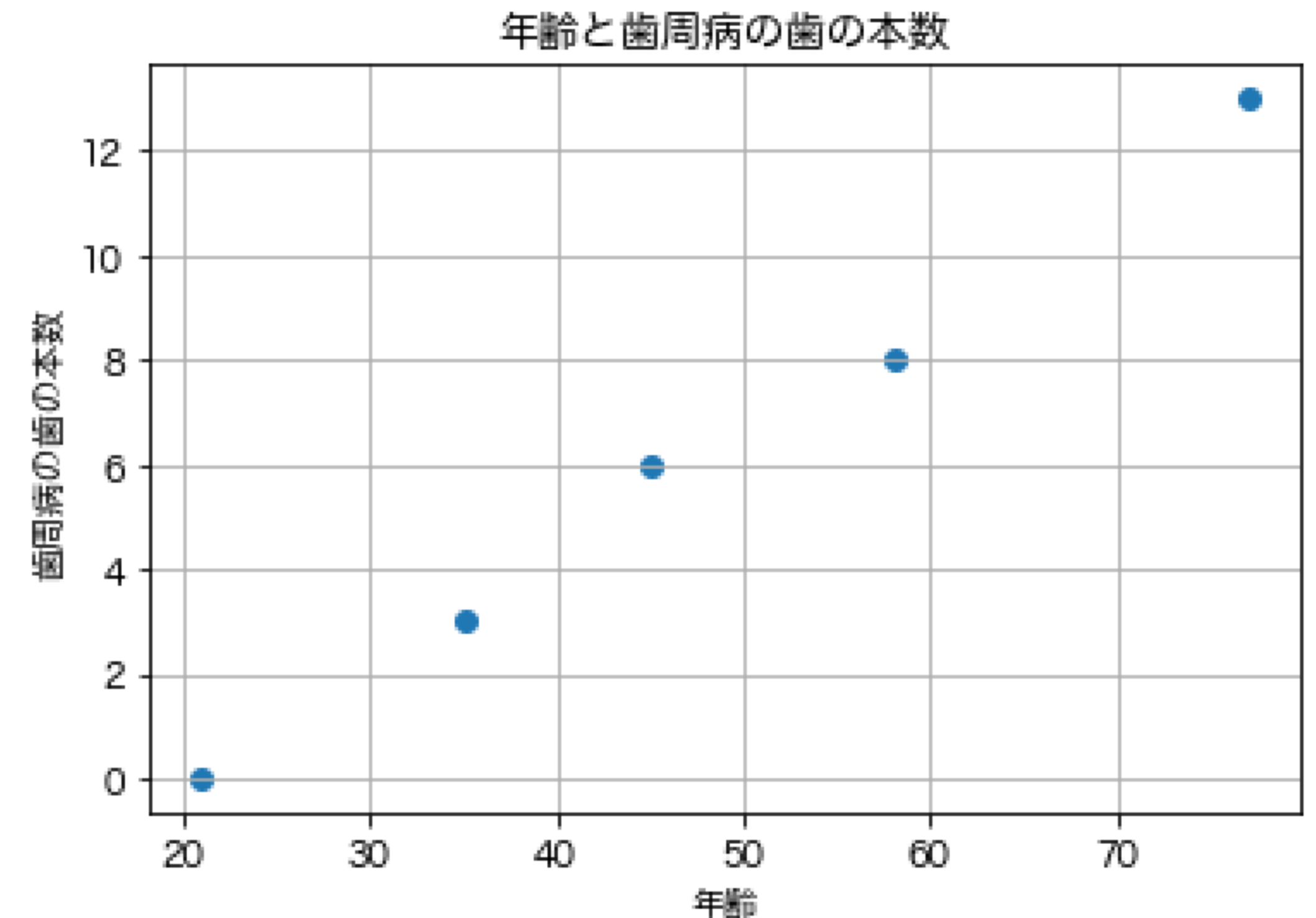
```
import matplotlib.pyplot as plt
from matplotlib import rcParams
rcParams['font.family'] = 'sans-serif'
rcParams['font.sans-serif'] = ['Hiragino Maru Gothic Pro', 'Yu Gothic', 'Meirio']
```

```
x = [[35],[21],[45],[58],[77]]
y = [3,0,6,8,13]
```

```
model = LinearRegression()
model.fit(x,y)
print(model.coef_)
print(model.intercept_)
test = [[70]]
num_teeth = model.predict(test)
print("70歳の時の本数は",num_teeth,"本")
```

```
plt.figure()
plt.title('年齢と歯周病の歯の本数')
plt.xlabel('年齢')
plt.ylabel('歯周病の歯の本数')
plt.grid(True)
plt.scatter(x,y)
plt.plot(x,model.predict(x))
plt.show()
```

青字は 1)の線形回帰
これだけだと図には関係ない



2)回帰直線の作図 をコピーしよう

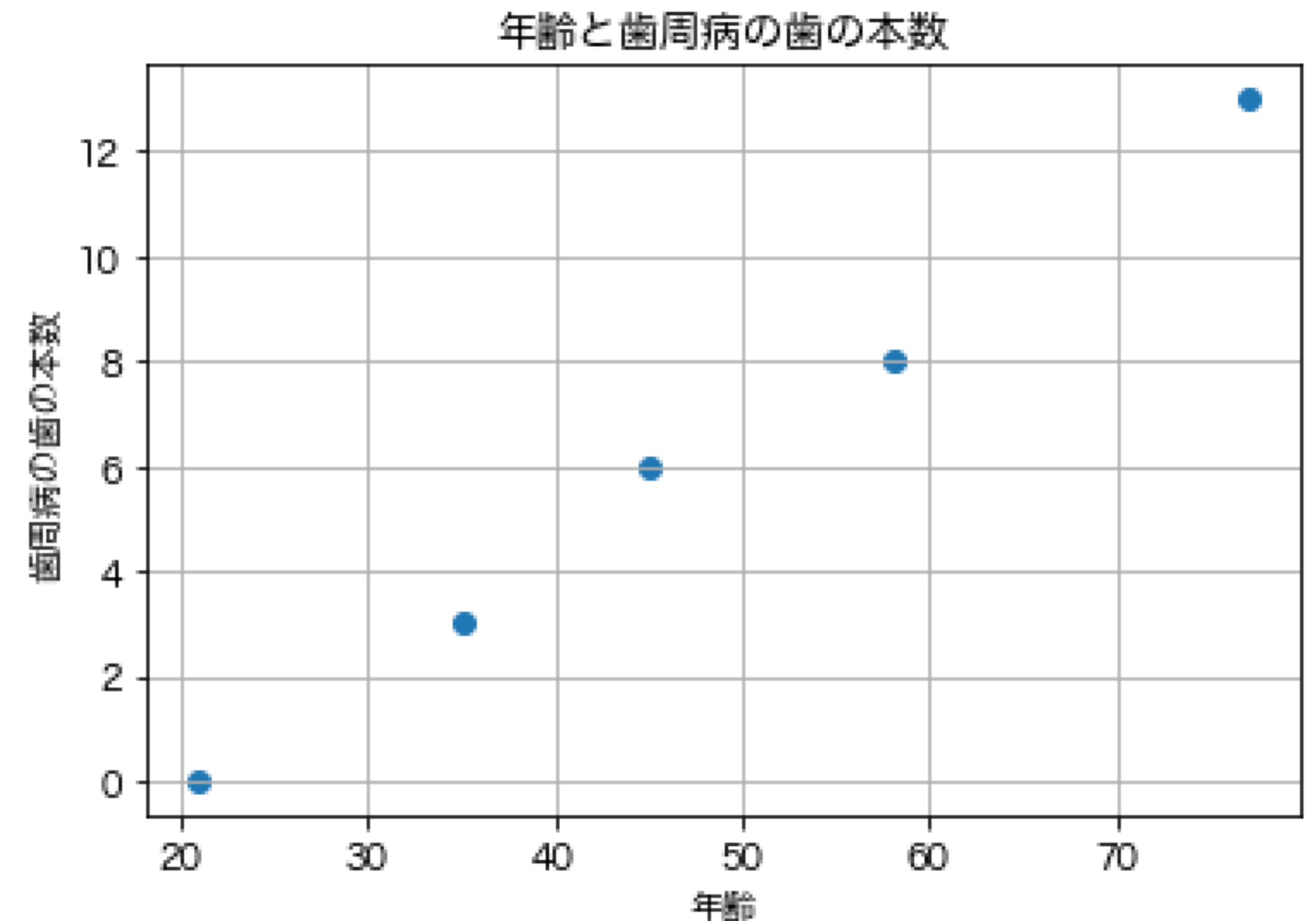
```
import matplotlib.pyplot as plt
from matplotlib import rcParams
rcParams['font.family'] = 'sans-serif'
rcParams['font.sans-serif'] = ['Hiragino Maru Gothic Pro', 'Yu Gothic', 'Meirio']

x = [[35],[21],[45],[58],[77]]
y = [3,0,6,8,13]

model = LinearRegression()
model.fit(x,y)
print(model.coef_)
print(model.intercept_)
test = [[70]]
num_teeth = model.predict(test)
print("70歳の時の本数は",num_teeth,"本")

plt.figure()
plt.title('年齢と歯周病の歯の本数')
plt.xlabel('年齢')
plt.ylabel('歯周病の歯の本数')
plt.grid(True)
plt.scatter(x,y)
plt.plot(x,model.predict(x))
plt.show()
```

作図plt.～が一行増えている(緑)
plt.plot(x,model.predict(x))



2)回帰直線の作図 をコピーしよう

```
x = [[35],[21],[45],[58],[77]]  
y = [3,0,6,8,13]
```

```
plt.scatter(x,y)
```

x軸y軸に

35と3、21と0、45と6、58と8、77と13
の点を打つ(散布図といいます)

```
plt.plot(x,model.predict(x))
```

x軸y軸に

35とmodel.predict([[35]])、
21とmode.predict([[21]])、
45とmodel.predict([[45]])、
58とmodel.predict([[58]])、
77とmodel.predict([[77]])

を通る直線(or曲線)を書く

← x=[[35]]の時のmodelが予測した値
← x=[[21]]の時のmodelが予測した値
← x=[[45]] ・ ・ ・
← x=[[58]] ・ ・ ・
← x=[[77]] ・ ・ ・

2)回帰直線の作図 をコピーしよう

2) 回帰直線の作図

```
x = [[35],[21],[45],[58],[77]]
y = [3,0,6,8,13]

model = LinearRegression()
model.fit(x,y)

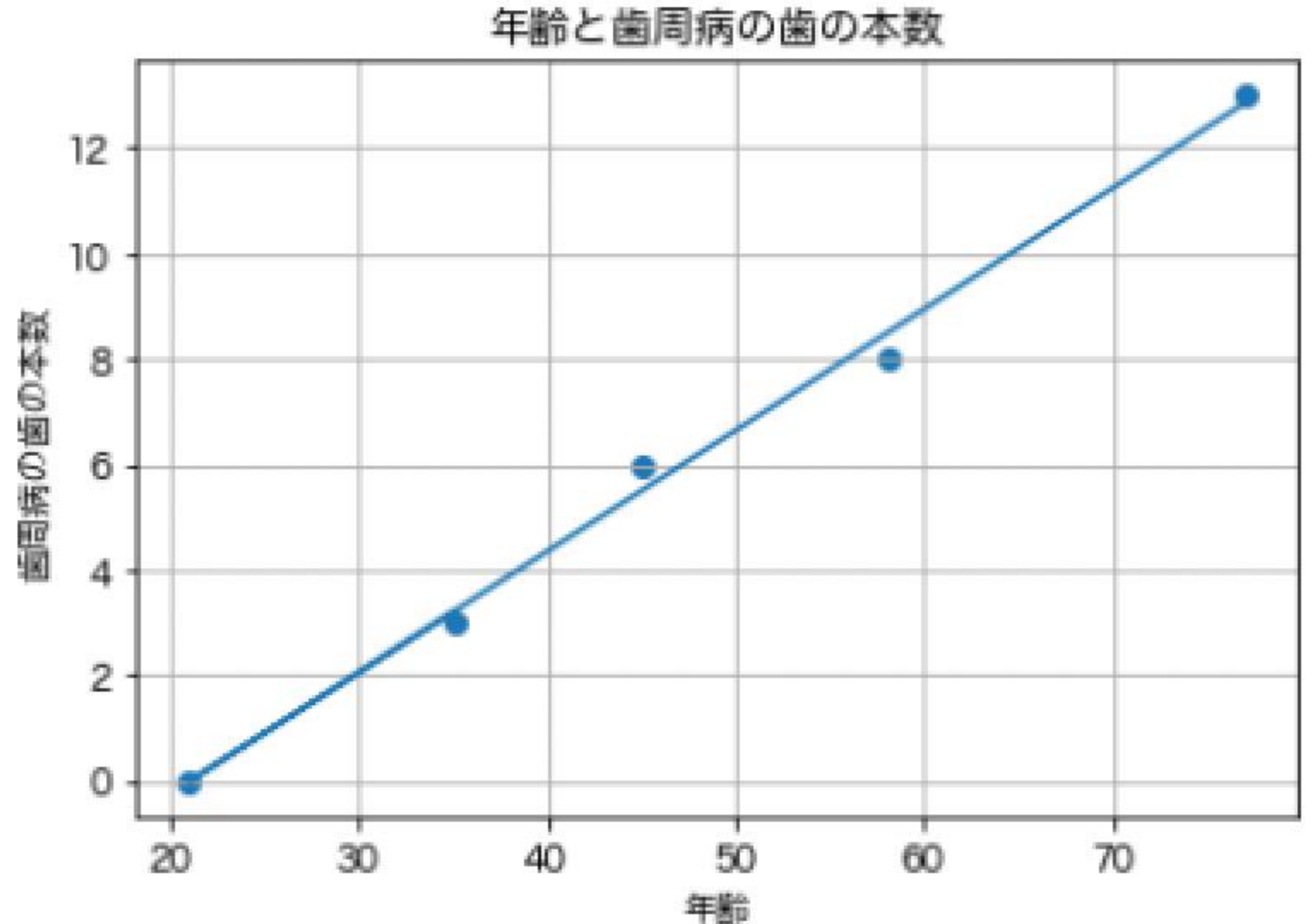
print(model.coef_)
print(model.intercept_)

test = [[70]]
num_teeth = model.predict(test)
print("70歳の時の本数は",num_teeth,"本")

plt.figure()
plt.title('年齢と歯周病の歯の本数')
plt.xlabel('年齢')
plt.ylabel('歯周病の歯の本数')
plt.grid(True)
plt.scatter(x,y)

plt.plot(x,model.predict(x))

plt.show()
```



回帰直線と予測した値を作図する

3) 回帰直線と予測した値の作図

```
x = [[35],[21],[45],[58],[77]]
y = [3,0,6,8,13]

model = LinearRegression()
model.fit(x,y)

print(model.coef_)
print(model.intercept_)

test = [[70]]
num_teeth = model.predict(test)
print("70歳の時の本数は",num_teeth,"本")

plt.figure()
plt.title('年齢と歯周病の歯の本数')
plt.xlabel('年齢')
plt.ylabel('歯周病の歯の本数')
plt.grid(True)
plt.scatter(x,y)
plt.plot(x,model.predict(x))
plt.scatter(test,num_teeth)
plt.show()
```

plt.scatter(test,num_teeth)

x軸にtest (= [[70]])

y軸にnum_teeth (=model.predict([[70]])
の点を打つ

回帰直線と予測した値を作図する

3) 回帰直線と予測した値の作図

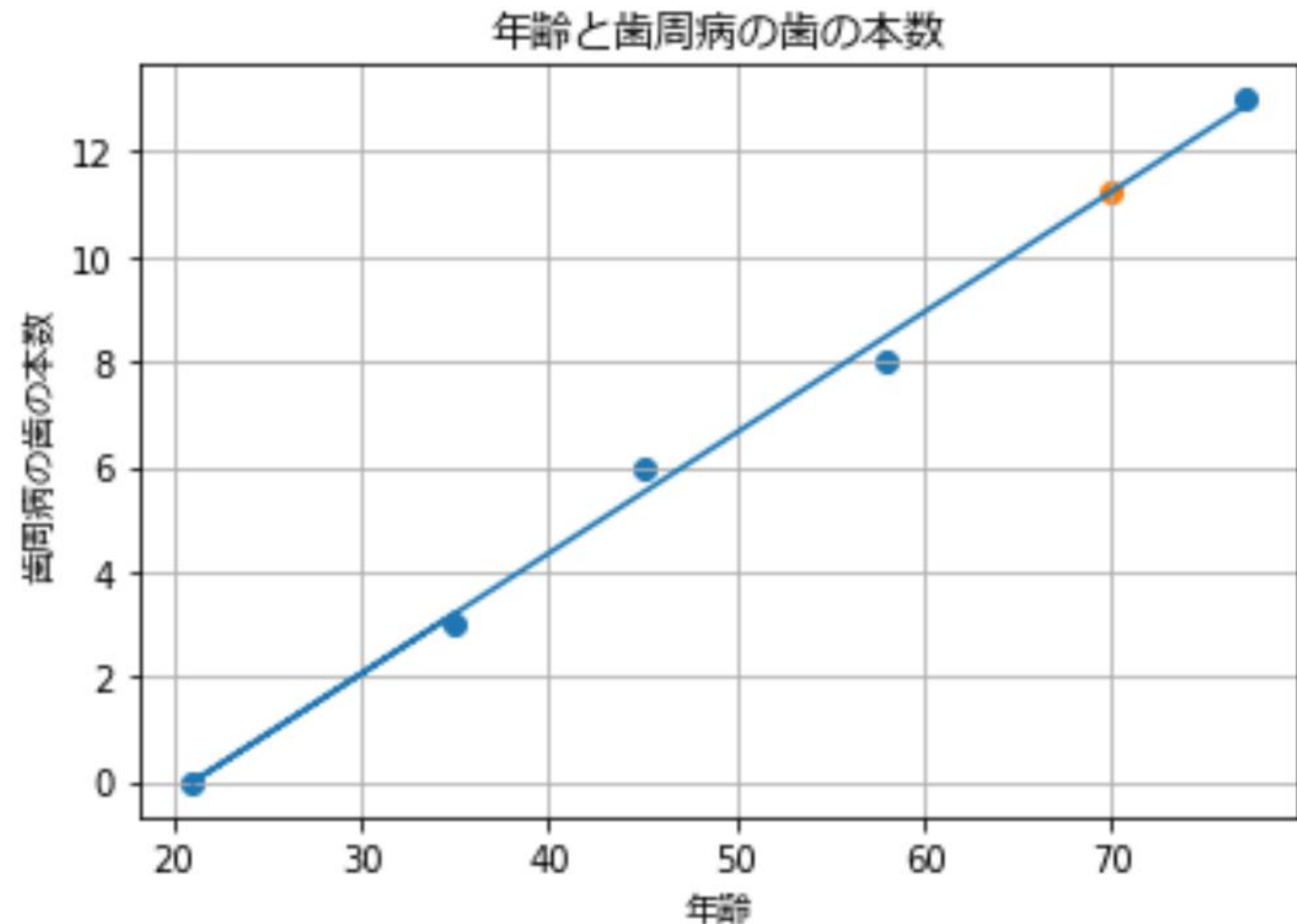
```
x = [[35],[21],[45],[58],[77]]
y = [3,0,6,8,13]

model = LinearRegression()
model.fit(x,y)

print(model.coef_)
print(model.intercept_)

test = [[70]]
num_teeth = model.predict(test)
print("70歳の時の本数は",num_teeth,"本")

plt.figure()
plt.title('年齢と歯周病の歯の本数')
plt.xlabel('年齢')
plt.ylabel('歯周病の歯の本数')
plt.grid(True)
plt.scatter(x,y)
plt.plot(x,model.predict(x))
plt.scatter(test,num_teeth)
plt.show()
```



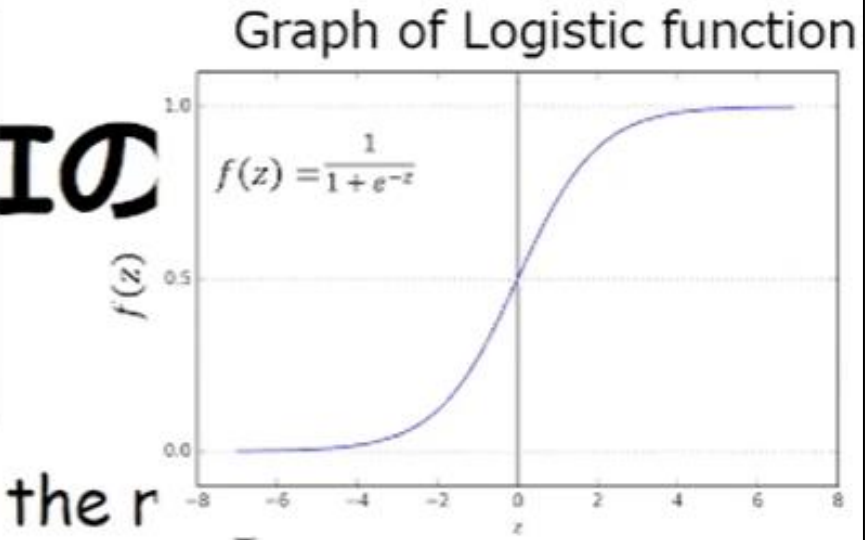
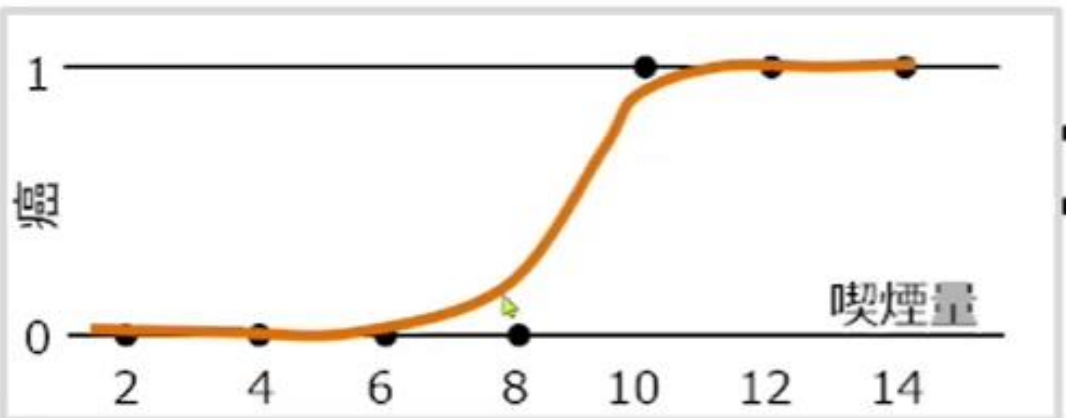
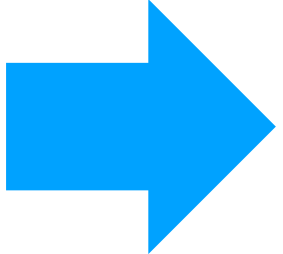
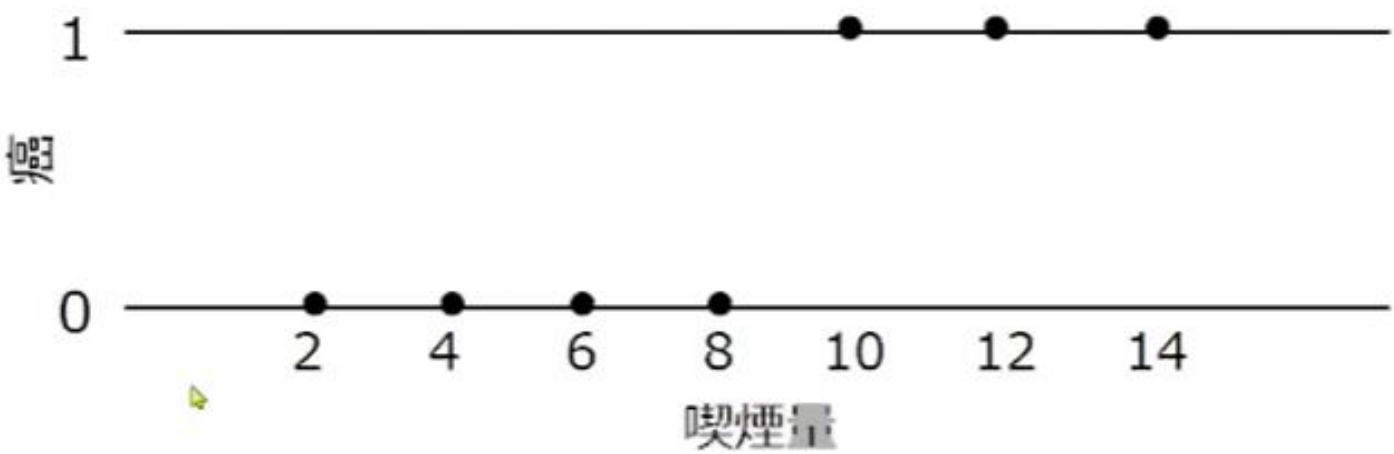
ロジスティック回帰分析にトライしよう

パーク先生の講義より

Data Science・AIの基礎

• Example : 喫煙量と肺癌

喫煙量 X	2	4	6	8	10	12	14
癌: Y	0	0	0	0	1	1	1
Probability of cancer (π)	0.02	0.10	0.15	0.23	0.62	0.68	0.73



• Model outputs the π values in the r

• Logistic function

$$f(z) = \frac{1}{1 + e^{-z}}$$

where $0 < f(z) < 1$ for $-\infty < z < \infty$.

• Logistic regression model

$$p(y = 1|x) = \pi = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)'}}$$

where $0 < \pi < 1$, $-\infty < x < \infty$, $y = 0,1$

まずはデータを加工する

データを読み込む(2.csv)

あやめのデータ(2.csv)



Iris Versicolor
ブルーフラッグ



Iris Setosa
ヒオウギアヤメ

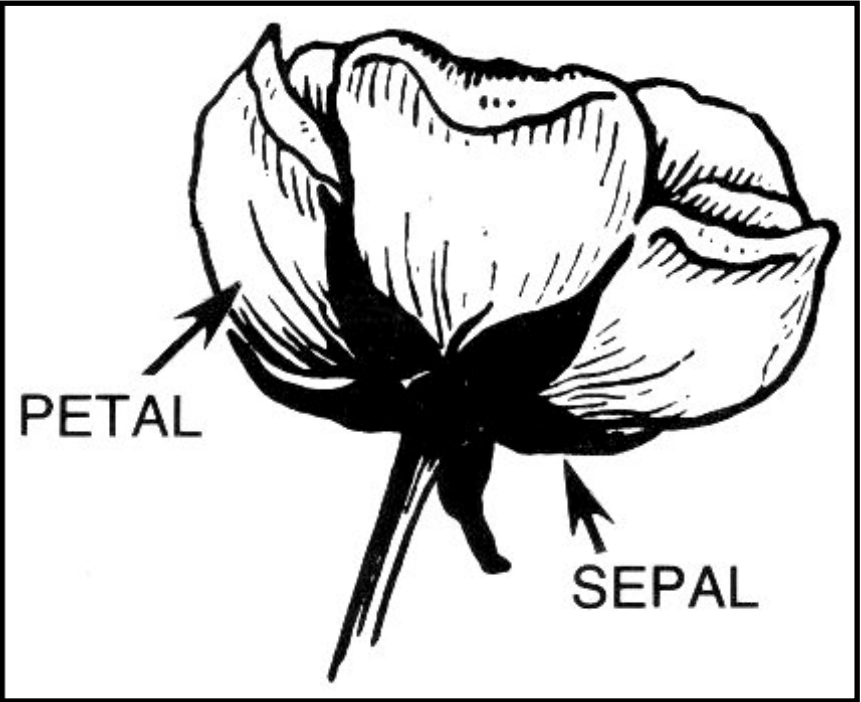


Iris Virginica
バージニカ

150行 × 6列

変数4つ
Sepal(がく片)の長さ と 幅
Petal(花びら)の長さ と 幅

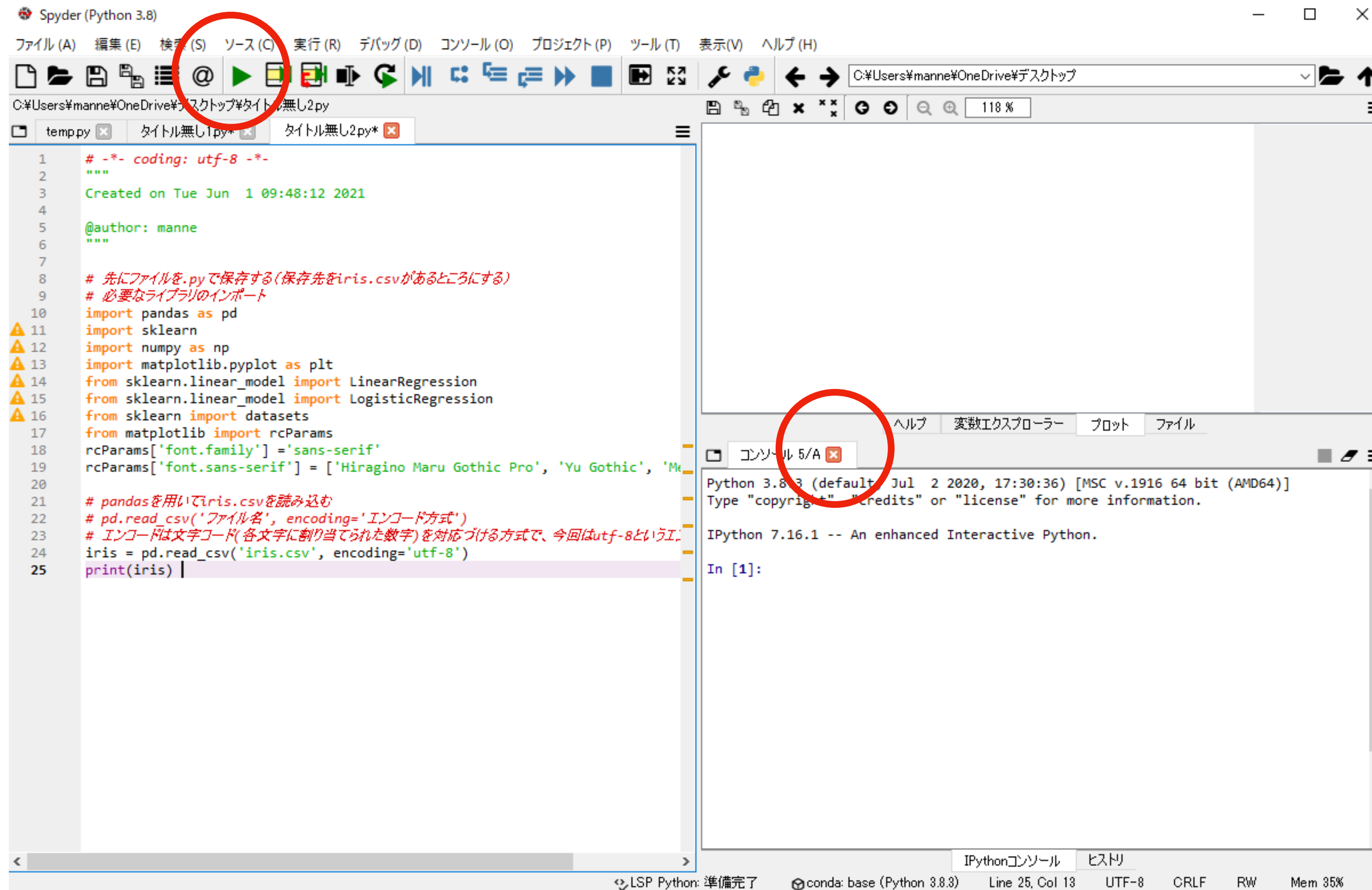
正解が3種類
ヒオウギアヤメ(0)
ブルーフラッグ(1)
バージニカ(2)



	A	B	C	D	E	F
1	がく片の長さ	がく片の幅	花びらの長さ	花びらの幅	アヤメの種類_数字	アヤメの種類
2	5.1	3.5	1.4	0.2	0	ヒオウギアヤメ
3	4.9	3	1.4	0.2	0	ヒオウギアヤメ
4	4.7	3.2	1.3	0.2	0	ヒオウギアヤメ
5	4.6	3.1	1.5	0.2	0	ヒオウギアヤメ
6	5	3.6	1.4	0.2	0	ヒオウギアヤメ
7	5.4	3.9	1.7	0.4	0	ヒオウギアヤメ
8	4.6	3.4	1.4	0.3	0	ヒオウギアヤメ
9	5	3.4	1.5	0.2	0	ヒオウギアヤメ
10	4.4	2.9	1.4	0.2	0	ヒオウギアヤメ
11	4.9	3.1	1.5	0.1	0	ヒオウギアヤメ
12	5.4	3.7	1.5	0.2	0	ヒオウギアヤメ
⋮						
50	5.3	3.7	1.5	0.2	0	ヒオウギアヤメ
51	5	3.3	1.4	0.2	0	ヒオウギアヤメ
52	7	3.2	4.7	1.4	1	ブルーフラッグ
53	6.4	3.2	4.5	1.5	1	ブルーフラッグ
54	6.9	3.1	4.9	1.5	1	ブルーフラッグ
55	5.5	2.3	4	1.3	1	ブルーフラッグ
56	6.5	2.8	4.6	1.5	1	ブルーフラッグ
57	5.7	2.8	4.5	1.3	1	ブルーフラッグ
58	6.3	3.3	4.7	1.6	1	ブルーフラッグ
⋮						
141	6.9	3.1	5.4	2.1	2	バージニカ
142	6.7	3.1	5.6	2.4	2	バージニカ
143	6.9	3.1	5.1	2.3	2	バージニカ
144	5.8	2.7	5.1	1.9	2	バージニカ
145	6.8	3.2	5.9	2.3	2	バージニカ
146	6.7	3.3	5.7	2.5	2	バージニカ
147	6.7	3	5.2	2.3	2	バージニカ
148	6.3	2.5	5	1.9	2	バージニカ
149	6.5	3	5.2	2	2	バージニカ
150	6.2	3.4	5.4	2.3	2	バージニカ
151	5.9	3	5.1	1.8	2	バージニカ

まずはデータを読み込んでみよう

エディタの内容を消して、コンソールもリセットして
4)アイリスデータを読み込む”をコピーして貼り付ける



まずはデータを読み込んでみよう

エラーが表示されず、変数エクスペローラーにirisが入っていれば読み込み完了です

The screenshot shows the Spyder Python IDE interface. The left pane displays a Python script with the following code:

```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Tue Jun  1 09:48:12 2021
4
5  @author: manne
6  """
7
8  # 先にファイルを.pyで保存する(保存先をiris.csvがあるところにする)
9  # 必要なライブラリのインポート
10 import pandas as pd
11 import sklearn
12 import numpy as np
13 import matplotlib.pyplot as plt
14 from sklearn.linear_model import LinearRegression
15 from sklearn.linear_model import LogisticRegression
16 from sklearn import datasets
17 from matplotlib import rcParams
18 rcParams['font.family'] = 'sans-serif'
19 rcParams['font.sans-serif'] = ['Hiragino Maru Gothic Pro', 'Yu Gothic', 'Me
20
21 # pandasを用いてiris.csvを読み込む
22 # pd.read_csv('ファイル名', encoding='エンコード方式')
23 # エンコードは文字コード(各文字に割り当てられた数字)を対応づける方式で、今回はutf-8というエ
24 iris = pd.read_csv('iris.csv', encoding='utf-8')
25 print(iris)
```

The right pane shows the Variable Explorer, which is circled in red. It displays the variable 'iris' as a DataFrame with a size of (150, 6). The column names are listed as: ながさ, 幅, ながさ, 幅, 種類_数字, 種類.

The bottom pane shows the IPython console output, which displays the first 150 rows of the iris dataset (150 rows x 6 columns). The output is as follows:

```
In [1]: runfile('C:/Users/manne/OneDrive/デスクトップ/タイトル無し2.py', wdir='C:/Users/
manne/OneDrive/デスクトップ')
      ながさ  幅  ながさ  幅  種類_数字  種類
0      5.1   3.5   1.4  0.2         0  ヒオウギアヤメ
1      4.9   3.0   1.4  0.2         0  ヒオウギアヤメ
2      4.7   3.2   1.3  0.2         0  ヒオウギアヤメ
3      4.6   3.1   1.5  0.2         0  ヒオウギアヤメ
4      5.0   3.6   1.4  0.2         0  ヒオウギアヤメ
...    ...   ...   ...   ...   ...   ...
145     6.7   3.0   5.2  2.3         2   バージニカ
146     6.3   2.5   5.0  1.9         2   バージニカ
147     6.5   3.0   5.2  2.0         2   バージニカ
148     6.2   3.4   5.4  2.3         2   バージニカ
149     5.9   3.0   5.1  1.8         2   バージニカ

[150 rows x 6 columns]

In [2]:
```

まずはデータを読み込んでみよう

1)アイリスデータを読み込む

```
import pandas as pd
import sklearn
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn import datasets
from matplotlib import rcParams
rcParams['font.family'] = 'sans-serif'
rcParams['font.sans-serif'] = ['Hiragino Maru Gothic Pro', 'Yu Gothic', 'Meirio']
```

```
# pandasを用いてiris.csvを読み込む
# pd.read_csv('ファイル名', encoding='エンコード方式')
# エンコードは文字コード(各文字に割り当てられた数字)を対応づける方式で、今回はutf-8というエンコード方式を選ぶ
iris = pd.read_csv('iris.csv', encoding='utf-8')
print(iris)
```

skleanというライブラリを
読み込んでます。

sklearn(scikit-learn)は機械学習の機能を
多く持ったライブラリです。

まずはデータを読み込んでみよう

pandasというライブラリをインストールしてpdと省略して使います

```
import pandas as pd
```

pd.read_csv('ファイル名.csv', encoding='utf-8')でファイルをpandasの形式で読み込みます

```
iris = pd.read_csv('2.csv', encoding='utf-8')
```

pandasで読み込んだデータの型はデータフレームでした。
今回は読み込んだ 2.csvのデータをirisという変数名で格納しています。

(エンコードは文字コード(各文字に割り当てられた数字)を対応づける方式で、
今回はutf-8というエンコード方式を選ぶ)

変数エクスプローラーのirisをダブルクリックするとデータフレームをみることが出来る

iris - DataFrame

インデックス	がく片の長さ	がく片の幅	てびらの長さ	花びらの幅	メの種類_数字	アヤメの種類
0	5.1	3.5	1.4	0.2	0	ヒオウギアヤメ
1	4.9	3	1.4	0.2	0	ヒオウギアヤメ
2	4.7	3.2	1.3	0.2	0	ヒオウギアヤメ
3	4.6	3.1	1.5	0.2	0	ヒオウギアヤメ
4	5	3.6	1.4	0.2	0	ヒオウギアヤメ
5	5.4	3.9	1.7	0.4	0	ヒオウギアヤメ
6	4.6	3.4	1.4	0.3	0	ヒオウギアヤメ
7	5	3.4	1.5	0.2	0	ヒオウギアヤメ
8	4.4	2.9	1.4	0.2	0	ヒオウギアヤメ
9	4.9	3.1	1.5	0.1	0	ヒオウギアヤメ
10	5.4	3.7	1.5	0.2	0	ヒオウギアヤメ
11	4.8	3.4	1.6	0.2	0	ヒオウギアヤメ
12	4.8	3	1.4	0.1	0	ヒオウギアヤメ
13	4.3	3	1.1	0.1	0	ヒオウギアヤメ
14	5.8	4	1.2	0.2	0	ヒオウギアヤメ
15	5.7	4.4	1.5	0.4	0	ヒオウギアヤメ

フォーマット リサイズ ☒ 背景色 ☒ 列の 最小/最大

変数エクスプローラーのirisをダブルクリックするとデータフレームをみることが出来る

iris DataFrame (150, 6)

Column names: がく片の長さ, がく片の幅, てびらの長さ, 花びらの幅, アヤメの種類_数字, アヤメの種類

ヘルプ 変数エクスプローラー プロット ファイル

コンソール 5/A

Type "copyright", "credits" or "license" for more information.

IPython 7.16.1 -- An enhanced Interactive Python.

```
In [1]: runfile('C:/Users/manne/OneDrive/デスクトップ/タイトル無し2.py', wdir='C:/Users/manne/OneDrive/デスクトップ')
がく片の長さ がく片の幅 てびらの長さ 花びらの幅 アヤメの種類_数字 アヤメの種類
0      5.1      3.5      1.4      0.2      0      ヒオウギアヤメ
1      4.9      3.0      1.4      0.2      0      ヒオウギアヤメ
2      4.7      3.2      1.3      0.2      0      ヒオウギアヤメ
3      4.6      3.1      1.5      0.2      0      ヒオウギアヤメ
4      5.0      3.6      1.4      0.2      0      ヒオウギアヤメ
...      ...      ...      ...      ...      ...      ...
145     6.7      3.0      5.2      2.3      2      パージニカ
146     6.3      2.5      5.0      1.9      2      パージニカ
147     6.5      3.0      5.2      2.0      2      パージニカ
148     6.2      3.4      5.4      2.3      2      パージニカ
149     5.9      3.0      5.1      1.8      2      パージニカ

[150 rows x 6 columns]
```

In [2]:

LSP Python: 準備完了 conda: base (Python 3.8.3) Line 25, Col 13 UTF-8 CRLF RW Mem 35%

ロジスティック回帰でアヤメを分類する

がく 片の長さでヒオウギアヤメとブルーフラッグを分類する

① 学習モデルの選択(今回は線形回帰)

(モデル名) = `LinearRegression()`

② データを入れて学習させる

(モデル名).`fit`(説明変数, 目的変数)

③ 傾き(偏回帰係数)と切片(定数項)を求める

(モデル名).`coef_` #傾き

(モデル名).`intercept_` #切片

④ 予測を行う

(モデル名).`predict`(新たな説明変数)

ロジスティック回帰でアヤメを分類する

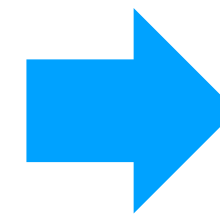
がく片の長さでヒオウギアヤメとブルーフラッグを分類する

① 学習モデルの選択(今回は線形回帰)
(モデル名) = `LinearRegression()`

② データを入れて学習させる
(モデル名).`fit`(説明変数, 目的変数)

③ 傾き(偏回帰係数)と切片(定数項)を求める
(モデル名).`coef_` #傾き
(モデル名).`intercept_` #切片

④ 予測を行う
(モデル名).`predict`(新たな説明変数)



① 学習モデルの選択(今回はロジスティック回帰)
(モデル名) = `LogisticRegression()`

② データを入れて学習させる
(モデル名).`fit`(説明変数, 目的変数)

③ 傾き(偏回帰係数)と切片(定数項)を求める
(モデル名).`coef_` #傾き
(モデル名).`intercept_` #切片

④ 予測を行う
(モデル名).`predict`(新たな説明変数)
(モデル名).`predict_proba`(新たな説明変数)

5) 学習に用いる説明変数と目的変数を設定する

説明変数(目的を知るために使用する変数) : がく 片の長さ → x

目的変数(目的(分類や予測)となる変数) : アヤメの種類_数字 → y

```
df = iris[0:100]  
x = df['がく 片の長さ']  
y = df['アヤメの種類_数字']  
print(x)  
print(y)
```

irisの1行目から100行目まで(ヒオウギアヤメとブルーフラッグ)を取り出して、dfという名前の変数に代入

5) 学習に用いる説明変数と目的変数を設定する

説明変数(目的を知るために使用する変数) : がく 片の長さ → x

目的変数(目的(分類や予測)となる変数) : アヤメの種類_数字 → y

```
df = iris[0:100]  
x = df['がく 片の長さ']  
y = df['アヤメの種類_数字']  
print(x)  
print(y)
```

xに"がく 片の長さ"、yに"アヤメの種類_数字"の列の内容を代入

5) 学習に用いる説明変数と目的変数を設定する

説明変数(目的を知るために使用する変数) : がく 片の長さ → x

目的変数(目的(分類や予測)となる変数) : アヤメの種類_数字 → y

```
df = iris[0:100]  
x = df['がく 片の長さ']  
y = df['アヤメの種類_数字']  
print(x)  
print(y)
```

xに"がく 片の長さ"、yに"アヤメの種類_数字"の列の内容を代入

```
0    5.1  
1    4.9  
2    4.7  
3    4.6  
4    5.0  
  
95    5.7  
96    5.7  
97    6.2  
98    5.1  
99    5.7  
Name: がく 片の長さ, Length: 100, dtype: float64
```

```
0    0  
1    0  
2    0  
3    0  
4    0  
..  
95    1  
96    1  
97    1  
98    1  
99    1  
Name: アヤメの種類_数字, Length: 100, dtype: int64
```

5) 学習に用いる説明変数と目的変数を設定する

今回はxが"がく 片の長さ"、yが"アヤメの種類_数字"なので
モデル名をmodel2とすれば、

① 学習モデルの選択

```
model2 = LogisticRegression()
```

② データを入れて学習させる

```
model2.fit(x,y)
```

```
x = df['がく 片の長さ']
```

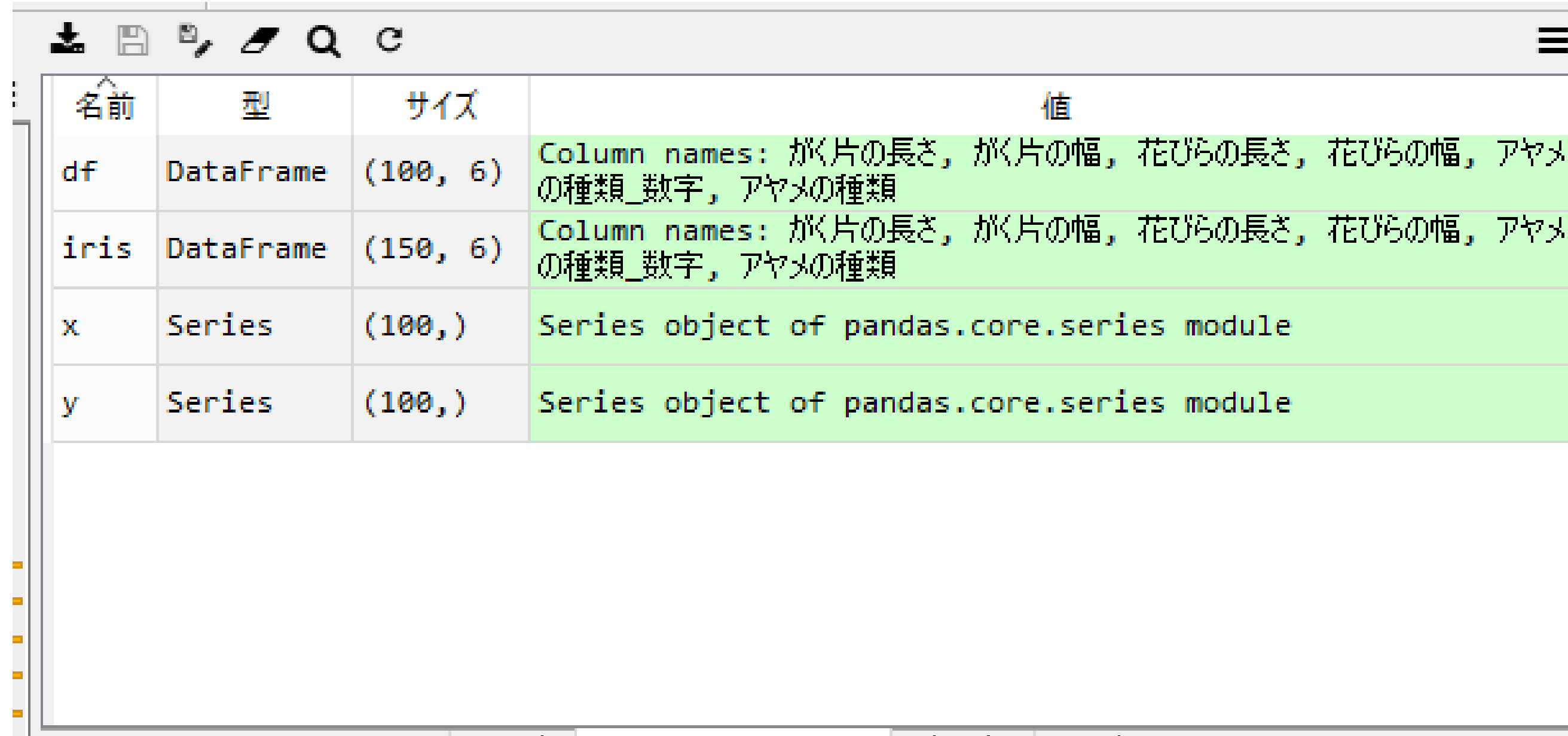
```
y = df['アヤメの種類_数字']
```

このままではうまくいかない!?

このfit()に入れる説明変数は2次元配列(行列の形)でなければならないというルールがあるので、

このxを2次元配列に変換する必要がある

5) 学習に用いる説明変数と目的変数を設定する



The screenshot shows a Jupyter Notebook interface with a table of variables. The table has four columns: '名前' (Name), '型' (Type), 'サイズ' (Size), and '値' (Value). The rows are as follows:

名前	型	サイズ	値
df	DataFrame	(100, 6)	Column names: がく片の長さ, がく片の幅, 花びらの長さ, 花びらの幅, アヤメの種類_数字, アヤメの種類
iris	DataFrame	(150, 6)	Column names: がく片の長さ, がく片の幅, 花びらの長さ, 花びらの幅, アヤメの種類_数字, アヤメの種類
x	Series	(100,)	Series object of pandas.core.series module
y	Series	(100,)	Series object of pandas.core.series module

データフレームを1列取り出すとSeries型という1次元配列になる

このxを2次元配列に変換する必要がある

pandas(前回のスライド)

pandasではデータフレームを使用してみます

```
変数 = pd.DataFrame(データ)
```

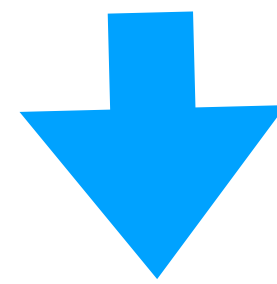
データは自分で作ることも出来ますが、外から読み込むことも出来ます。

	体重	身長	年齢
Aさん	40	160	20
Bさん	55	170	45
Cさん	62	175	38

データフレームはこの形状でデータを扱うことが出来る

5) 学習に用いる説明変数と目的変数を設定する

```
x = df['がく 片の長さ']  
y = df['アヤメの種類_数字']
```



```
x = pd.DataFrame(df['がく 片の長さ'])  
y = df['アヤメの種類_数字']
```

6) ヒオウギアヤメのがく片の長さとかく片の幅で線形回帰

xがデータフレームになっていることを確認

名前	型	サイズ	値
df	DataFrame	(100, 6)	Column names: がく片の長さ, がく片の幅, 花びらの長さ, 花びらの幅, アヤメの種類_数字, アヤメの種類
iris	DataFrame	(150, 6)	Column names: がく片の長さ, がく片の幅, 花びらの長さ, 花びらの幅, アヤメの種類_数字, アヤメの種類
x	DataFrame	(100, 1)	Column names: がく片の長さ
y	Series	(100,)	Series object of pandas.core.series module

6) ヒオウギアヤメのがく片の長さとかく片の幅で線形回帰

線形回帰と同様

```
x = pd.DataFrame(df['がく 片の長さ'])  
y = df['アヤメの種類_数字']  
  
model2 = LogisticRegression()  
model2.fit(x, y)  
  
print(model2.coef_)  
print(model2.intercept_)
```

#7) 作った学習モデルで分類をする

学習モデルを作ったのでpredictで予測する
ではがく 片の長さが4.5, 5, 7,0の時のブルーフラッグの確率は？

In

```
check1 = model2.predict([[4.5]])  
print(check1)  
check2 = model2.predict([[5.0]])  
print(check2)  
check3 = model2.predict([[7.0]])  
print(check3))
```

#7) 作った学習モデルで分類をする

学習モデルを作ったのでpredictで予測する
ではがく 片の長さが4.5, 5, 7,0の時のブルーフラッグの確率は？

In

```
check1 = model2.predict([[4.5]])  
print(check1)  
check2 = model2.predict([[5.0]])  
print(check2)  
check3 = model2.predict([[7.0]])  
print(check3))
```

Out

```
print(check1)  
[0]  
print(check2)  
[0]  
print(check3)  
[1]
```

ヒオウギアヤメ (=0)、ヒオウギアヤメ (=0)、ブルーフラッグ (=1)と予測された

#7) 作った学習モデルで分類をする

学習モデルを作ったのでpredictで予測する
ではがく 片の長さが4.5, 5, 7,0の時のブルーフラッグの確率は？

In

```
check1 = model2.predict([[4.5]])  
print(check1)  
check2 = model2.predict([[5.0]])  
print(check2)  
check3 = model2.predict([[7.0]])  
print(check3))
```

```
check4 = model2.predict_proba([[4.5]])  
print(check4)  
check5 = model2.predict_proba([[5.0]])  
print(check5)  
check6 = model2.predict_proba([[7.0]])  
print(check6))
```

(モデル名).predict_proba()で確率を計算

Out

```
print(check1)  
[0]  
print(check2)  
[0]  
print(check3)  
[1]
```

ヒオウギアヤメ (=0)、ヒオウギアヤメ (=0)、ブルーフラッグ (=1)と予測された

#7) 作った学習モデルで分類をする

学習モデルを作ったのでpredictで予測する
ではがく 片の長さが4.5, 5, 7,0の時のブルーフラッグの確率は？

In

```
check1 = model2.predict([[4.5]])  
print(check1)  
check2 = model2.predict([[5.0]])  
print(check2)  
check3 = model2.predict([[7.0]])  
print(check3))
```

```
check4 = model2.predict_proba([[4.5]])  
print(check4)  
check5 = model2.predict_proba([[5.0]])  
print(check5)  
check6 = model2.predict_proba([[7.0]])  
print(check6))
```

(モデル名).predict_proba()で確率を計算

Out

```
print(check1)  
[0]  
print(check2)  
[0]  
print(check3)  
[1]
```

```
print(check4)  
[[0.95036498 0.04963502]]  
print(check5)  
[[0.79665518 0.20334482]]  
print(check6)  
[[0.00682033 0.99317967]]
```

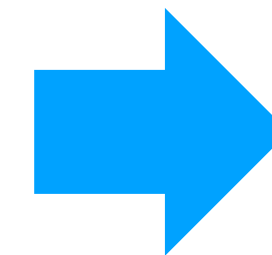
[[(ヒオウギアヤメの確率) (ブルーフラッグの確率)]]

ヒオウギアヤメ (=0)、ヒオウギアヤメ (=0)、ブルーフラッグ (=1)と予測された

#8) 説明変数を2つ使って分類する①

がく 片の長さ と 幅 の2つの変数だとどうなる？

```
x = pd.DataFrame(df['がく 片の長さ'])  
y = df['アヤメの種類_数字']  
  
model2 = LogisticRegression()  
model2.fit(x, y)  
  
print(model2.coef_)  
print(model2.intercept_)
```



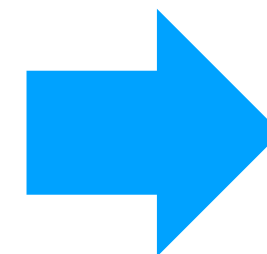
#8) 説明変数を2つ使って分類する①

がく 片の長さ と 幅 の2つの変数だとどうなる？

```
x = pd.DataFrame(df['がく 片の長さ'])  
y = df['アヤメの種類_数字']
```

```
model2 = LogisticRegression()  
model2.fit(x, y)
```

```
print(model2.coef_)  
print(model2.intercept_)
```



```
x = pd.DataFrame(df[['がく 片の長さ','がく 片の幅']])  
y = df['アヤメの種類_数字']
```

```
model2 = LogisticRegression()  
model2.fit(x, y)
```

```
print(model2.coef_)  
print(model2.intercept_)
```

#8) 説明変数を2つ使って分類する①

がく 片の長さ と 幅 の2つの変数だとどうなる？

列に対するデータフレームの操作

列名を指定して取得する場合は[]を用います。

データフレーム['列名']

複数の列を取得したい場合は、[]をもう1つ入れます

入力

```
print(df['体重'])
```

```
print(df[['体重','年齢']])
```

出力

```
0  40
1  55
2  62
3  66
4  55
5  80
Name: 体重, dtype: int64
```

```
   体重  年齢
0  40  20
1  55  45
2  62  38
3  66  40
4  55  52
5  80  30
```

```
x = pd.DataFrame(df[['がく 片の長さ','がく 片の幅']])
y = df['アヤメの種類_数字']
```

```
model2 = LogisticRegression()
model2.fit(x, y)
```

```
print(model2.coef_)
print(model2.intercept_)
```

#9) 説明変数を2つ使って分類する②

がく片の長さ と 幅が、(4.5, 3.2)、(5.0, 5.5)、(7.0, 6.0)の時 は？

9) 説明変数を2つ使って分類する②

がく 片の長さ と 幅が、(4.5 , 3.2)、(5.0 , 5.5)、(7.0, 6.0)の時は？

```
check7 = model2.predict_proba([[4.5,3.2]])  
print(check7)
```

```
check8 = model2.predict_proba([[5.0, 5.5]])  
print(check8)
```

```
check9 = model2.predict_proba([[7.0, 6.0]])  
print(check9)
```

9) 説明変数を2つ使って分類する②

がく 片の長さ と 幅が、(4.5 , 3.2)、(5.0 , 5.5)、(7.0, 6.0)の時は？

```
check7 = model2.predict_proba([[4.5,3.2]])  
print(check7)  
[[0.95781722 0.04218278]]  
  
check8 = model2.predict_proba([[5.0, 5.5]])  
print(check8)  
[[9.99803356e-01 1.96643916e-04]]  
  
check9 = model2.predict_proba([[7.0, 6.0]])  
print(check9)  
[[0.97991578 0.02008422]]
```

9.99803356e-01 は $9.99803356 \times 10^{-1}$

まとめ

機械学習の一步目として、 線形回帰とロジスティック回帰を実践しました

①学習モデルの選択(今回は線形回帰)
(モデル名) = `LinearRegression()`

②データを入れて学習させる
(モデル名).`fit`(説明変数,目的変数)

③傾き(偏回帰係数)と切片(定数項)を求める
(モデル名).`coef_` #傾き
(モデル名).`intercept_` #切片

④予測を行う
(モデル名).`predict`(新たな説明変数)

①学習モデルの選択(今回はロジスティック回帰)
(モデル名) = `LogisticRegression()`

②データを入れて学習させる
(モデル名).`fit`(説明変数,目的変数)

③傾き(偏回帰係数)と切片(定数項)を求める
(モデル名).`coef_` #傾き
(モデル名).`intercept_` #切片

④予測を行う
(モデル名).`predict`(新たな説明変数)
(モデル名).`predict_proba`(新たな説明変数)

次回はもっと多くの機械学習に触れてみたいと思います

他の説明変数でも試してみよう

ヒオウギアヤメとブルーフラッグのデータを用いて、
花びらの長さと幅を説明変数としてロジスティック回帰を行い、
花びらの長さが3.2、花びらの幅が1.1の時の
ブルーフラッグである確率を求めよう

ブルーフラッグとバージニカのデータを用いて、
花びらの長さと幅を説明変数としてロジスティック回帰を行い、
花びらの長さが3.2、花びらの幅が1.1の時の
ブルーフラッグである確率を求めよう