

演習授業中の質問対応について

Zoom ミーティング

表示

ミーティング チャット

演習授業中の質問をチューターの先生方が対応させていただきます。

演習にエラーが出たなど問題があったらリアクションの**挙手**を押してください。

質問内容を入力して、「**全員**」宛てに送信してください。

Miho Ishimaru

ここにメッセージは誰に表示されますか？

宛先: **全員** v

ここにメッセージを入力します...

ミュート解除

ビデオの開始

セキュリティ

参加者 2

画面共有

リアクション

アプリ

ホワイトボード

ノート

詳細

終了

演習7

データクレンジングとデータの可視化

統合教育機構 曹 日丹

医療とAI・ビッグデータ入門

演習2-7の構成

Python基礎を学びましょう

演習2

Pythonの変数とデータの型

演習3

プログラミング基礎

演習4

モジュール、パッケージ、ライブラリ

Pythonを使ってみましょう

演習5

患者の歯に関する~~病院のリアルデータ~~の説明

架空データ

演習6 12/21 10:40-11:35

データクレンジングに必要なライブラリ（Pandas）の応用

演習7 12/21 11:35-12:20

データクレンジングとデータの可視化

検索google colab Colaboratory へようこそ - Colaboratory - Google

Colaboratory へようこそ
ファイル 編集 表示

目次

- はじめに
- データサイエンス
- 機械学習
- その他のリソース
- 使用例
- セクション

ノートブックを開く

例 >

最近 >

Google ドライブ >

GitHub >

アップロード >

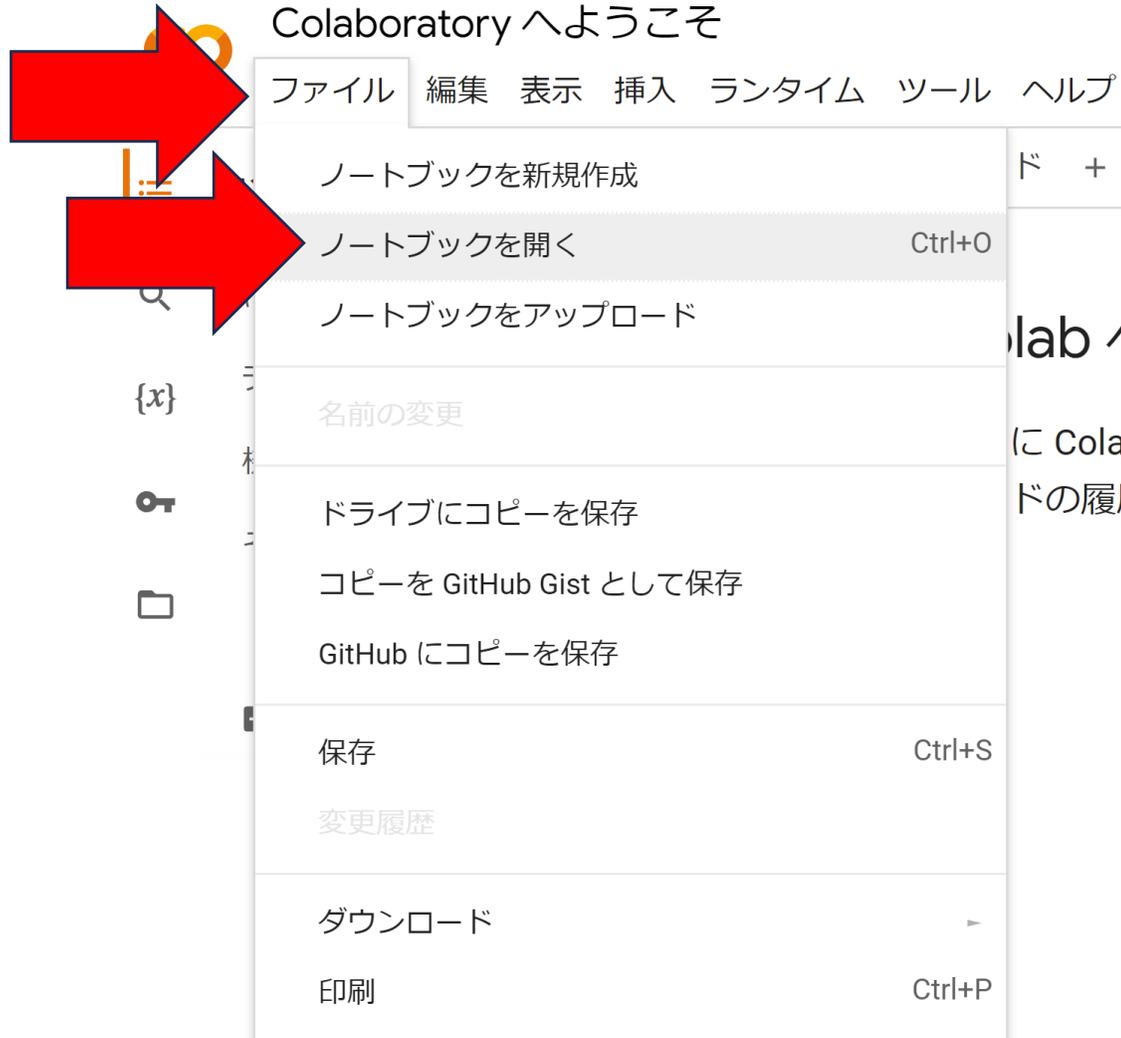
検索: ノートブックを検索

タイトル	最終閲覧 ▲	最初に開いた日 時 ▼	
 演習6コード	9:43	10月25日	 
 Colaboratory へようこそ	9:21	2022年12月23日	
 演習7コード	9:21	9:21	 
 演習4コード	11月2日	11月1日	 
 演習5コード	11月1日	10月13日	 

+ ノートブックを新規作成

キャンセル

検索google colab Colaboratory へようこそ - Colaboratory - Google



Colab へようこそ

に Colab をよくご存じの場合は、この動画でインタラクティブなラドの履歴表示、コマンドパレットについてご覧ください。



Colab とは

検索google colab Colaboratory へようこそ - Colaboratory - Google

ノートブックを開く

例 >

最近 >

Google ドライブ >

GitHub >

アップロード >

タイトル	所有者	最終閲覧 ▲	最終更新 ▼		
演習7コード.ipynb					
演習準備資料.ipynb	曹日丹	11月1日	11月1日		
演習1116確認.ipynb のコピー	曹日丹	10月31日	10月27日		
2023入門dataframe.ipynb	曹日丹	10月31日	10月27日		
演習1116確認.ipynb	曹日丹	10月25日	10月25日		

+ ノートブックを新規作成

キャンセル

検索google colab Colaboratory へようこそ - Colaboratory - Google

ノートブックを開く

例 >

最近 >

Google ドラ
イブ >

GitHub >

アップロード >



参照

または、ここにファイルをドラッグしてください

演習7コード.ipynb

Colab

演習授業中の質問対応について

Zoom ミーティング

表示

ミーティング チャット

演習授業中の質問をチューターの先生方が対応させていただきます。

演習にエラーが出たなど問題があったらリアクションの**挙手**を押してください。

質問内容を入力して、「**全員**」宛てに送信してください。

Miho Ishimaru

ここにメッセージは誰に表示されますか？

宛先: **全員** v

ここにメッセージを入力します...

ミュート解除

ビデオの開始

セキュリティ

参加者 2

画面共有

リアクション

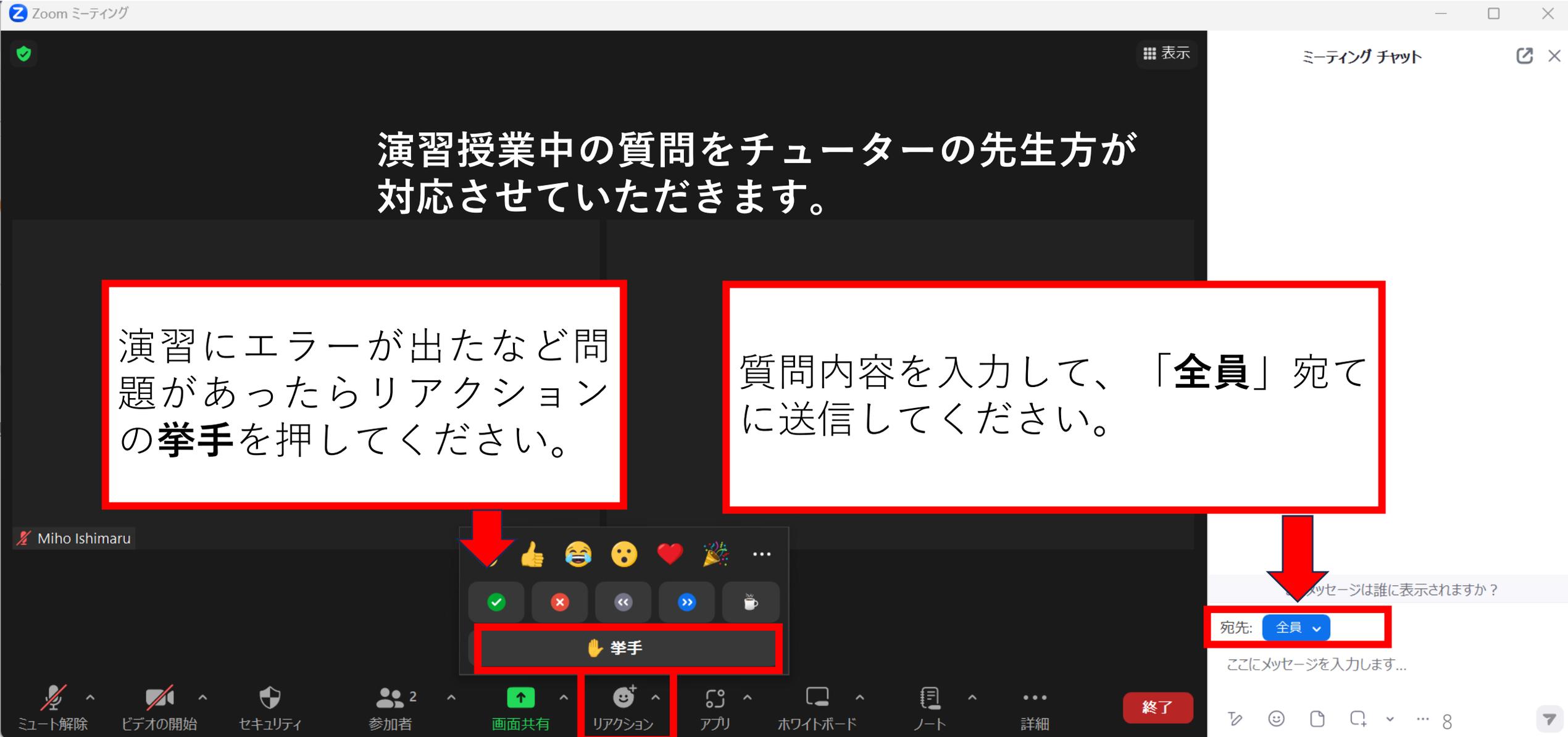
アプリ

ホワイトボード

ノート

詳細

終了



Pythonを使用してデータクレンジングを行う

■データの読み込み:
CSVファイルをGoogleドライブにアップロード、GoogleColabで読み込みました。

■ライブラリのインポート:
Pandasライブラリをインポートしました。

■データの集計:
患者の人数を集計します。1657人

■データの集計:
歯の本数を集計します。

 患者初診時の年齢

演習7 データクレンジングとデータの可視化

患者年齢を求めましょう

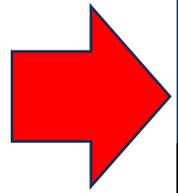
Klistデータフレーム

id列 birthmonth列 yearmonth列

pt_1	1943-01	2020-10
	1943-01	2020-10
	1943-01	2020-10
	1943-01	2020-10
生年月		初診月
pt_2	1982-12	2022-12
	1982-12	2022-12

初診時の年齢

●歳



演習7 データクレンジングとデータの可視化

患者年齢を求めましょう

Klistデータフレーム

	id列	birthmonth列	yearmonth列
pt_1	pt_1	1943-01	2020-10
		生年月	初診月
pt_2	pt_2	1982-12	2022-12
	pt_2	1982-12	2022-12

誕生日の年月と初診日の年月を比較します。

条件式で年齢を計算します

```
if 初診月 >= 誕生日:  
    age = 初診日の年 - 生まれた年  
else:  
    age = 初診日の年 - 生まれた年 - 1
```

初診時の年齢

変数: age

●歳
●歳
●歳
●歳
●歳

演習7 データクレンジングとデータの可視化

患者年齢を求めましょう

Klistデータフレーム

	id列	birthmonth列	yearmonth列
	pt_1	1943-01	2020-10
		生年月	初診月
	pt_2	1982-12	2022-12
	pt_2	1982-12	2022-12

誕生日の年月と初診の年月を比較します。

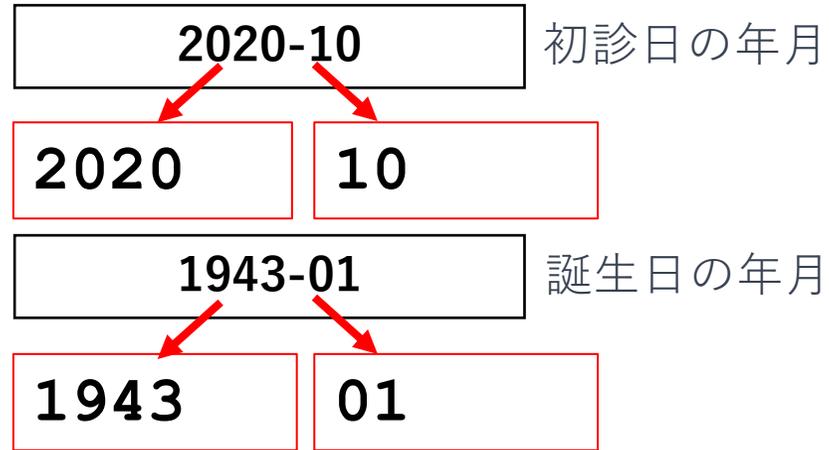
条件式で年齢を計算します

if 初診月 >= 誕生日:

age = 初診の年 - 生まれた年

else:

age = 初診の年 - 生まれた年 - 1



初診時の年齢	age
	●歳
	●歳
	●歳
	●歳
	●歳
	●歳

演習7 データクレンジングとデータの可視化

患者年齢を求めましょう

Klistデータフレーム

id列	birthmonth列	yearmonth列
pt_1	1943-01	2020-10
	生年月	初診月
pt_2	1982-12	2022-12
pt_2	1982-12	2022-12

年と月を分解します

split() メソッド
文字列を特定の区切り文字で分割してリストにすることができます。

文字列: 1943-01
区切り文字: "-"

コード書き方:

文字列.`split("-")`

↓
["1943", "01"]

↓
文字列.`split("-")[0]` → ["1943"]
文字列.`split("-")[1]` → ["01"]

演習7 データクレンジングとデータの可視化

患者年齢を求めましょう

Klistデータフレーム

id列

birthmonth列

yearmonth列

年と月を分解します



pt_1	1943-01	2020-10
	生年月	初診月
pt_2	1982-12	2022-12
pt_2	1982-12	2022-12

インデックス番号0

文字列:

生年月: `Klist["birthmonth"][0]`

文字列:

初診月: `Klist["yearmonth"][0]`

演習7 データクレンジングとデータの可視化

患者年齢を求めましょう

Klistデータフレーム

id列 birthmonth列 yearmonth列

年と月を分解します

pt_1	1943-01	2020-10
	生年月	初診月
pt_2	1982-12	2022-12
pt_2	1982-12	2022-12

インデックス番号0

```
生年月 : Klist["birthmonth"][0]  
Klist["birthmonth"][0].split("-")  
1943-01  →  ["1943", "01"]
```

```
初診月 : Klist["yearmonth"][0]  
Klist["yearmonth"][0].split("-")  
2020-10  →  ["2020", "10"]
```

演習7 データクレンジングとデータの可視化

患者年齢を求めましょう

Klistデータフレーム

id列	birthmonth列	yearmonth列
pt_1	1943-01	2020-10
	生年月	初診月
pt_2	1982-12	2022-12
pt_2	1982-12	2022-12

年と月を分解します

インデックス番号0

`["1943", "01"]`

`Klist["birthmonth"][0].split("-")[0]`

1943

`Klist["birthmonth"][0].split("-")[1]`

01

`["2020", "10"]`

`Klist["yearmonth"][0].split("-")[0]`

2020

`Klist["yearmonth"][0].split("-")[1]`

10

演習7 データクレンジングとデータの可視化

患者年齢を求めましょう

Klistデータフレーム

	id列	birthmonth列	yearmonth列
	pt_1	1943-01	2020-10
		生年月	初診月
	pt_2	1982-12	2022-12
	pt_2	1982-12	2022-12

年と月を分解します

コード

```
print(Klist["birthmonth"][0].split("-")[0])  
1943  
print(Klist["birthmonth"][0].split("-")[1])  
01  
print(Klist["yearmonth"][0].split("-")[0])  
2020  
print(Klist["yearmonth"][0].split("-")[1])  
10
```

演習7 データクレンジングとデータの可視化

患者年齢を求めましょう

Klistデータフレーム

	id列	birthmonth列	yearmonth列
	pt_1	1943-01	2020-10
		生年月	初診月
	pt_2	1982-12	2022-12
	pt_2	1982-12	2022-12

年と月を分解します

コード

```
print(Klist["birthmonth"][0].split("-")[0])  
1943  
print(Klist["birthmonth"][0].split("-")[1])  
01  
print(Klist["yearmonth"][0].split("-")[0])  
2020  
print(Klist["yearmonth"][0].split("-")[1])  
10
```

文字列リストの要素であるため、型は文字です。比較演算子で比較するため、型を文字から整数に変換する必要があります。

演習7データクレンジングとデータの可視化

患者年齢を求めましょう

Klistデータフレーム

	id列	birthmonth列	yearmonth列
	pt_1	1943-01	2020-10
		生年月	初診月
	pt_2	1982-12	2022-12
	pt_2	1982-12	2022-12

年と月をの型を整数に変換します

int() 関数は文字列の値を、整数に変換します。

コードの書き方：`int` (文字列)

コード 文字列 → 整数

```
print(int(Klist["birthmonth"][0].split("-")[0]))
1943
print(int(Klist["birthmonth"][0].split("-")[1]))
01
print(int(Klist["yearmonth"][0].split("-")[0]))
2020
print(int(Klist["yearmonth"][0].split("-")[1]))
10
```

演習7データクレンジングとデータの可視化

患者年齢を求めましょう

Klistデータフレーム

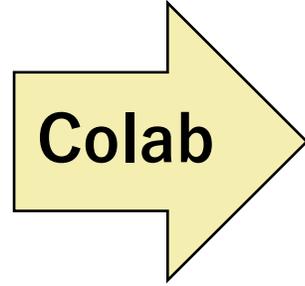
id列	birthmonth列	yearmonth列
pt_1	1943-01	2020-10
	生年月	初診月
pt_2	1982-12	2022-12
pt_2	1982-12	2022-12

インデックス番号53023

年と月をの型を整数に変換します

int() 関数は文字列の値を、整数に変換します。

コードの書き方：**int** (文字列)



```
print(int(Klist["birthmonth"][53023].split("-")[0]))
1982
print(int(Klist["birthmonth"][53023].split("-")[1]))
12
print(int(Klist["yearmonth"][53023].split("-")[0]))
2022
print(int(Klist["yearmonth"][53023].split("-")[1]))
12
```

演習7データクレンジングとデータの可視化

患者年齢を求めましょう

Klistデータフレーム

誕生日の年月と初診の年月を比較します。

初診時の年齢

id列	birthmonth列	yearmonth列
pt_1	1943-01	2020-10
	生年月	初診月
pt_2	1982-12	2022-12
pt_2	1982-12	2022-12

インデックス番号0



インデックス番号0

●歳
●歳
●歳
●歳
●歳
●歳

コード：

```
条件式で年齢を計算します
if 初診月 >= 誕生日:
    age = 初診の年 - 生まれた年
else:
    age = 初診の年 - 生まれた年 - 1
```

```
条件式で年齢を計算します = ●歳
if visitmonth >= birthmonth:
    age = visityear - birthyear
else:
    age = visityear - birthyear - 1
```

演習7データクレンジングとデータの可視化

患者年齢を求めましょう

Klistデータフレーム

id列 birthmonth列 yearmonth列

id列	birthmonth列	yearmonth列
pt_1	1943-01	2020-10
	生年月	初診月
pt_2	1982-12	2022-12
pt_2	1982-12	2022-12

インデックス番号0



インデックス番号0

初診時の年齢

●歳
●歳
●歳
●歳
●歳
●歳

コード:

```
birthyear = int(Klist["birthmonth"][0].split("-")[0])
birthmonth = int(Klist["birthmonth"][0].split("-")[1])
visityear = int(Klist["yearmonth"][0].split("-")[0])
visitmonth = int(Klist["yearmonth"][0].split("-")[1])
```

```
if visitmonth >= birthmonth:
    age = visityear - birthyear
else:
    age = visityear = birthyear - 1
```

print(age) = ●歳

演習7データクレンジングとデータの可視化

患者年齢を求めましょう

Klistデータフレーム

	id列	birthmonth列	yearmonth列
	pt_1	1943-01	2020-10
		生年月	初診月
	pt_2	1982-12	2022-12
	pt_2	1982-12	2022-12

インデックス番号0

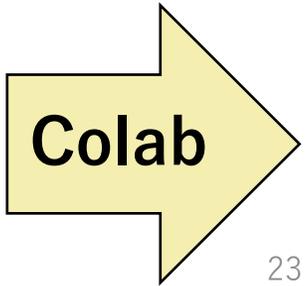
コード:

```
条件式で年齢を計算します  
if 10 >= 01:  
    age = 2020 - 1943  
else:  
    age = 初診の年 - 生まれた年 - 1
```

= 77歳

初診時の年齢

77歳
●歳
●歳
●歳
●歳
●歳



演習7データクレンジングとデータの可視化

患者年齢を求めましょう

Klistデータフレーム

id列 birthmonth列 yearmonth列

pt_1	1943-01	2020-10
	生年月	初診月
pt_2	1982-12	2022-12
pt_2	1982-12	2022-12

インデックス番号0



コード：

```
条件式で年齢を計算します  
if 初診月 >= 誕生月:  
    age = 初診の年 - 生まれた年  
else:  
    age = 初診の年 - 生まれた年 - 1
```

初診時の年齢：age



77歳
●歳
●歳
●歳
●歳
●歳

演習7 データクレンジングとデータの可視化

患者年齢を求めましょう

コード書き方：

新しいageという列を追加

Klistデータフレーム

id列

birthmonth列

yearmonth列

初診時の年齢：age

id列	birthmonth列	yearmonth列
pt_1	1943-01	2020-10
	生年月	初診月
pt_2	1982-12	2022-12
pt_2	1982-12	2022-12

インデックス番号0
インデックス番号1
インデックス番号2
インデックス番号3

for文：繰り返す作業

if文：年齢計算作業

ageという列に値を代入

初診時の年齢：age
77歳
●歳
●歳
●歳
●歳
●歳

演習7 データクレンジングとデータの可視化

患者年齢を求めましょう

Klistデータフレーム

id列	birthmonth	yearmonth	age
pt_1	1943-01	2020-10	None
pt_1	1943-01	2020-10	None
pt_1	1943-01	2020-10	None
pt_1	1943-01	2020-10	None
	生年月	初診月	
pt_2	1982-12	2022-12	None
pt_2	1982-12	2022-12	None

コード:

新しいageという列を追加

```
Klist["age"] = None  
print(Klist)
```

演習7 データクレンジングとデータの可視化

患者年齢を求めましょう

Klistデータフレーム

id列	birthmonth	yearmonth	age
pt_1	1943-01	2020-10	None
pt_1	1943-01	2020-10	None
pt_1	1943-01	2020-10	None
pt_1	1943-01	2020-10	None
	生年月	初診月	
pt_2	1982-12	2022-12	None
pt_2	1982-12	2022-12	None

コード：

新しいageという列を追加

```
Klist["age"] = None  
print(Klist)
```

for文：繰り返す作業

```
for i in range(len(Klist)):
```

演習7 データクレンジングとデータの可視化

患者年齢を求めましょう

Klistデータフレーム

id列	birthmonth	yearmonth	age
pt_1	1943-01	2020-10	None
pt_1	1943-01	2020-10	None
pt_1	1943-01	2020-10	None
pt_1	1943-01	2020-10	None
	生年月	初診月	
pt_2	1982-12	2022-12	None
pt_2	1982-12	2022-12	None

コード：

新しいageという列を追加

```
Klist["age"] = None  
print(Klist)
```

for文：繰り返す作業

```
for i in range(len(Klist)):  
    birthmonth = int(Klist['birthmonth'][i].split('-')[1])  
    birthyear = int(Klist['birthmonth'][i].split('-')[0])  
    visitmonth = int(Klist["yearmonth"][i].split('-')[1])  
    visityear = int(Klist["yearmonth"][i].split('-')[0])  
  
    if visitmonth >= birthmonth:  
        age = visityear - birthyear  
    else:  
        age = visityear - birthyear-1
```

演習7 データクレンジングとデータの可視化

患者年齢を求めましょう

Klistデータフレーム

id列	birthmonth	yearmonth	age
pt_1	1943-01	2020-10	●歳
pt_1	1943-01	2020-10	●歳
pt_1	1943-01	2020-10	●歳
pt_1	1943-01	2020-10	●歳
	生年月	初診月	
pt_2	1982-12	2022-12	●歳
pt_2	1982-12	2022-12	●歳

コード：

新しいageという列を追加

```
Klist["age"] = None  
print(Klist)
```

for文：繰り返す作業

```
for i in range(len(Klist)):  
    birthmonth = int(Klist['birthmonth'][i].split('-')[1])  
    birthyear = int(Klist['birthmonth'][i].split('-')[0])  
    visitmonth = int(Klist["yearmonth"][i].split('-')[1])  
    visityear = int(Klist["yearmonth"][i].split('-')[0])  
  
    if visitmonth >= birthmonth:  
        age = visityear - birthyear  
    else:  
        age = visityear - birthyear-1
```

ageという列に値を代入

```
Klist["age"][i]=age
```

演習7 データクレンジングとデータの可視化

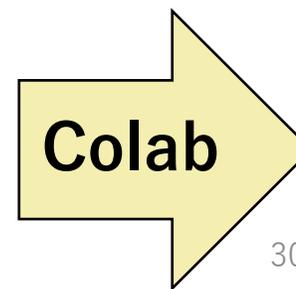
患者年齢を求めましょう

```
print(Klist)
```

	tooth name	tooth record	gender	id	birthmonth	yearmonth	¥
0	A1	残存歯	男	pt_1	1943-01	2020-10	
1	A2	喪失歯	男	pt_1	1943-01	2020-10	
2	A3	残存歯	男	pt_1	1943-01	2020-10	
3	A4	残存歯	男	pt_1	1943-01	2020-10	
4	A5	残存歯	男	pt_1	1943-01	2020-10	
...
53019	D4	喪失歯	女	pt_1657	1982-12	2022-12	
53020	D5	残存歯	女	pt_1657	1982-12	2022-12	
53021	D6	残存歯	女	pt_1657	1982-12	2022-12	
53022	D7	喪失歯	女	pt_1657	1982-12	2022-12	
53023	D8	残存歯	女	pt_1657	1982-12	2022-12	

	tooth exist	age
0	1	77
1	0	77
2	1	77
3	1	77
4	1	77
...
53019	0	40
53020	1	40
53021	1	40
53022	0	40
53023	1	40

[53024 rows x 8 columns]



演習7 データクレンジングとデータの可視化

散布図を作しましょう

患者年齢と歯の本数の関係

縦軸y：
歯の本数

● 患者pt_m

● 患者pt_n

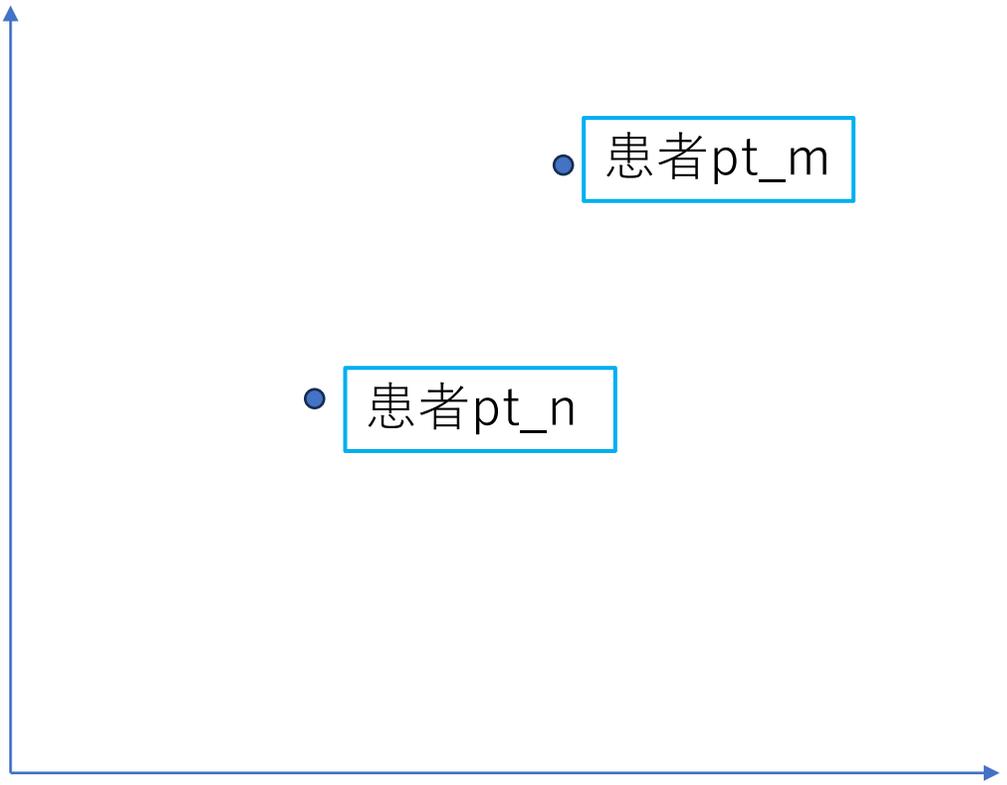
横軸x：患者年齢

演習7 データクレンジングとデータの可視化

散布図を作しましょう

患者年齢と歯の本数の関係

縦軸y：
歯の本数



横軸x：患者年齢

患者pt_n
患者pt_m

横軸x	縦軸y
● 歳	● 本
● 歳	● 本
numpy 配列	numpy 配列

演習7 データクレンジングとデータの可視化

散布図を作りたい

患者年齢と歯の本数の関係

患者pt_n

横軸x

縦軸y

●歳

●本

患者pt_m

●歳

●本

横軸x

コード

```
x = Klist.groupby("id")["age"].mean()  
print(x)
```

縦軸y

コード

```
y = Klist.groupby("id")["tooth exist"].sum()  
print(y)
```

Pandasの**groupbyメソッド**は、
列を指定してデータをグループ化し、それぞれのグループに対して
集計できます。

numpy
配列

numpy
配列

列Aでグループ化し、列Bを集計します。
DataFrame名.groupby("A")["B"].**mean()**

列Aでグループ化し、列Bを集計します。
DataFrame名.groupby("A")["B"].**sum()**

演習7 データクレンジングとデータの可視化

散布図を作しましょう

患者年齢と歯の本数の関係

患者pt_n

患者pt_m

横軸x	縦軸y
●歳	●本
●歳	●本
numpy 配列	numpy 配列

コード

```
import numpy as np  
  
x = np.array(x)  
y = np.array(y)
```

散布図を作りたい

患者年齢と歯の本数の関係

コード

matplotlib ライブラリと日本語入力方法をインポートします。

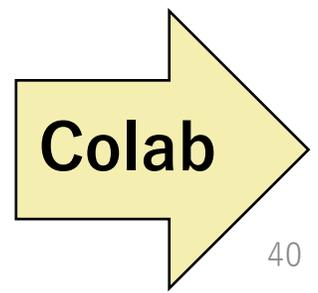
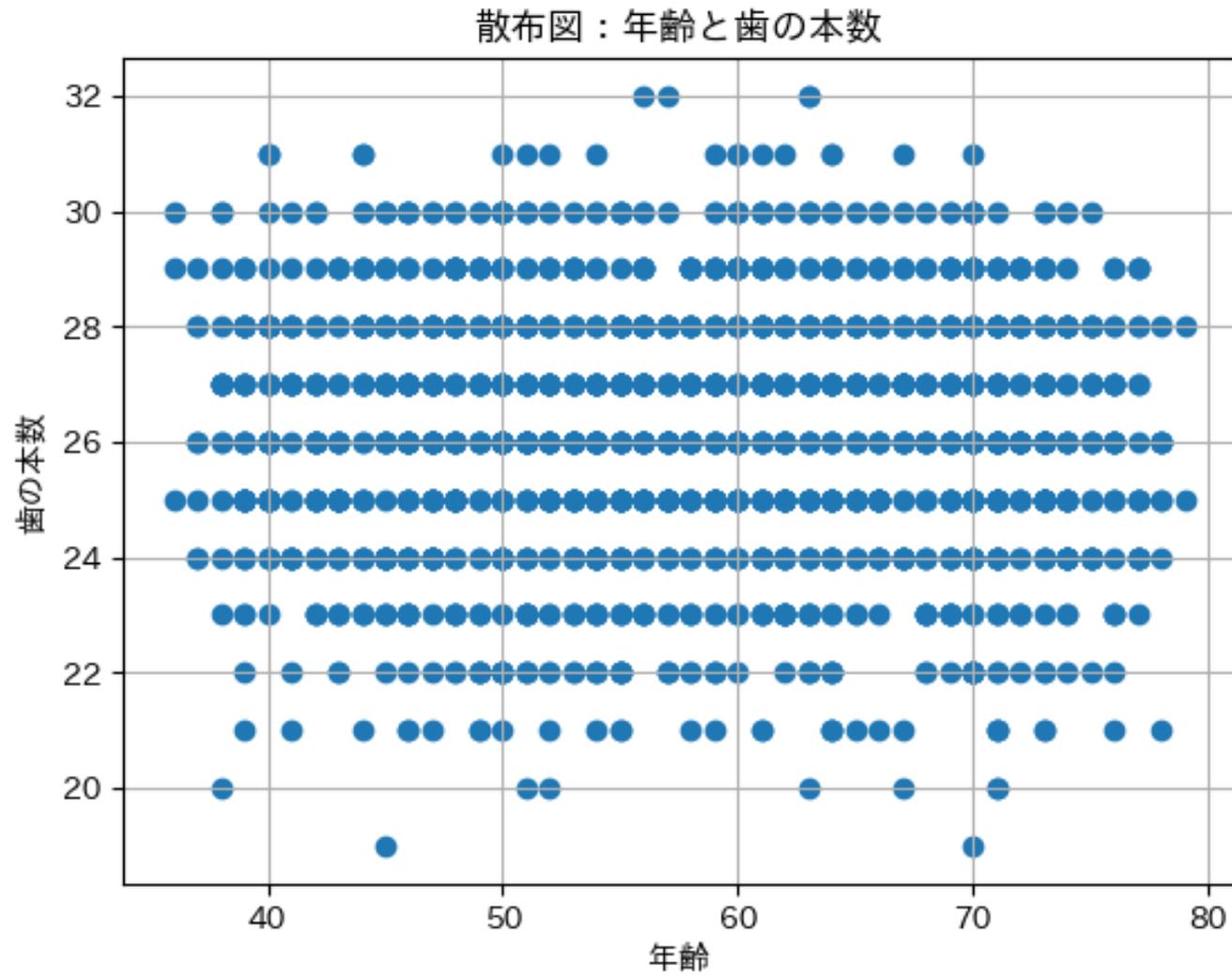
```
import matplotlib.pyplot as plt
!pip install japanize-matplotlib
import japanize_matplotlib
```

matplotlib ライブラリで散布図を作成します。

```
plt.figure()
plt.title('散布図：年齢と歯の本数')
plt.xlabel('年齢')
plt.ylabel('歯の本数')
plt.grid()
plt.scatter(x, y)
plt.show()
```

散布図を作しましょう

患者年齢と歯の本数の関係



WebClassで課題を提出してください、締め切りは**1月11日23:59**までです。

課題1：print関数を使って、コードを1行に書いてください。

Klistデータフレームのid列のインデックス番号1行の文字データ (pt_1) をsplit()メソッドを使って、アルファベットと数字の部分を分解し、また数字の部分を整数に変換してください。

データフレーム名：Klist

列名："id"

インデックス番号：1

文字データ："pt_1"

区切り文字：_

課題2：matplotlibのpyplotモジュールをインポートするコードを書いてください。略称は、pltとします。

演習授業中の質問対応について

The image shows a Zoom meeting window with a dark background. At the top, the Zoom logo and 'Zoom ミーティング' are visible on the left, and window controls on the right. A chat window is open on the right side, showing 'ミーティング チャット'. The main content area has white text: '演習授業中の質問をチューターの先生方が対応させていただきます。'. Below this, two red-bordered boxes contain instructions: '演習にエラーが出たなど問題があったらリアクションの挙手を押してください。' and '質問内容を入力して、「全員」宛てに送信してください.'. At the bottom, the Zoom control bar is shown with icons for Mute, Video, Security, Participants, Screen Share, Reactions, App, Whiteboard, and Notes. A red arrow points to the Reactions icon, and another red arrow points to the '宛先: 全員' dropdown in the chat input area.

Zoom ミーティング

ミーティング チャット

表示

演習授業中の質問をチューターの先生方が対応させていただきます。

演習にエラーが出たなど問題があったらリアクションの挙手を押してください。

質問内容を入力して、「全員」宛てに送信してください。

Miho Ishimaru

1

挙手

宛先: 全員

ここにメッセージを入力します...

終了

2024/1/11 11:35-12:20

医療とAI・ビッグデータ入門

演習8

回帰

統合教育機構
石丸美穂

演習8-20の概要

- 今までの演習1~7ではGoogle ColaboratoryでPythonの基礎的な概念の説明や診療情報データのデータクレンジングを実施した
Numpy, Pandasの基礎的な使い方について習得
- 演習8~20ではGoogle Colaboratoryで機械学習・深層学習を行います
- 今回からは、Pythonで利用できるサンプルデータを利用します

機械学習の分類

教師あり学習



演習8~演習14

教師なし学習



医療とAI・ビッグ
データ応用で取り
扱い予定

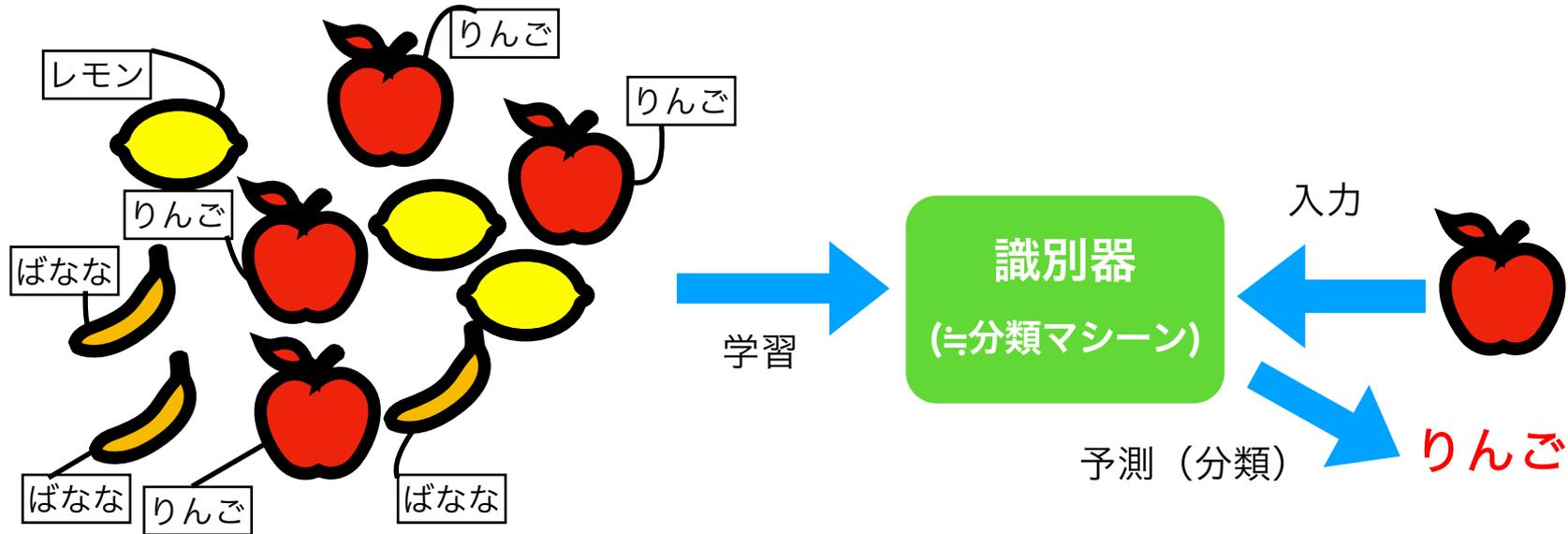
深層学習 (Deep
Learning)



演習15~演習20

教師あり機械学習

教師あり機械学習は特徴のデータ(絵や色など)と「正解」をセットで学習させて識別器を作る



機械学習

医療とAI・ビッグデータ入門の演習8~14では、医療分野でよく使われている「教師あり機械学習」について実践していきます

教師あり機械学習のアルゴリズム（方法）はたくさんある

最初に**回帰分析**のアルゴリズムを実践

回帰分析

回帰は与えられたデータを関数（式）に当てはめる方法

線形回帰式

$$y = b_0 + b_1x_1 + b_2x_2 + \dots + b_nx_n$$

(y : 目的変数、 x_n : 説明変数、 b_0 : 切片、 b_n : 傾き)

説明変数とは、機械学習でいう「特徴量」（特徴のデータ）
目的変数とは、「正解の値」

特徴量が1つ($n=1$)の時、単回帰と呼ぶ

$$y = b_0 + b_1x_1$$

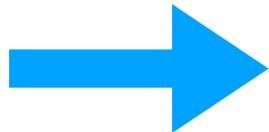
単回帰の基礎

単回帰で演習7で取り扱った
 $y = \text{現在歯数}$ 、 $x = \text{年齢}$ の二つの変数の関連を調べる

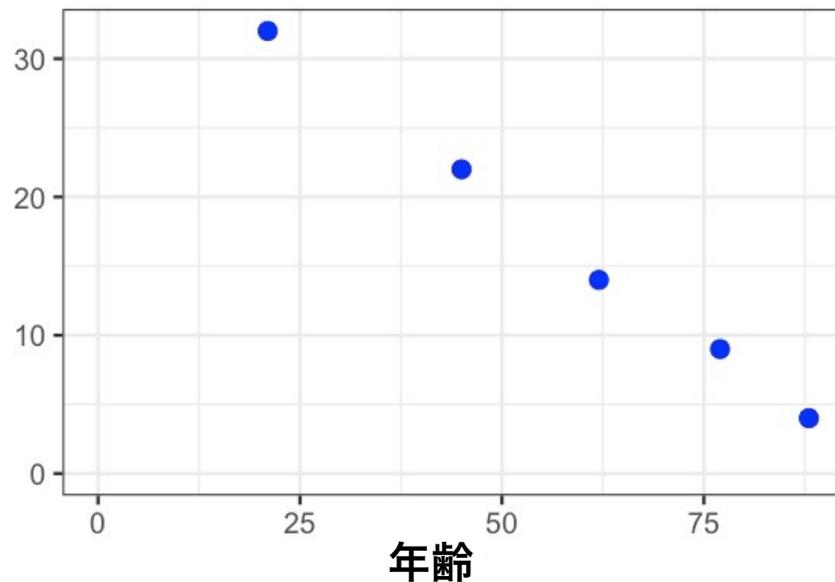
5人の患者の年齢と現在歯数のデータ

被験者	年齢	歯の本数
1	21	32
2	45	22
3	62	14
4	77	9
5	88	4

散布図

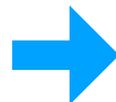
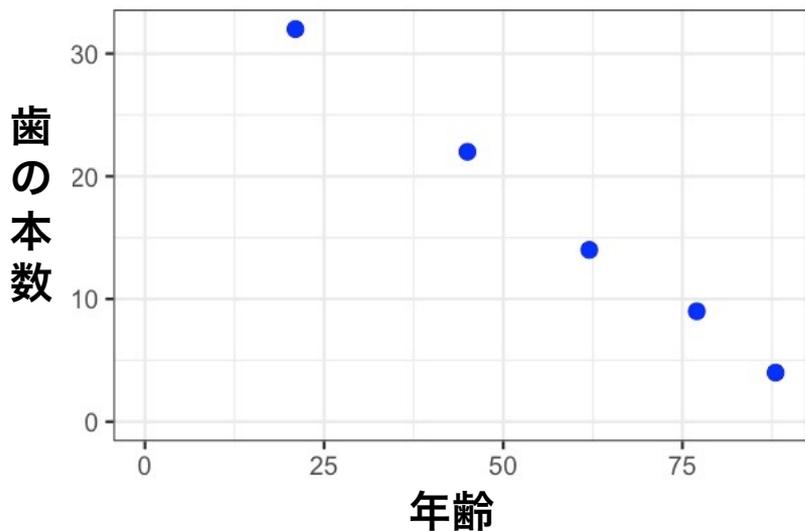


歯の本数

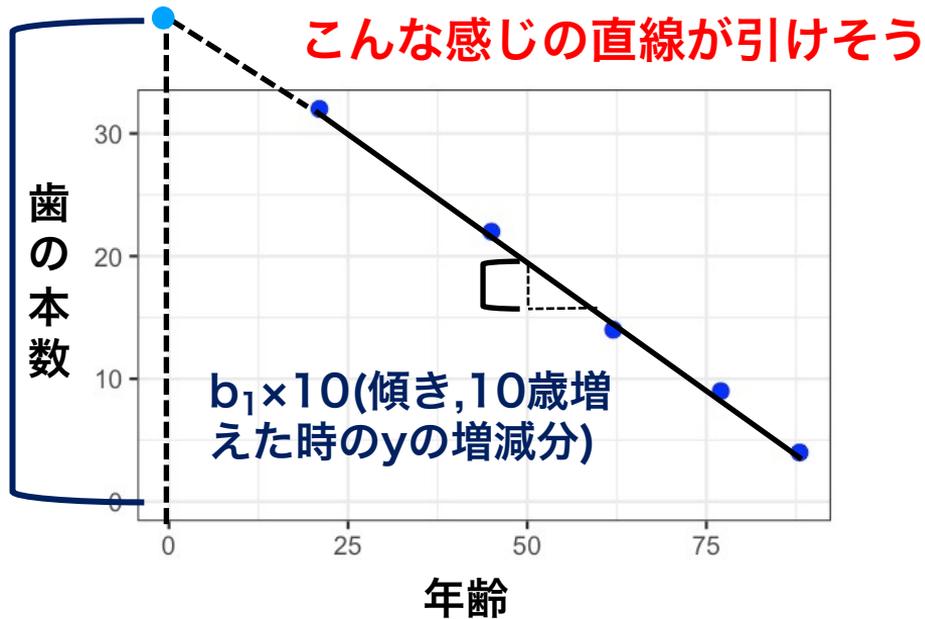


単回帰の基礎

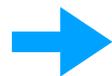
5人の年齢と歯数の散布図を描出した



b_0 (切片)



$$y = b_0 + b_1 X_1$$



b_0 は40ぐらい? b_1 は-0.4ぐらい?

このようにデータを直線の式に当てはめる回帰を線形回帰と呼ぶ

回帰の基礎

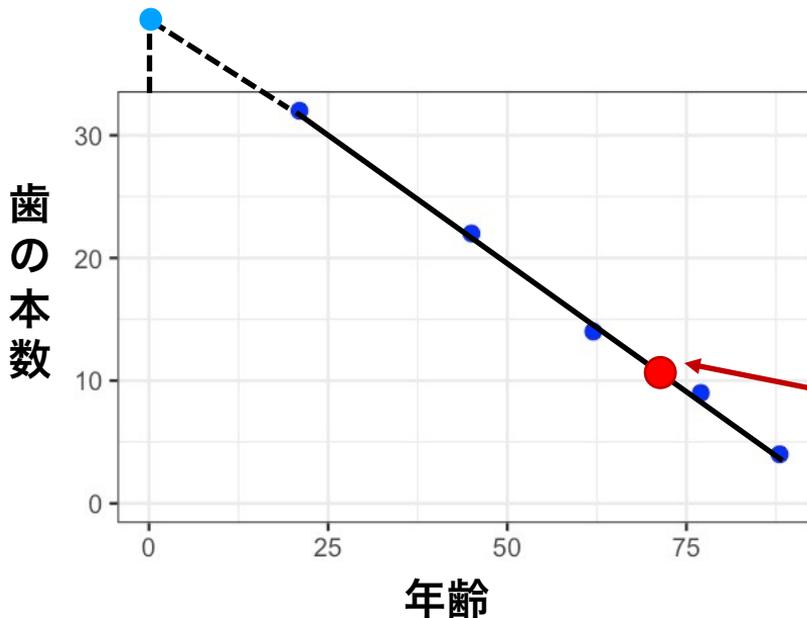
特徴量(x)と正解値(y)の関係を**線形モデル**や**ロジスティックモデル**に近似して予測することを回帰分析と呼ぶ

*モデルとはデータに内在する構造的な関係

医学系では生物統計（医療統計）の文脈で**オッズ比**や**偏回帰係数**を求めることを目的として使うことが多いが、今回は機械学習（yを予測することを目的）で用いる

回帰の基礎

回帰式を使って「予測」をしてみよう



←単回帰モデルの切片と傾きを計算して求める

切片 = 40.6 傾き = -0.42

$$y = 40.6 + (-0.42) \times \text{age}$$

age = 70 (70歳の時)

$$y = 40.6 + (-0.42) \times 70 = \underline{11.5\text{本}}$$

これを求めるのが「予測」である

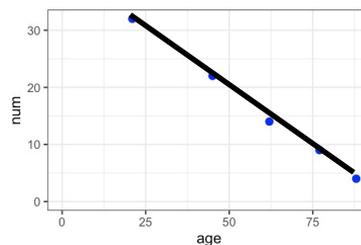
教師あり機械学習の流れ

①データの準備

No.	年齢	歯の本数
1	21	32
2	45	22
3	62	14
4	77	9
5	88	4

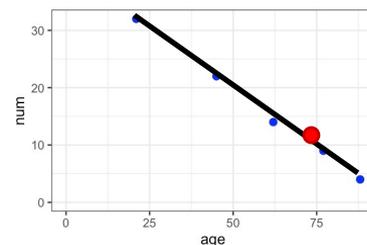
②学習モデルの決定

線形回帰



③学習

④予測・分類



機械学習ではいずれかの**学習モデル**を適用して学習させ、予測や分類を行う

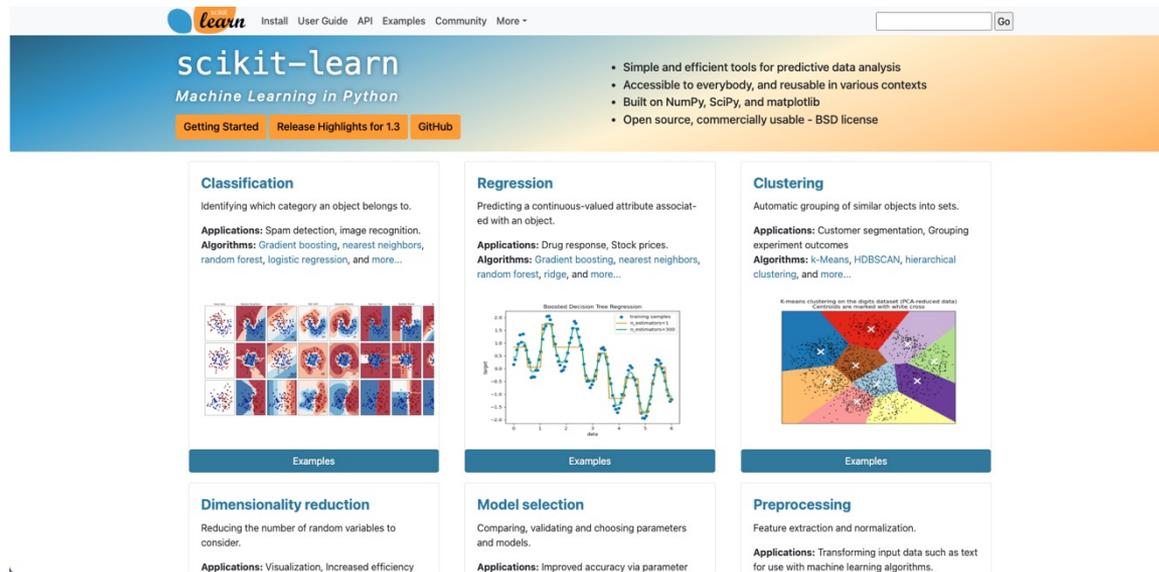
演習

線形回帰

Pythonで回帰を使って予測を行う

scikit-learn (サイキットラーン)

Python で利用できるデータ分析や機械学習のためのライブラリの一つ



The screenshot shows the scikit-learn website homepage. At the top, there is a navigation bar with links for 'Install', 'User Guide', 'API', 'Examples', 'Community', and 'More'. Below this is the main header with the 'scikit-learn' logo and the tagline 'Machine Learning in Python'. A search bar is located on the right side of the header. Below the header, there are several sections, each with a title, a brief description, and a list of applications and algorithms. The sections are: Classification, Regression, Clustering, Dimensionality reduction, Model selection, and Preprocessing. Each section includes a small image or plot illustrating the concept.

- Classification**
Identifying which category an object belongs to.
Applications: Spam detection, image recognition.
Algorithms: Gradient boosting, nearest neighbors, random forest, logistic regression, and more...
- Regression**
Predicting a continuous-valued attribute associated with an object.
Applications: Drug response, Stock prices.
Algorithms: Gradient boosting, nearest neighbors, random forest, ridge, and more...
- Clustering**
Automatic grouping of similar objects into sets.
Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, HDBSCAN, hierarchical clustering, and more...
- Dimensionality reduction**
Reducing the number of random variables to consider.
Applications: Visualization, Increased efficiency
- Model selection**
Comparing, validating and choosing parameters and models.
Applications: Improved accuracy via parameter
- Preprocessing**
Feature extraction and normalization.
Applications: Transforming input data such as text for use with machine learning algorithms.

➡ 今後の演習ではこのライブラリを用いて機械学習を行う

Pythonで回帰を使って予測を行う

`scikit-learn`を使い、予測を行う演習の流れ

STEP0 : 事前準備

STEP1 : データの用意

STEP2 : 学習モデルの選択

STEP3 : データを入れて学習させる

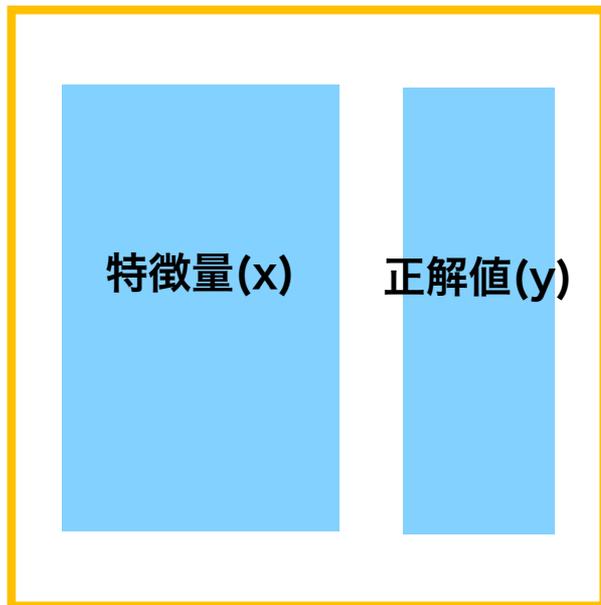
STEP4 : 傾き (偏回帰係数) と切片 (定数項) を求める

STEP5 : 予測を行う

STEP6 : モデルの評価

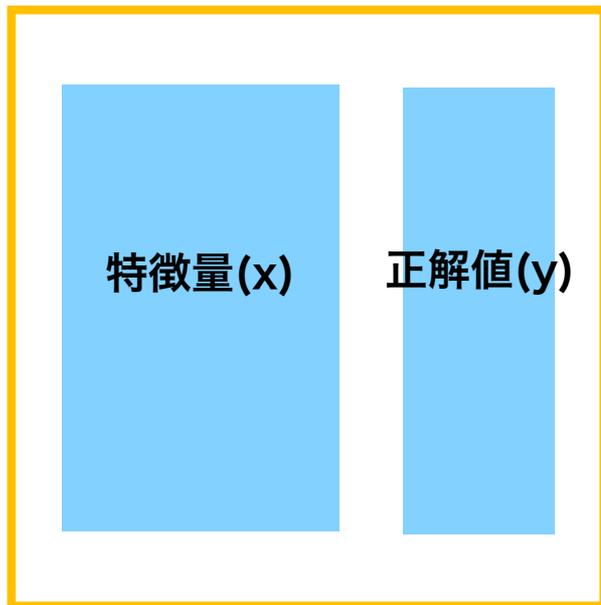
機械学習の流れのまとめ

STEP1:データの用意



機械学習の流れのまとめ

STEP1:データの用意



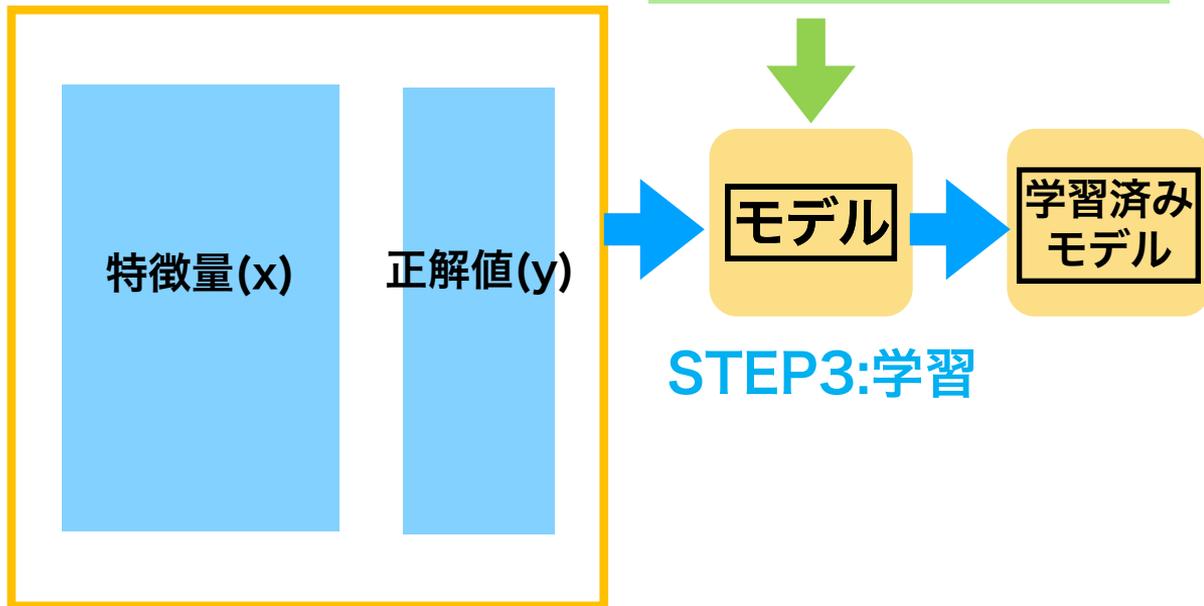
STEP2:学習モデルの選定



機械学習の流れのまとめ

STEP1:データの用意

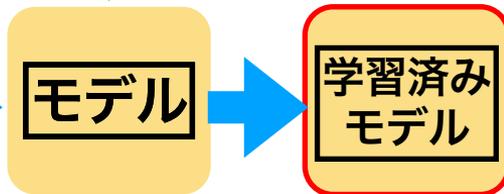
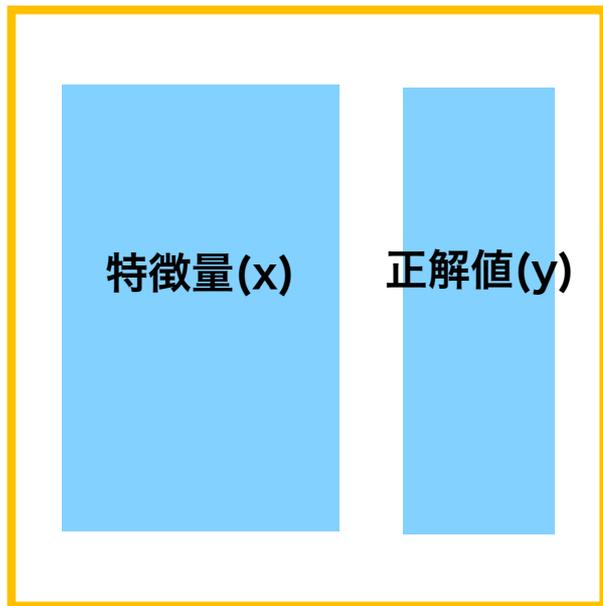
STEP2:学習モデルの選定



機械学習の流れのまとめ

STEP1:データの用意

STEP2:学習モデルの選定



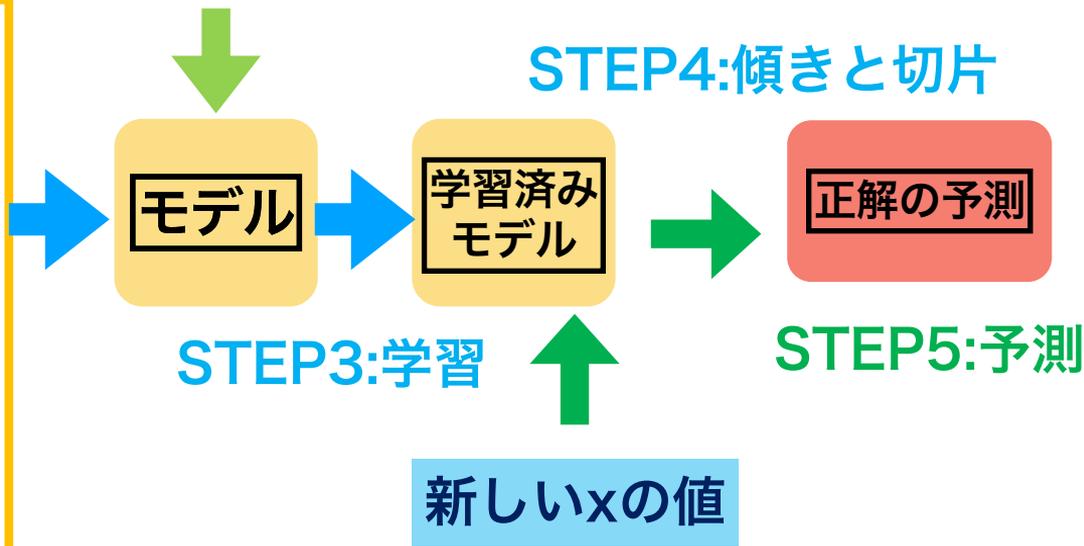
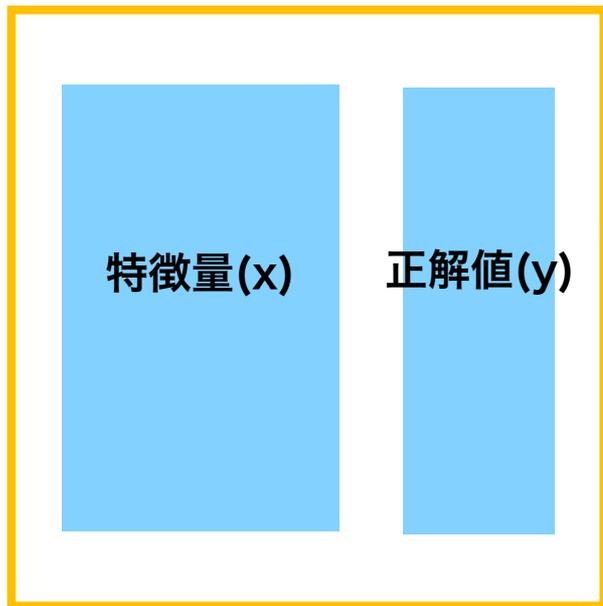
STEP3:学習

STEP4:傾きと切片

機械学習の流れのまとめ

STEP1:データの用意

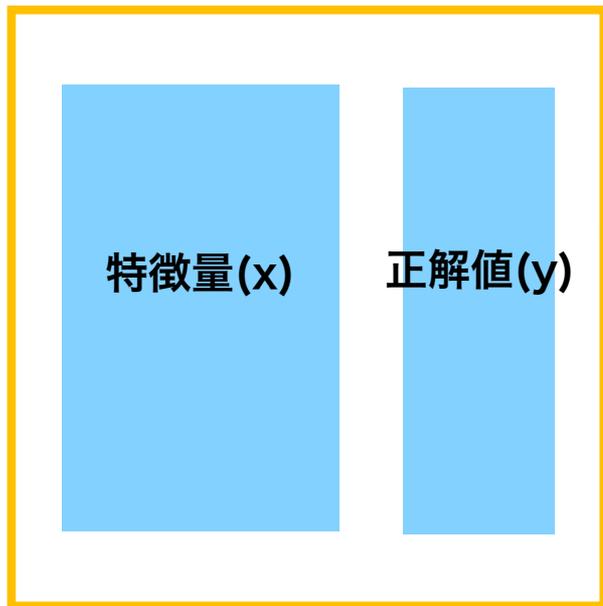
STEP2:学習モデルの選定



機械学習の流れのまとめ

STEP1:データの用意

STEP2:学習モデルの選定



STEP4:傾きと切片



STEP3:学習

STEP5:予測

新しいxの値



STEP6:モデルの評価

機械学習のコードのまとめ

STEP1 データの用意：特徴量のデータと正解値のデータ

STEP2 学習モデルの選択 (今回は線形回帰)

(モデル名) = `LinearRegression()`

STEP3 データを入れて学習させる

(モデル名).`fit`(特徴量, 正解値)

STEP4 傾き(偏回帰係数)と切片(定数項)を求める

(モデル名).`coef_`

(モデル名).`intercept_`

STEP5 予測を行う：(モデル名).`predict`(新たなインスタンス)

STEP6 モデルの評価：(モデル名).`score`(特徴量, 正解値)

機械学習のコードのまとめ

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

STEP0 : ライブラリの読み込み

```
from sklearn.datasets import load_diabetes
dm = load_diabetes(as_frame = True, scaled = False)
y_dm = dm.target
x_dm = dm.data
x_bmi = x_dm[['bmi']]
```

STEP1 : データの準備

```
plt.scatter(x_bmi, y_dm)
```

```
from sklearn.linear_model import LinearRegression
model_bmi = LinearRegression()
```

STEP2 : 学習モデルの選択

```
model_bmi.fit(x_bmi, y_dm)
```

STEP3 : データを入れて学習させる

```
print(model_bmi.coef_)
print(model_bmi.intercept_)
```

STEP4 : 傾きと切片を求める

```
pre = pd.DataFrame([[20]], columns=['bmi'])
print(model_bmi.predict(pre))
```

STEP5 : 予測を行う

```
print(model_bmi.score(x_bmi, y_dm))
```

STEP6 : モデルの評価

線形回帰分析

STEP0 : Google Colaboratoryの立ち上げ

STEP0 : 事前準備

STEP1 : データの用意

STEP2 : 学習モデルの選択

STEP3 : データを入れて学習させる

STEP4 : 傾きと切片を求める

STEP5 : 予測を行う

STEP6 : モデルの評価

検索google colab [Laboratory へようこそ - Colaboratory - Google](#)

The screenshot shows the Google Colaboratory web interface. The 'File' menu is open, displaying options such as 'ノートブックを新規作成', 'ノートブックを開く', 'ノートブックをアップロード', '名前の変更', 'ドライブにコピーを保存', 'コピーを GitHub Gist として保存', 'GitHub にコピーを保存', '保存', '変更履歴', 'ダウンロード', and '印刷'. Two red arrows point to the 'File' menu and the 'ノートブックを開く' option. In the background, a video player is visible with the title '3 Cool Google Colab Features' and a play button. The text 'Colab とは' is partially visible at the bottom.

演習8コード.ipynb

ノートブックを開く

例 >

最近 >

Google ドラ
イブ >

GitHub >

アップロード >

2
5



参照

または、ここにファイルをドラッグしてください

演習8コード.ipynb

線形回帰分析

STEP0 : ライブラリのインポート

STEP0 : 事前準備

STEP1 : データの用意

STEP2 : 学習モデルの選択

STEP3 : データを入れて学習させる

STEP4 : 傾きと切片を求める

STEP5 : 予測を行う

STEP6 : モデルの評価

コード8-1 前回までの演習で使ったnumpy, pandas,
matplotlib.pyplotをインポートする

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

ライブラリをインポート : import (ライブラリ名) as 省略形

numpyとpandas

モジュールをインポート : import (ライブラリ名) . (モジュール名) as 省略形

matplotlib

pyplot

colab

線形回帰分析

STEP1：糖尿病データの読み込み

STEP0：事前準備

STEP1：データの用意

STEP2：学習モデルの選択

STEP3：データを入れて学習させる

STEP4：傾きと切片を求める

STEP5：予測を行う

STEP6：モデルの評価

コード8-2 diabetes(糖尿病) のデータセットを読み込む

```
from sklearn.datasets import load_diabetes
```

sklearnはscikit-learnのライブラリ名

関数だけをインポート `from sklearn.datasets import load_diabetes`

ライブラリ名

モジュール名

関数名

●`load_diabetes()` はsklearnに含まれているデータセットの中から、`diabetes`データを読み込むための関数

線形回帰分析

STEP1：糖尿病データの読み込み

STEP0：事前準備

STEP1：データの用意

STEP2：学習モデルの選択

STEP3：データを入れて学習させる

STEP4：傾きと切片を求める

STEP5：予測を行う

STEP6：モデルの評価

コード8-2 diabetes(糖尿病) のデータセットを読み込む

```
dm = load_diabetes(as_frame = True, scaled = False)
```

● `load_diabetes()` 関数を実行し、機械学習用のサンプルデータである `diabetes` データセットを読み込み `dm` (糖尿病の省略形) の変数に入れる

● `load_diabetes(as_frame = True/False, scaled = True/False)` と二つの引数で、`True` (真) か `False` (偽) を選択する

読み込むデータの形式を指定

`True` → `pd` のデータフレーム型

`False` → `numpy` 配列

読み込むデータのスケールを指定

`True` → 正規化された値 (全て1~0に変換)

`False` → 正規化されていない値 (生データ)

線形回帰分析

STEP1 : 糖尿病データの内容確認

コード8-3 dmの内容を調べる①

```
dm #データ自体を出力
```

```
type(dm) #データの型を出力
```

```
sklearn.utils._bunch.Bunch
```

Bunch型はdictionary(辞書)型的一种

```
dataset = {"key1": "a", "key2": "b", "key3": "c"}  
辞書名      キー1 値1   キー2 値2   キー3 値3  
            要素1   要素2   要素3
```

辞書型は順序づけたキーと値の集まりで、(データ名).(キー) で格納したデータ(値)を呼び出す

STEP0 : 事前準備

STEP1 : データの用意

STEP2 : 学習モデルの選択

STEP3 : データを入れて学習させる

STEP4 : 傾きと切片を求める

STEP5 : 予測を行う

STEP6 : モデルの評価

```
{'data': age sex bmi bp s1 s2 s3 s4 s5 s6  
0 59.0 2.0 32.1 101.00 157.0 93.2 38.0 4.00 4.8598 87.0  
1 48.0 1.0 21.6 87.00 183.0 103.2 70.0 3.00 3.8918 69.0  
2 72.0 2.0 30.5 93.00 156.0 93.6 41.0 4.00 4.6728 85.0  
3 24.0 1.0 25.3 84.00 198.0 131.4 40.0 5.00 4.8903 89.0  
4 50.0 1.0 23.0 101.00 192.0 125.4 52.0 4.00 4.2905 80.0  
... ..  
437 60.0 2.0 28.2 112.00 185.0 113.8 42.0 4.00 4.9836 93.0  
438 47.0 2.0 24.9 75.00 225.0 166.0 42.0 5.00 4.4427 102.0  
439 60.0 2.0 24.9 99.67 162.0 106.6 43.0 3.77 4.1271 95.0  
440 36.0 1.0 30.0 95.00 201.0 125.2 42.0 4.79 5.1299 85.0  
441 36.0 1.0 19.6 71.00 250.0 133.2 97.0 3.00 4.5951 92.0
```

[442 rows x 10 columns]

```
'target': 151.0
```

```
1 75.0  
2 141.0  
3 206.0  
4 135.0  
...  
437 178.0  
438 104.0  
439 132.0  
440 220.0  
441 57.0
```

```
'DESCR': .. _diabetes_dataset:\n\nDiabetes dat  
sex, body mass index, average blood\npressure,  
each of n=\n442 diabetes patients, as well as th  
disease progression one year after baseline.\n\nInstances: 442\n\n :Number of Attributes: First  
:Target: Column 11 is a quantitative measure of c  
:Attribute Information:\n - age age in years
```

線形回帰分析

STEP1：糖尿病データの内容確認

STEP0：事前準備

STEP1：データの用意

STEP2：学習モデルの選択

STEP3：データを入れて学習させる

STEP4：傾きと切片を求める

STEP5：予測を行う

STEP6：モデルの評価

コード8-4 dmの内容を調べる②

```
print(dm.DESCR)
```

辞書型は順序づけられたキーと値の集まりで、(データ名).(キー) で格納したデータ(値)を呼び出す

dmのキーの一つであるDESCRはデータセットの中身の定義について記載されている部分

線形回帰分析

STEP1：糖尿病データの内容確認

```
⇒ .._diabetes_dataset:
```

```
Diabetes dataset
```

```
-----
```

Ten baseline variables, age, sex, body mass index, average blood pressure, and six blood serum measurements were obtained for each of n = 442 diabetes patients, as well as the response of interest, a quantitative measure of disease progression one year after baseline.

Data Set Characteristics:

:Number of Instances: 442

行は442人

:Number of Attributes: First 10 columns are numeric predictive values

:Target: Column 11 is a quantitative measure of disease progression one year after baseline

:Attribute Information:

- age age in years
- sex
- bmi body mass index
- bp average blood pressure
- s1 tc, total serum cholesterol
- s2 ldl, low-density lipoproteins
- s3 hdl, high-density lipoproteins
- s4 tch, total cholesterol / HDL
- s5 ltg, possibly log of serum triglycerides level
- s6 glu, blood sugar level

カラム (列) の定義

s1:血清コレステロール値

s2:LDLコレステロール

s3:HDLコレステロール

s4:総コレステロール/HDL

s5:log(血清トリグリセミド)

s6:血糖値

STEP0：事前準備

STEP1：データの用意

STEP2：学習モデルの選択

STEP3：データを入れて学習させる

STEP4：傾きと切片を求める

STEP5：予測を行う

STEP6：モデルの評価

10個のベースライン変数(年齢、性別、BMI、平均血圧、6個の血清測定値)が442人の糖尿病患者についてそれぞれ取得

興味がある応答変数は、ベースラインの1年後の糖尿病の進行度を「量的」に示したもの

線形回帰分析

STEP1：糖尿病データの内容確認

STEP0：事前準備

STEP1：データの用意

STEP2：学習モデルの選択

STEP3：データを入れて学習させる

STEP4：傾きと切片を求める

STEP5：予測を行う

STEP6：モデルの評価

年齢：59歳
性別：女性
BMI：32
...



target : 151



時間

ベースライン(基準)時点

→10個の特徴量の測定時点

1年後

正解値 (target :
糖尿病の進行度を量的に示し
たもの)の測定時点

ベースライン時点のx(特徴量)で1年後のtargetをy(正解値) と
の関連を調べる、学習モデルの作成を行う

colab

線形回帰分析

STEP1：正解データを作成

コード8-5 正解値データを作成

```
y_dm = dm.target  
y_dm.head()
```

- `dm`の`target`キーで格納されている値を呼び出し、`y_dm`に格納する
- (データフレーム名).`head()`で最初の5行を出力する(見るだけ)

```
y_dm.shape #データの配列構造を出力
```

➡ (442,) 442個の1次元配列(ベクトル)

```
type(y_dm) #データの型を出力
```

➡ `pandas.core.series.Series`

STEP0：事前準備

STEP1：データの用意

STEP2：学習モデルの選択

STEP3：データを入れて学習させる

STEP4：傾きと切片を求める

STEP5：予測を行う

STEP6：モデルの評価

0 151.0

1 75.0

2 141.0

3 206.0

4 135.0

Name: target, dtype: float64

線形回帰分析

STEP1：正解データの記述

STEP0：事前準備

STEP1：データの用意

STEP2：学習モデルの選択

STEP3：データを入れて学習させる

STEP4：傾きと切片を求める

STEP5：予測を行う

STEP6：モデルの評価

コード8-6 y_dmの記述統計値を調べる

```
y_dm.describe()
```



count	442.000000
mean	152.133484
std	77.093005
min	25.000000
25%	87.000000
50%	140.500000
75%	211.500000
max	346.000000

Name: target, dtype: float64

pandasのDataFrameやSeriesには `describe()` メソッドがあり、(データフレーム名).describe() で各列の記述統計値（平均値 (mean) や最小 (min)、最大値 (max) など）を出力できる

進行度の数値と言ってもピンとこないため、どのような数値を取るのか記述統計を出力

平均 (mean) は152.13、最小値は25、最大値は346の値をとる
大きいほど1年後の糖尿病が悪化している

線形回帰分析

STEP1：特徴量データの整形

STEP0：事前準備

STEP1：データの用意

STEP2：学習モデルの選択

STEP3：データを入れて学習させる

STEP4：傾きと切片を求める

STEP5：予測を行う

STEP6：モデルの評価

コード8-7 x(特徴量データ) のデータを作成

```
x_dm = dm.data  
x_dm.head() #最初の5行を出力
```

dmのdataキーで格納された値を呼び出し、x_dmに格納する

```
x_dm.shape #データの配列構造を出力
```

➡ (442, 10) 442行×10列の2次元配列(行列)

```
type(x_dm) #データの型を出力
```

➡ pandas.core.frame.DataFrame



	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6
0	59.0	2.0	32.1	101.0	157.0	93.2	38.0	4.0	4.8598	87.0
1	48.0	1.0	21.6	87.0	183.0	103.2	70.0	3.0	3.8918	69.0
2	72.0	2.0	30.5	93.0	156.0	93.6	41.0	4.0	4.6728	85.0
3	24.0	1.0	25.3	84.0	198.0	131.4	40.0	5.0	4.8903	89.0
4	50.0	1.0	23.0	101.0	192.0	125.4	52.0	4.0	4.2905	80.0

age~s6までの10変数

線形回帰分析

STEP1 : x(特徴量データ) の記述

STEP0 : 事前準備

STEP1 : データの用意

STEP2 : 学習モデルの選択

STEP3 : データを入れて学習させる

STEP4 : 傾きと切片を求める

STEP5 : 予測を行う

STEP6 : モデルの評価

コード8-8 x_dmの記述統計値を調べる

```
x_dm.describe()
```

	年齢	性別	BMI	平均血圧	TC (総コレステ ロール値)	LDL (悪玉コレス テロール)	HDL (善玉コレス テロール)	TCH (TC/HDL)	LTG (血液中の 中性脂肪値 の対数)	GLU (血糖値)
	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6
count	442.000000	442.000000	442.000000	442.000000	442.000000	442.000000	442.000000	442.000000	442.000000	442.000000
mean	48.518100	1.468326	26.375792	94.647014	189.140271	115.439140	49.788462	4.070249	4.641411	91.260181
std	13.109028	0.499561	4.418122	13.831283	34.608052	30.413081	12.934202	1.290450	0.522391	11.496335
min	19.000000	1.000000	18.000000	62.000000	97.000000	41.600000	22.000000	2.000000	3.258100	58.000000
25%	38.250000	1.000000	23.200000	84.000000	164.250000	96.050000	40.250000	3.000000	4.276700	83.250000
50%	50.000000	1.000000	25.700000	93.000000	186.000000	113.000000	48.000000	4.000000	4.620050	91.000000
75%	59.000000	2.000000	29.275000	105.000000	209.750000	134.500000	57.750000	5.000000	4.997200	98.000000
max	79.000000	2.000000	42.200000	133.000000	301.000000	242.400000	99.000000	9.090000	6.107000	124.000000

線形回帰分析

STEP1 : x(特徴量データ) の整形

コード8-9 x_dmからbmi列だけを抽出

```
x_bmi = x_dm[['bmi']]  
x_bmi
```

特徴量を一つだけにして単回帰を行いたいので、x_dmのbmiを回帰に入れる変数に選択し、bmiだけのデータを作成する

二重カッコ [[]] で列名を指定する

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6
count	442.000000	442.000000	442.000000	442.000000	442.000000	442.000000	442.000000	442.000000	442.000000	442.000000
mean	48.518100	1.468326	26.375792	146.647014	189.140271	115.439140	49.788462	4.070249	4.641411	91.260181
std	13.109028	0.499561	4.418122	3.831283	34.608052	30.413081	12.934202	1.290450	0.522391	11.496335
min	19.000000	1.000000	18.000000	120.000000	97.000000	41.600000	22.000000	2.000000	3.258100	58.000000
25%	38.250000	1.000000	23.200000	140.000000	164.250000	96.050000	40.250000	3.000000	4.276700	83.250000
50%	50.000000	1.000000	25.700000	173.000000	186.000000	113.000000	48.000000	4.000000	4.620050	91.000000
75%	59.000000	2.000000	29.275000	195.000000	209.750000	134.500000	57.750000	5.000000	4.997200	98.000000
max	79.000000	2.000000	42.200000	193.000000	301.000000	242.400000	99.000000	9.090000	6.107000	124.000000

STEP0 : 事前準備

STEP1 : データの用意

STEP2 : 学習モデルの選択

STEP3 : データを入れて学習させる

STEP4 : 傾きと切片を求める

STEP5 : 予測を行う

STEP6 : モデルの評価



	bmi
0	32.1
1	21.6
2	30.5
3	25.3
4	23.0
...	...
437	28.2
438	24.9
439	24.9
440	30.0
441	19.6

442 rows x 1 columns

データフレーム型から列を取り出す方法

二重の角括弧[[]]

```
x_bmi = x_dm[['bmi']]
```

一重の角括弧[]

```
x_bmi1 = x_dm['bmi']
```

コード8-10

```
x_bmi.shape #データの配列構造を出力
```

➡ (442, 1) 442行×1列の2次元配列(行列)

```
type(x_bmi) #データの型を出力
```

➡ pandas.core.frame.DataFrame

```
x_bmi1.shape #データの配列構造を出力
```

➡ (442,) 442個の1次元配列(ベクトル)

```
type(x_bmi1) #データの型を出力
```

➡ pandas.core.series.Series

回帰で学習を行うには、 x と y の配列がPythonでは指定されている

`x_bmi`

`y_dm`

x は2次元配列 (行列の形)

y は1次元配列 (ベクトルの形)

151.0, 75.0, 141.0...

	BMI
0	32.1
1	21.6
2	30.5

442 rows x 1 columns

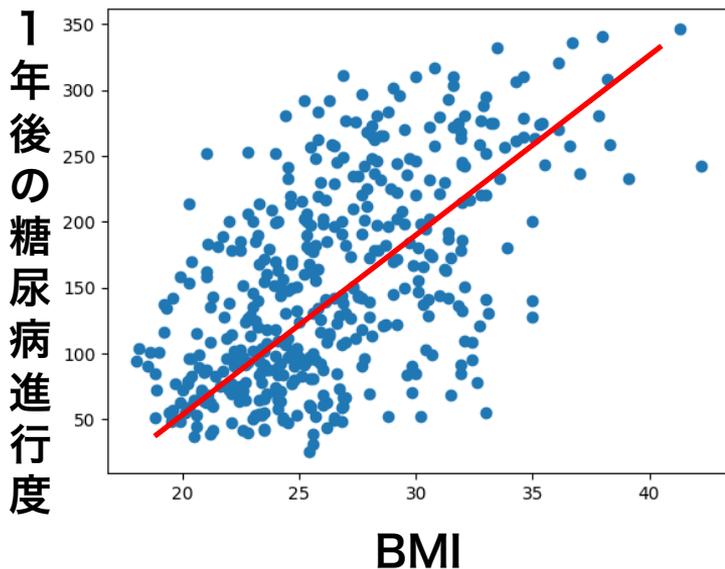
配列を間違えるとエラーになりコードが動かないので、作成したデータの配列を確認するように気をつける

線形回帰分析

STEP1：用意したデータのプロット

コード8-11 正解値と特徴量で散布図を描出

```
plt.scatter(x_bmi, y_dm)
```



散布図は`plt.scatter(x,y)`で描出 (演習7の復習)

この後の回帰では、
回帰式： $y=b_0+b_1x_1$
の b_0 と b_1 を決定する

決定する方法は、最小二乗法(Ordinary Least Square, OLS)を用いる

b_0 と b_1 が決まると赤い線が引ける

STEP0：事前準備

STEP1：データの用意

STEP2：学習モデルの選択

STEP3：データを入れて学習させる

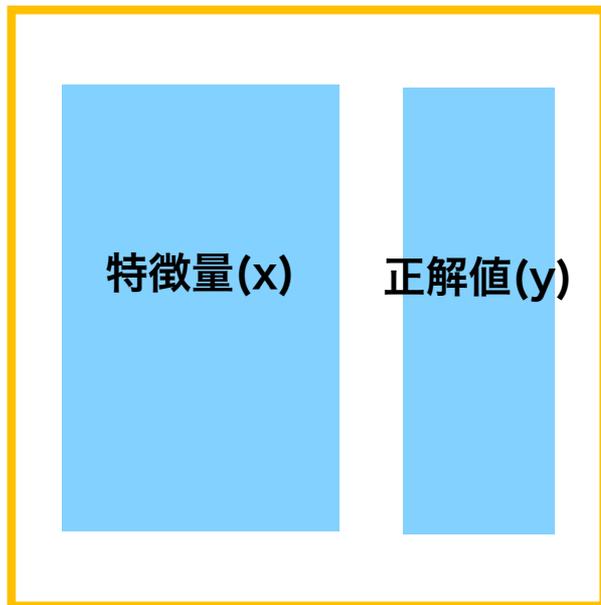
STEP4：傾きと切片を求める

STEP5：予測を行う

STEP6：モデルの評価

機械学習の流れのまとめ

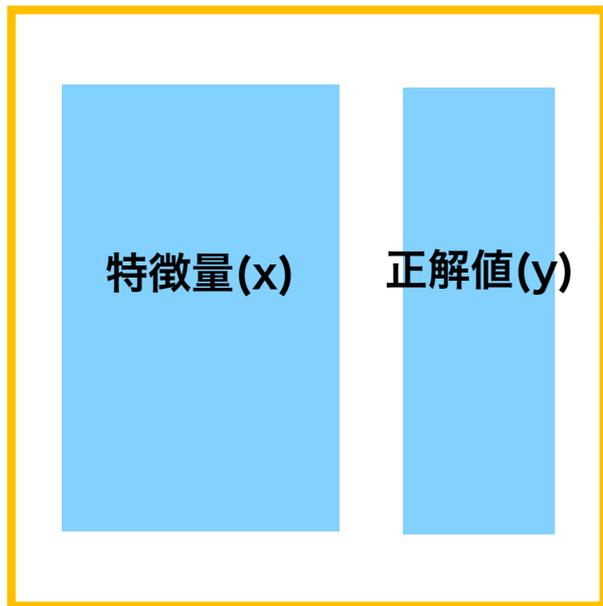
STEP1:データの用意



機械学習の流れのまとめ

STEP1:データの用意

STEP2:学習モデルの選定



STEP4:傾きと切片



STEP3:学習

STEP5:予測

新しいxの値



STEP6:モデルの評価

演習8：課題

Webclassで課題を提出してください。締め切りは**2024/01/25 23:59**まで

(1)今回使用した糖尿病のデータの特徴量データからageだけのx_ageを作成するコードを記載してください

(2)x_ageの先頭5行を出力するコードを記載してください

(3)x_ageの配列構造を記載してください(コードではなく出力結果)

(4)x_ageの型を記載してください(コードではなく出力結果)

来週から水曜日・木曜日で授業あるので注意

授業準備：Webclassからコードをダウンロードし、 Google colaboratoryで開いておいてください

演習授業中の質問対応について

The screenshot shows a Zoom meeting window with a black background and white text. The main text reads: "演習授業中の質問をチューターの先生が対応させていただきます。" (We will respond to questions during the exercise class by the tutor teacher.)

Annotations include:

- A red box on the left contains the text: "演習にエラーが出たなど問題があったらリアクションの挙手を押してください。" (If there is an error during the exercise or a problem, please raise your hand with a reaction.) An arrow points from this box to the reaction bar in the Zoom interface.
- A red box on the right contains the text: "質問内容を入力して、「全員」宛てに送信してください。" (Enter the question content and send it to "Everyone"). An arrow points from this box to the chat input field in the Zoom interface.
- The Zoom interface shows a reaction bar with icons for clapping, thumbs up, thumbs down, sad face, and heart. The "挙手" (Raise Hand) button is highlighted with a red box.
- The chat window shows a dropdown menu for "宛先" (Recipient) with "全員" (Everyone) selected, also highlighted with a red box.

2024/1/17 10:40-11:25

演習9

回歸2

統合教育機構
石丸美穂

線形回帰のコードのまとめ

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

STEP0 : ライブラリの読み込み

```
from sklearn.datasets import load_diabetes
dm = load_diabetes(as_frame = True, scaled = False)
y_dm = dm.target
x_dm = dm.data
x_bmi = x_dm[['bmi']]
```

STEP1 : データの準備

```
plt.scatter(x_bmi, y_dm)
```

```
from sklearn.linear_model import LinearRegression
model_bmi = LinearRegression()
```

STEP2 : 学習モデルの選択

```
model_bmi.fit(x_bmi, y_dm)
```

STEP3 : データを入れて学習させる

```
print(model_bmi.coef_)
print(model_bmi.intercept_)
```

STEP4 : 傾きと切片を求める

```
pre = pd.DataFrame([[20]], columns=['bmi'])
print(model_bmi.predict(pre))
```

STEP5 : 予測を行う

```
print(model_bmi.score(x_bmi, y_dm))
```

STEP6 : モデルの評価

線形回帰分析

前回の復習

STEP0 : 事前準備

STEP1 : データの用意

STEP2 : 学習モデルの選択

STEP3 : データを入れて学習させる

STEP4 : 傾きと切片を求める

STEP5 : 予測を行う

STEP6 : モデルの評価

STEP1 : 糖尿病データの内容確認

年齢 : 59歳
性別 : 女性
BMI : 32
...



target : 151



時間

ベースライン(基準)時点
→10個の特徴量の測定時点

1年後
正解値 (target :
糖尿病の進行度を量的に示し
たもの)の測定時点

ベースライン時点のx(特徴量)で1年後のtargetをy(正解値) との関連を調べる、学習モデルの作成を行う

回帰で学習を行うには、 x と y の配列がPythonでは指定されている

`x_bmi`, `y_dm`

`x`は2次元配列 (行列の形)

`y`は1次元配列 (ベクトルの形)

151.0, 75.0, 141.0...

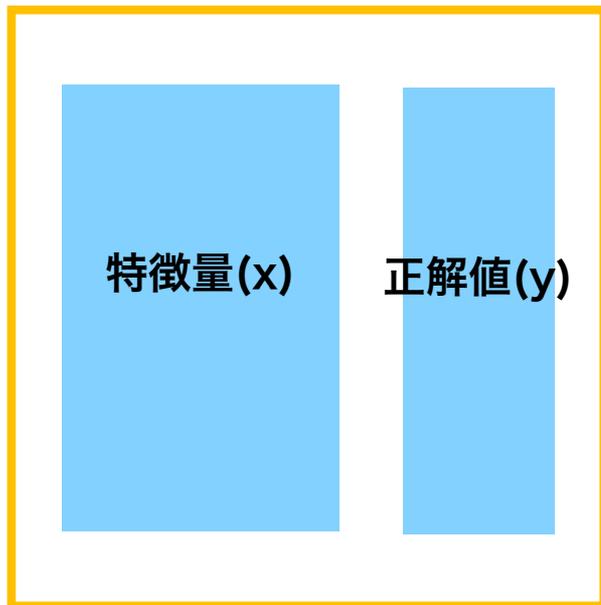
	BMI
0	32.1
1	21.6
2	30.5

配列を間違えるとエラーになりコードが動かないので、作成したデータの配列を確認するように気をつける

442 rows x 1 columns

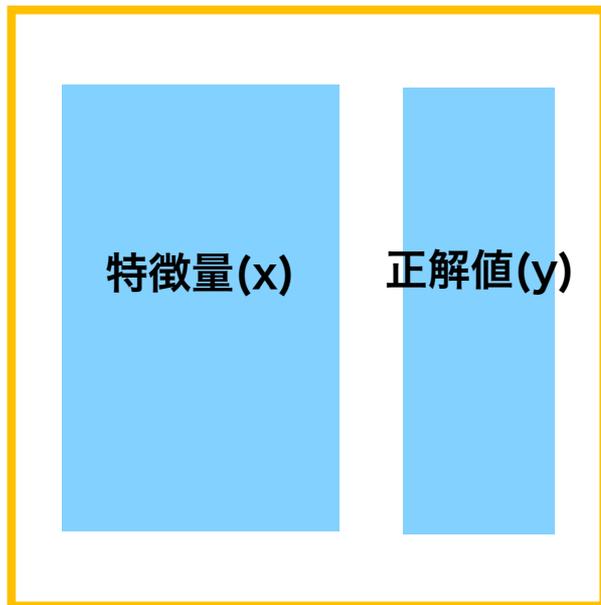
機械学習の流れのまとめ

STEP1:データの用意



機械学習の流れのまとめ

STEP1:データの用意



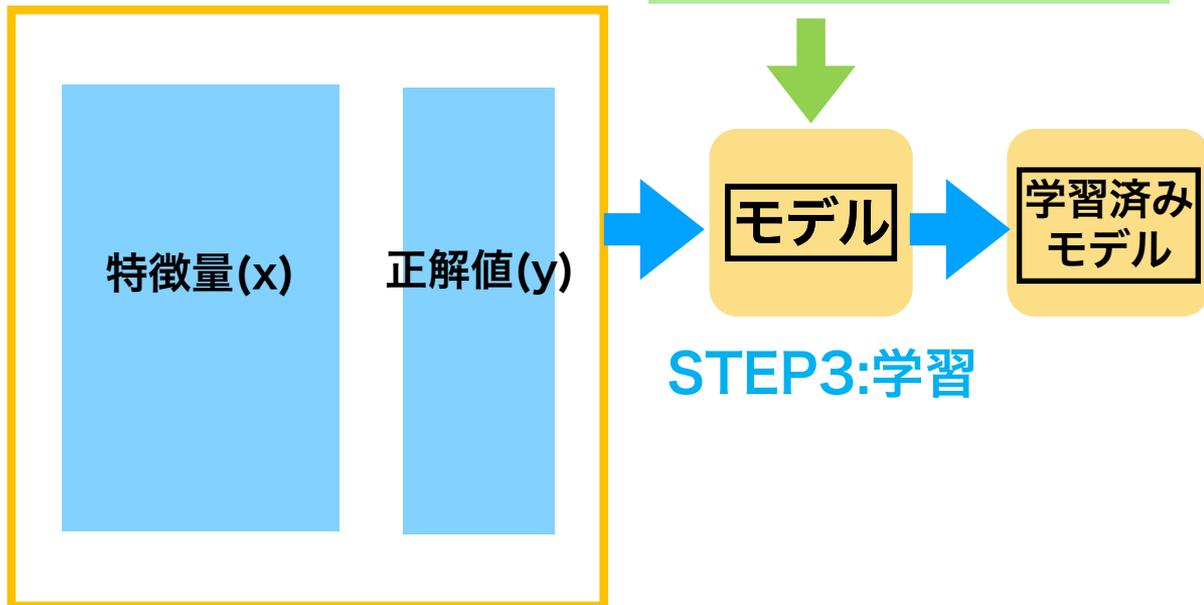
STEP2:学習モデルの選定



機械学習の流れのまとめ

STEP1:データの用意

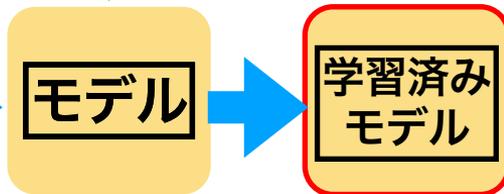
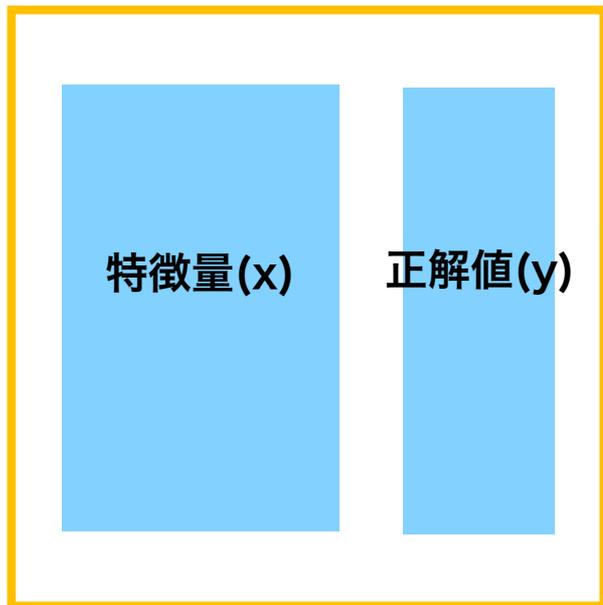
STEP2:学習モデルの選定



機械学習の流れのまとめ

STEP1:データの用意

STEP2:学習モデルの選定



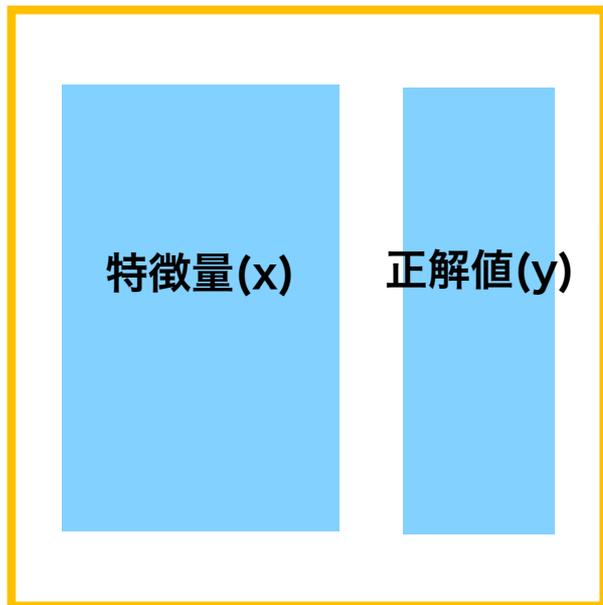
STEP3:学習

STEP4:傾きと切片

機械学習の流れのまとめ

STEP1:データの用意

STEP2:学習モデルの選定



STEP4:傾きと切片



STEP3:学習

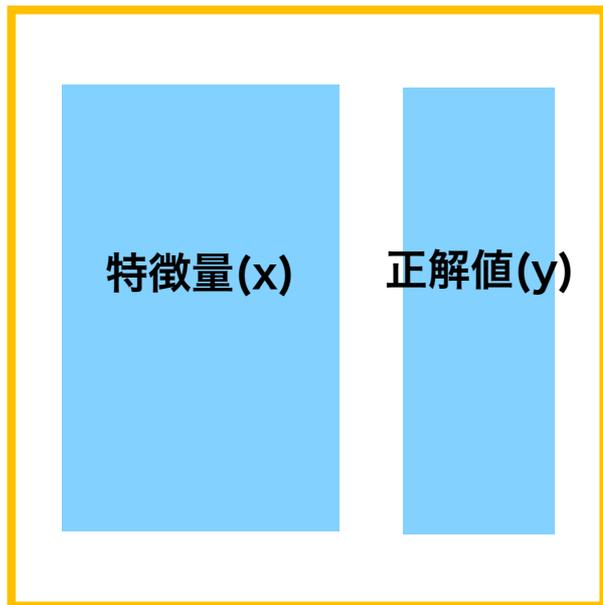
STEP5:予測

新しいxの値

機械学習の流れのまとめ

STEP1:データの用意

STEP2:学習モデルの選定



STEP4:傾きと切片



STEP3:学習

STEP5:予測

新しいxの値

正解との比較・評価

STEP6:モデルの評価

線形回帰分析

STEP0 : Google Colaboratoryの立ち上げ

STEP0 : 事前準備

STEP1 : データの用意

STEP2 : 学習モデルの選択

STEP3 : データを入れて学習させる

STEP4 : 傾きと切片を求める

STEP5 : 予測を行う

STEP6 : モデルの評価

検索google colab [Colaboratory へようこそ - Colaboratory - Google](#)

The screenshot shows the Google Colaboratory web interface. The 'File' menu is open, displaying options such as 'ノートブックを新規作成', 'ノートブックを開く', 'ノートブックをアップロード', '名前の変更', 'ドライブにコピーを保存', 'コピーを GitHub Gist として保存', 'GitHub にコピーを保存', '保存', '変更履歴', 'ダウンロード', and '印刷'. Two red arrows point to the 'File' menu and the 'ノートブックを開く' option. In the background, a video player is visible with the title '3 Cool Google Colab Features' and a play button icon. The text 'Colab へようこそ' and 'Colab とは' are also visible on the page.

演習9コード.ipynb

ノートブックを開く

例 >

最近 >

Google ドラ
イブ >

GitHub >

アップロード >

1
3



参照

または、ここにファイルをドラッグしてください

演習9コード.ipynb

線形回帰分析

STEP1：データの用意

STEP0：事前準備

STEP1：データの用意

STEP2：学習モデルの選択

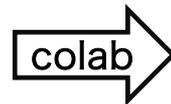
STEP3：データを入れて学習させる

STEP4：傾きと切片を求める

STEP5：予測を行う

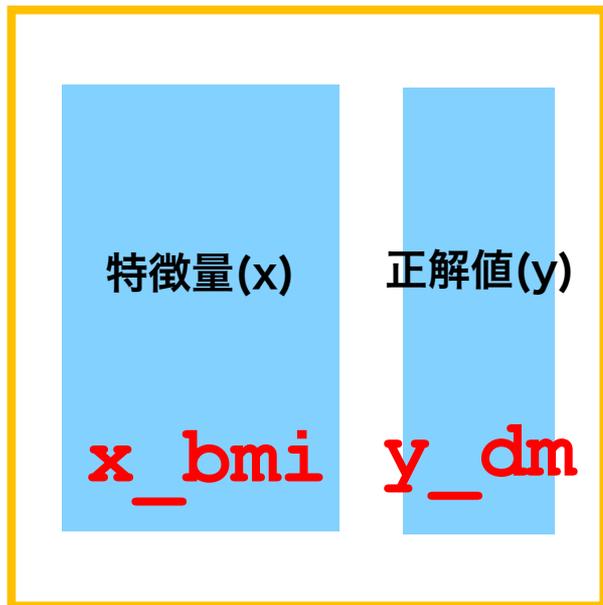
STEP6：モデルの評価

前回の続きのため、配布した演習9.ipynbの
コード9-1~コード9-2を実行してください



機械学習の流れのまとめ

STEP1:データの用意



STEP2:学習モデルの選択



線形回帰分析

STEP2：学習モデルの選択

STEP0：事前準備

STEP1：データの用意

STEP2：学習モデルの選択

STEP3：データを入れて学習させる

STEP4：傾きと切片を求める

STEP5：予測を行う

STEP6：モデルの評価

コード9-3 線形回帰を学習モデルに選択

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

クラスだけをインポート `from sklearn.linear_model import LinearRegression`

ライブラリ名

モジュール名

クラス名

クラス (Class)

属性 (引数、データ) と メソッド (処理) を一つにまとめた設計図のようなもの

クラス (設計図)に実際のデータを入れたものをインスタンスと呼ぶ

```
class BMI(weight, height)
```

属性

体重 (weight)
身長 (height)

メソッド

bmi_calc():
Weight /Height/ Height

Aさん



体重 70kg
身長 1.7m

```
Asan = BMI(70, 1.7)
```

Bさん



体重 45kg
身長 1.5m

```
Bsan = BMI(45, 1.5)
```

BMIクラスからインスタンス(Asan, Bsan)を作成

Asan.weight

➡ 70

Asan.bmi_calc()

➡ 24.2

Bsan.weight

➡ 45

Bsan.bmi_calc()

➡ 20

線形回帰分析

STEP2：学習モデルの選択

STEP0：事前準備

STEP1：データの用意

STEP2：学習モデルの選択

STEP3：データを入れて学習させる

STEP4：傾きと切片を求める

STEP5：予測を行う

STEP6：モデルの評価

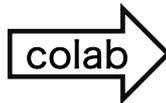
コード9-3 線形回帰を学習モデルに選択

```
model_bmi = LinearRegression()
```

- `LinearRegression()` クラスから `model_bmi` インスタンスを作成する
(この時点ではモデル構造のみが指定されている)

モデル名は何でも良い → 今回は `model_bmi`

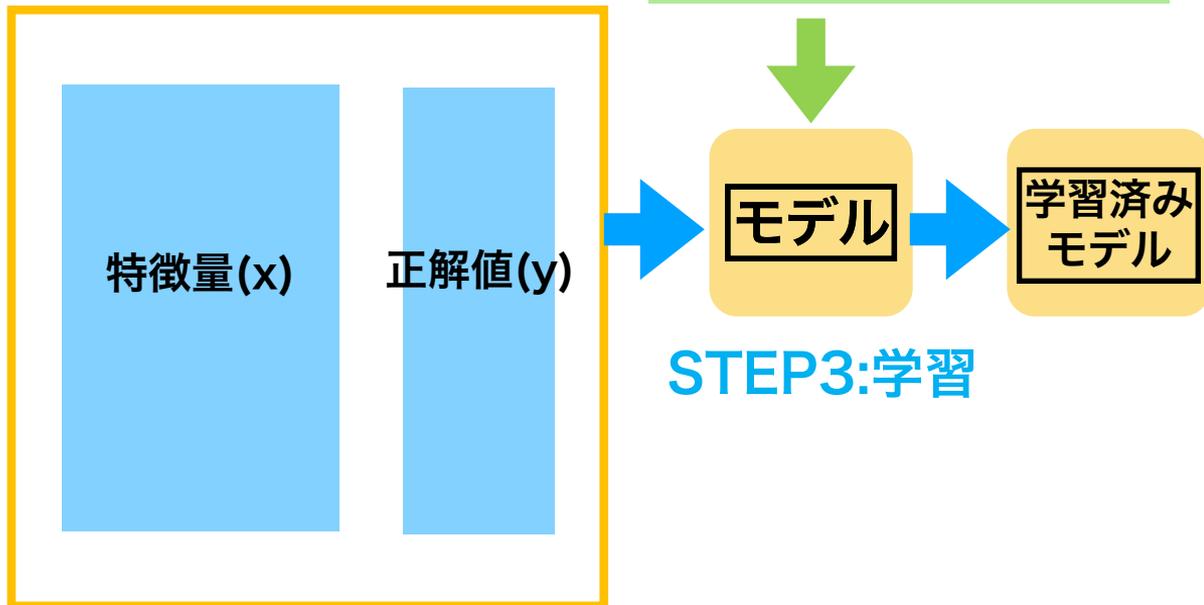
* `model1, mode2...` と作ると何を入れたものか忘れるので、
何のモデルなのか明示しておく方が便利



機械学習の流れのまとめ

STEP1:データの用意

STEP2:学習モデルの選定



線形回帰分析

STEP3：データを入れて学習させる

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習させる
STEP4：傾きと切片を求める
STEP5：予測を行う
STEP6：モデルの評価

コード9-4 データを入れて学習させる

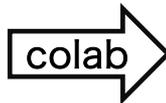
```
model_bmi.fit(x_bmi, y_dm)
```

- (モデル名).**fit(x,y)**メソッドで学習できる
- 今回の特徴量はx_bmi, 正解値はy_dmのデータ



これだけで「学習」は完了!

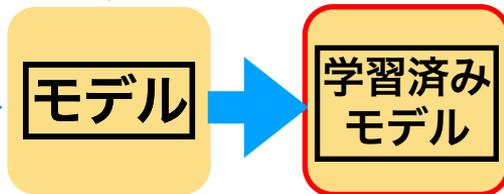
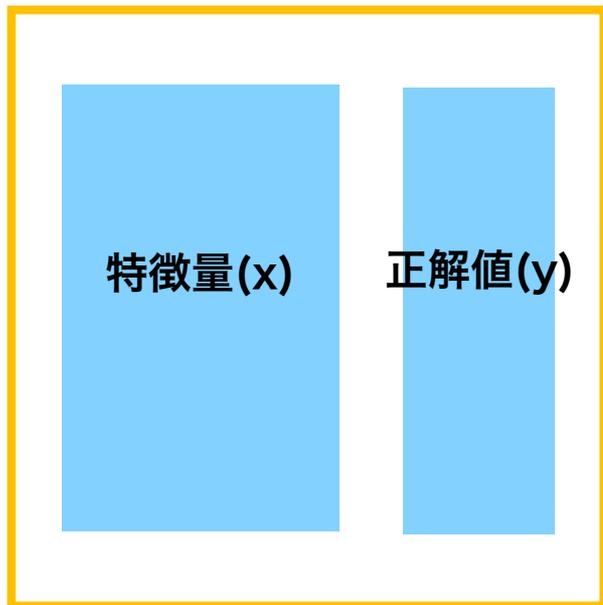
この後からはmodel_bmiは学習済みモデルになる



機械学習の流れのまとめ

STEP1:データの用意

STEP2:学習モデルの選定



STEP3:学習

STEP4:傾きと切片

線形回帰分析

STEP4：傾きと切片を求める

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習させる
STEP4：傾きと切片を求める
STEP5：予測を行う
STEP6：モデルの評価

コード9-5 学習済みモデルの傾きと切片を求める

```
print(model_bmi.coef_)  
print(model_bmi.intercept_)
```

→ `[[10.23312787]]`

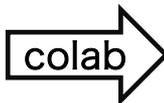
`[-117.77336657]`

(モデル名).coef_：係数 (coefficient)

(モデル名).intercept_：切片 (intercept)

model_bmiの回帰式の完成!!

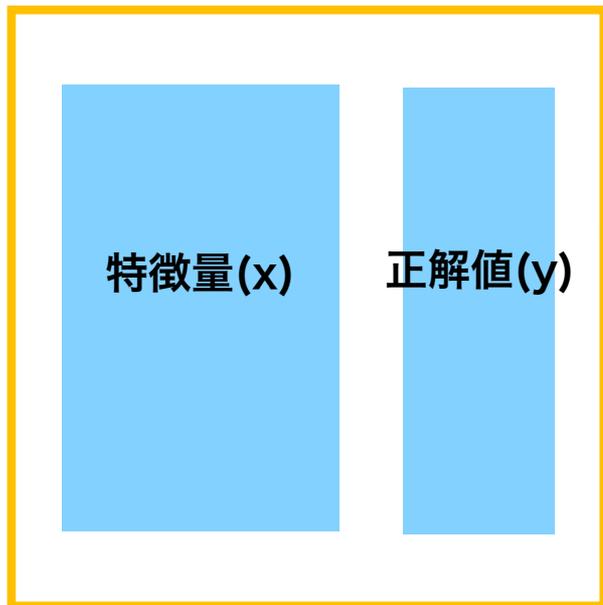
$$y = -117.77 + (10.23 \times \text{BMI})$$



機械学習の流れのまとめ

STEP1:データの用意

STEP2:学習モデルの選定



STEP4:傾きと切片



STEP3:学習

STEP5:予測

新しいxの値

線形回帰分析

STEP5：正解値の予測

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習させる
STEP4：傾きと切片を求める
STEP5：予測を行う
STEP6：モデルの評価

コード9-6 新しいxの値 (bmi=20の人) データ作成

```
pre = pd.DataFrame([[20]], columns=['bmi'])
```

- preという変数に、bmi=20の1行1列のデータを作成する
 - x (特徴量データ) は2次元配列である必要がある
- ➡ そのため、今回はpd.DataFrame型にして、column名に'bmi'をつける



線形回帰分析

STEP5：正解値の予測

- STEP0：事前準備
- STEP1：データの用意
- STEP2：学習モデルの選択
- STEP3：データを入れて学習させる
- STEP4：傾きと切片を求める
- STEP5：予測を行う**
- STEP6：モデルの評価

コード9-7 作成したpreのデータを確認

```
pre #データを出力
```



```
bmi  
0 20
```

```
pre.shape #データの配列構造を出力
```



```
(1, 1)
```

1行1列の2次元配列(行列)

```
type(pre) #データの型を出力
```



```
pandas.core.frame.DataFrame
```

線形回帰分析

STEP5：正解値の予測

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習させる
STEP4：傾きと切片を求める
STEP5：予測を行う
STEP6：モデルの評価

コード9-8 新しいxの値(pre)で正解値の予測

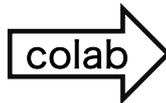
```
print(model_bmi.predict(pre))
```

➡ [86.88919084]

- (モデル名).predict(pre) でmodel_bmiの学習済みモデルのxにpreを代入し、yの値を求める
- print() で結果を出力する

STEP4で求めた回帰式にbmi=20を代入する

$$y = -117.77336657 + (10.2331787 \times 20) = \underline{86.88919084}$$



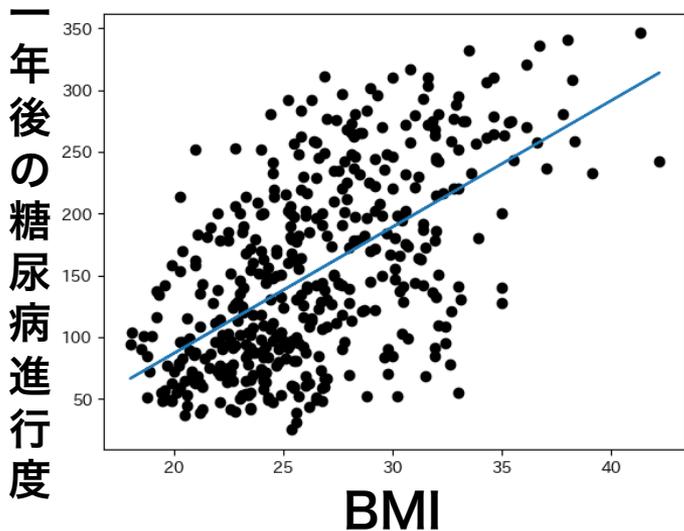
線形回帰分析

STEP5：プロット

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習させる
STEP4：傾きと切片を求める
STEP5：予測を行う
STEP6：モデルの評価

コード9-9 散布図と回帰線を描出する

```
plt.scatter(x_bmi, y_dm, color='black')  
plt.plot(x_bmi, model_bmi.predict(x_bmi))
```



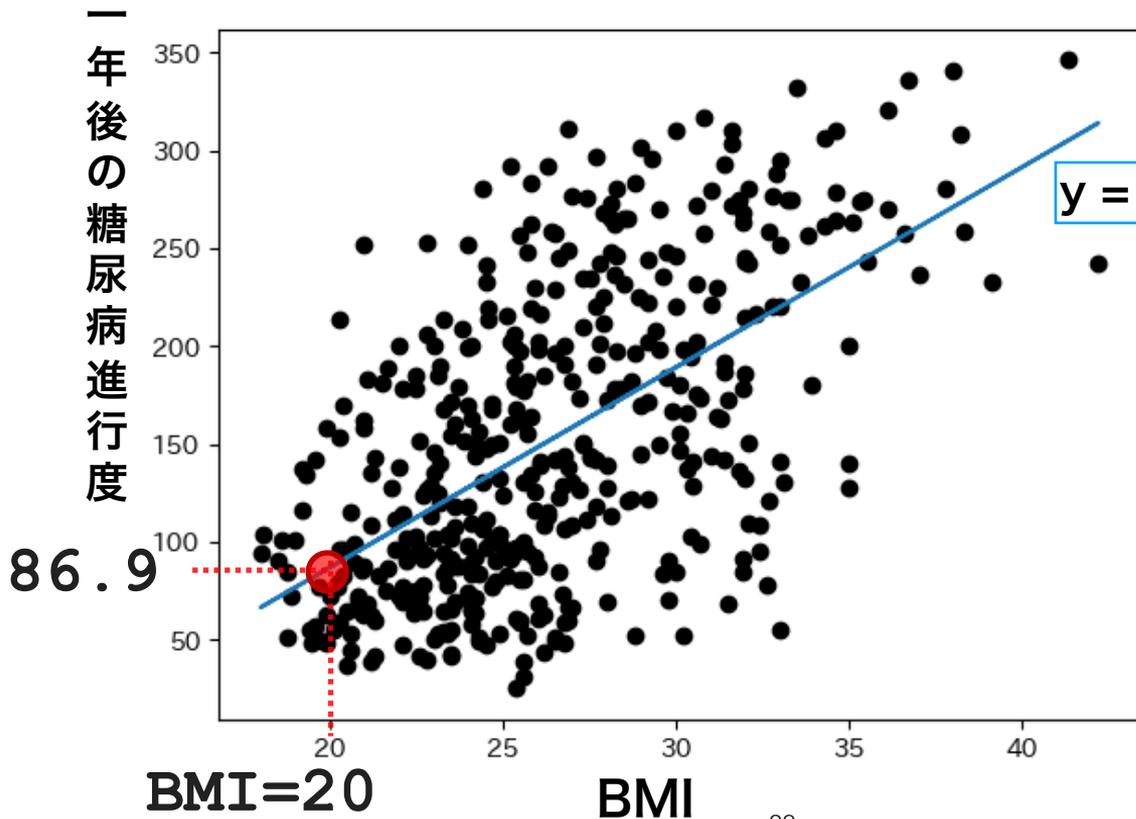
$$y = -117.77 + (10.23 \times \text{BMI})$$

- `plt.plot(x, y)` で各点を実線で結ぶ線が描出できる
- 回帰線のyは実測値ではなく、モデルで予測されたyの値であるため、`(モデル名).predict(x_bmi)`となる

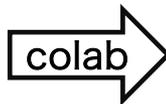
線形回帰分析

STEP5：プロット

- STEP0：事前準備
- STEP1：データの用意
- STEP2：学習モデルの選択
- STEP3：データを入れて学習させる
- STEP4：傾きと切片を求める
- STEP5：予測を行う**
- STEP6：モデルの評価



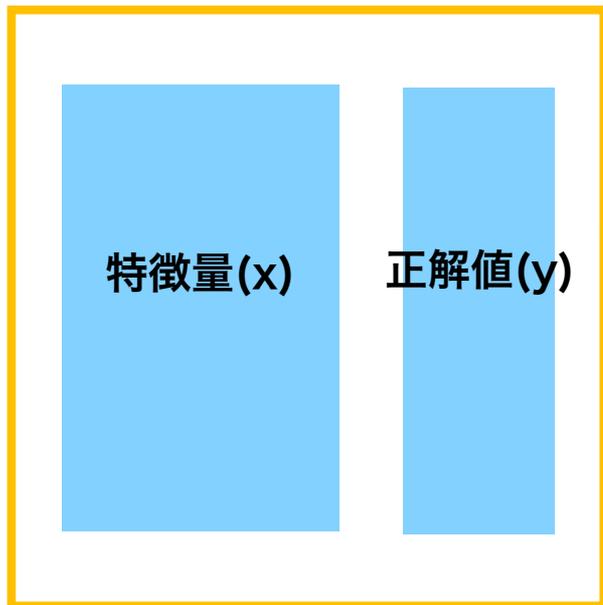
$$y = -117.77 + (10.23 \times \text{BMI})$$



機械学習の流れのまとめ

STEP1:データの用意

STEP2:学習モデルの選定



STEP4:傾きと切片



STEP3:学習

STEP5:予測

新しいxの値

正解との比較・評価

STEP6:モデルの評価

線形回帰分析

STEP6：モデルの予測性能評価

コード9-10 モデルがどれくらい予測できているかを調べる

```
print(model_bmi.score(x_bmi, y_dm))
```

➡ 0.3439237602253802

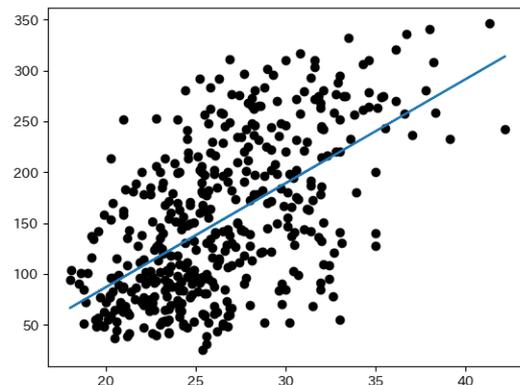
(モデル名).score(x,y) で決定係数 (R^2) を計算する

*決定係数：xとyの相関係数の二乗の値

1~0の範囲をとる

1に近いほどxでyを予測していると言える

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習させる
STEP4：傾きと切片を求める
STEP5：予測を行う
STEP6：モデルの評価



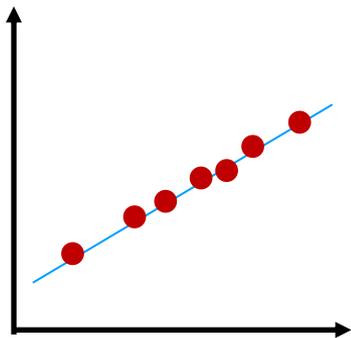
線形回帰分析

STEP6：モデルの予測性能評価

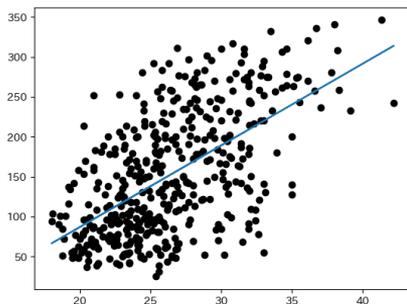
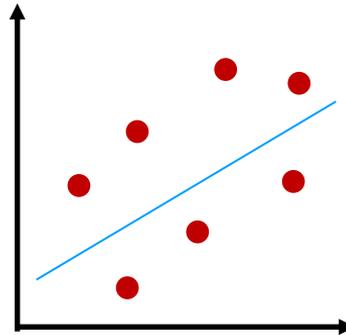
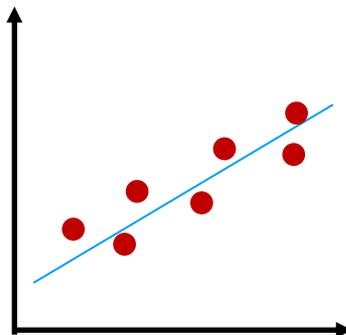
- STEP0：事前準備
- STEP1：データの用意
- STEP2：学習モデルの選択
- STEP3：データを入れて学習させる
- STEP4：傾きと切片を求める
- STEP5：予測を行う
- STEP6：モデルの評価**

決定係数 R^2

高い



低い



$$R^2=0.34$$



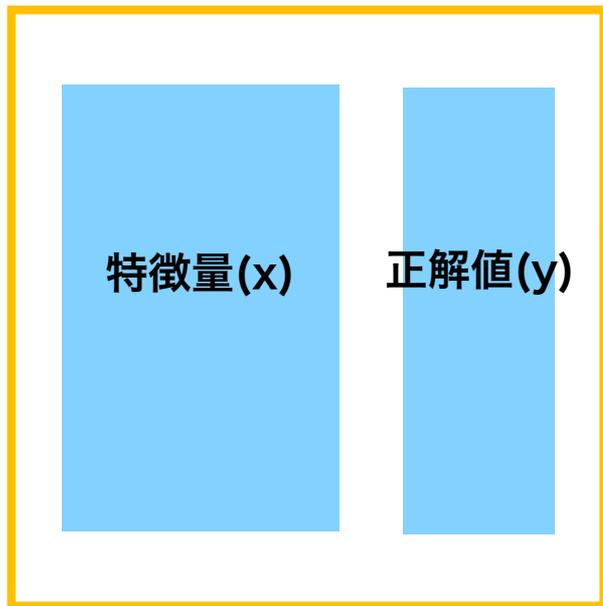
今回のモデルはそれほど
予測性能は高くない



機械学習の流れのまとめ

STEP1:データの用意

STEP2:学習モデルの選定



STEP4:傾きと切片



STEP3:学習

STEP5:予測

新しいxの値

正解との比較・評価

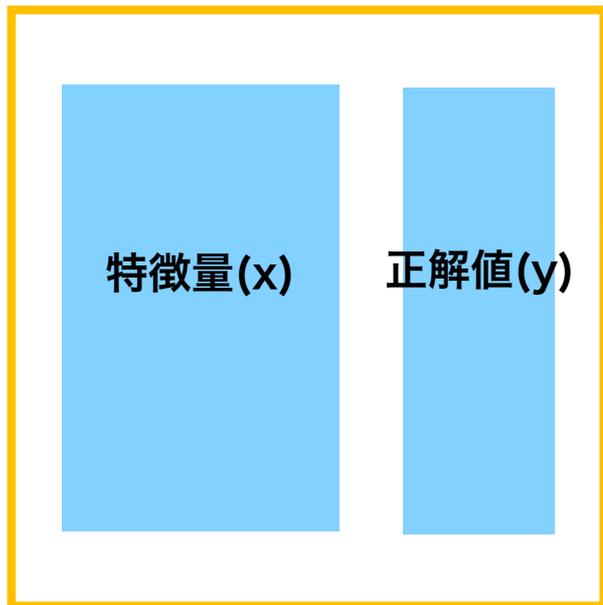
STEP6:モデルの評価

新しく `model_dm10` で 10 個の特徴
量で線形回帰を行う

機械学習の流れのまとめ

STEP1:データの用意

STEP2:学習モデルの選定



STEP4:傾きと切片



STEP3:学習

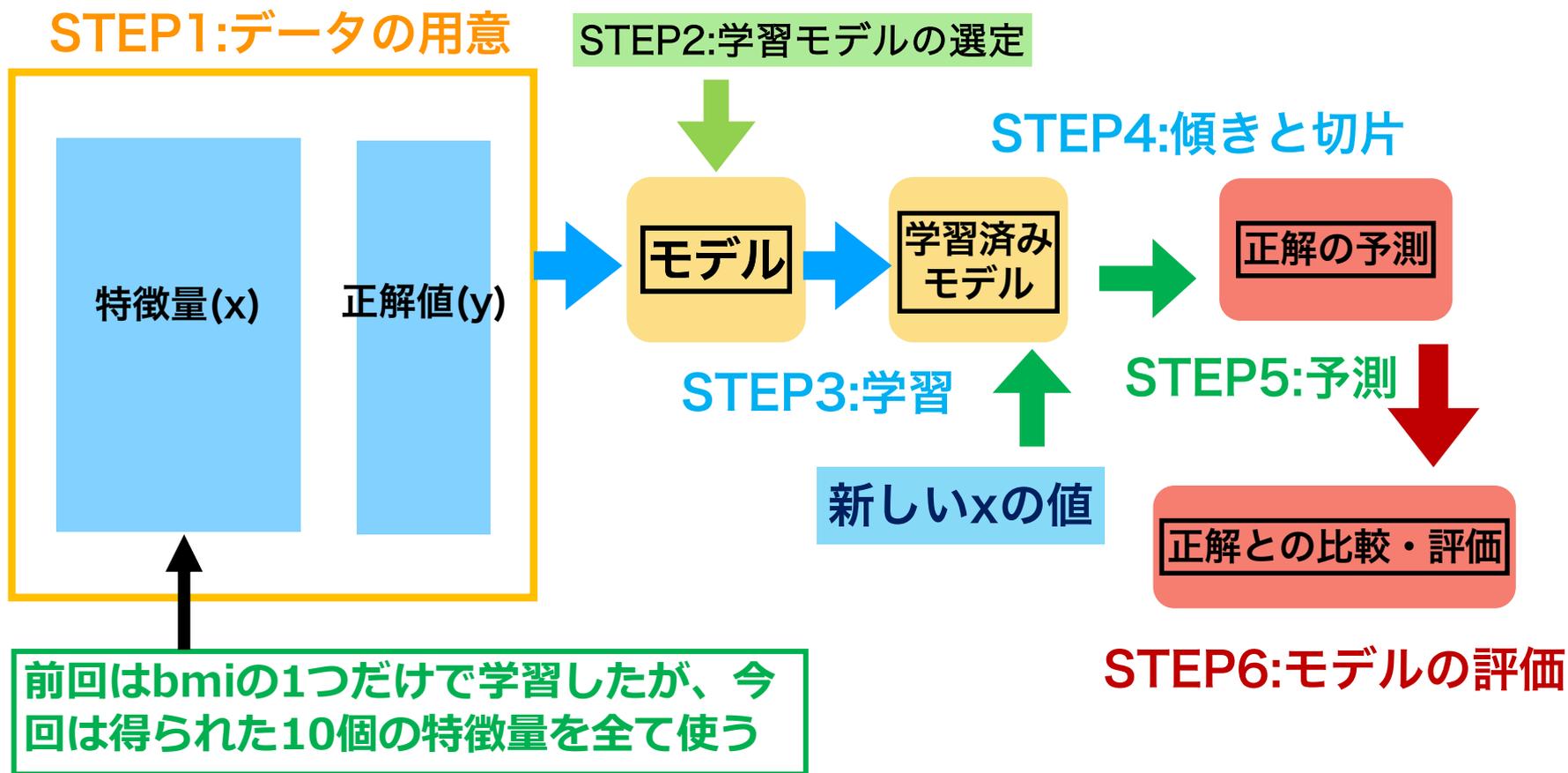
STEP5:予測

新しいxの値

正解との比較・評価

STEP6:モデルの評価

機械学習の流れのまとめ



重回帰分析

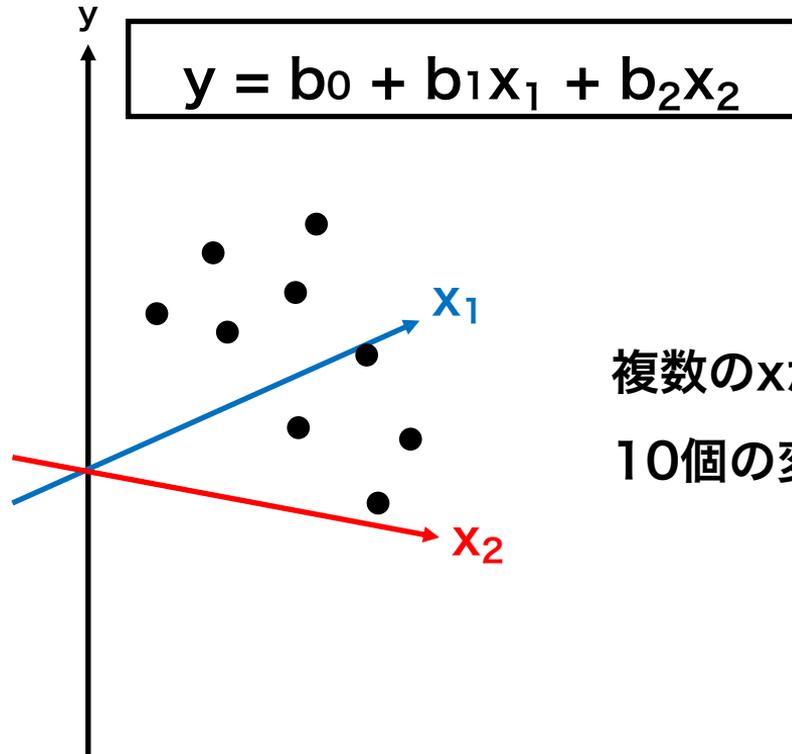
単回帰に対して、複数の変数で回帰することを**多変量回帰**や**重回帰**と呼ぶ

重回帰式

$$y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + b_4x_4 + b_5x_5 + b_6x_6 + b_7x_7 + b_8x_8 + b_9x_9 + b_{10}x_{10}$$

(y : 予測値、 x_n : 特徴量、 b_0 : 切片、 b_n : 傾き)

重回帰分析



複数のxがあるときは多次元になる

10個の変数があるときは10次元のデータになる

線形回帰分析：複数の特徴量で予測

- STEP0：事前準備
- STEP1：データの用意
- STEP2：学習モデルの選択
- STEP3：データを入れて学習させる
- STEP4：傾きと切片を求める
- STEP5：予測を行う
- STEP6：モデルの評価

コード9-11 新しくmodel_dm10インスタンスを作成し、10個の特徴量(x_dm)でy_dmを予測する

```
model_dm10 = LinearRegression()  
model_dm10.fit(x_dm, y_dm)
```

```
model_bmi.fit(x_bmi, y_dm)
```

x_dmの内容 (再掲)

	年齢	性別	BMI	平均血圧	TC (総コレステ ロール値)	LDL コレステ ロール	HDL コレステ ロール	TCH (TC/HDL)	LTG (血液中の 中性脂肪値 の対数)	GLU (血糖値)
	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6
count	442.000000	442.000000	442.000000	442.000000	442.000000	442.000000	442.000000	442.000000	442.000000	442.000000
mean	48.518100	1.468326	26.375792	94.647014	189.140271	115.439140	49.788462	4.070249	4.641411	91.260181
std	13.109028	0.499561	4.418122	13.831283	34.608052	30.413081	12.934202	1.290450	0.522391	11.496335
min	19.000000	1.000000	18.000000	62.000000	97.000000	41.600000	22.000000	2.000000	3.258100	58.000000
25%	38.250000	1.000000	23.200000	84.000000	164.250000	96.050000	40.250000	3.000000	4.276700	83.250000
50%	50.000000	1.000000	25.700000	93.000000	186.000000	113.000000	48.000000	4.000000	4.620050	91.000000
75%	59.000000	2.000000	29.275000	105.000000	209.750000	134.500000	57.750000	5.000000	4.997200	98.000000
max	79.000000	2.000000	42.200000	133.000000	301.000000	242.400000	99.000000	9.090000	6.107000	124.000000

colab

線形回帰分析：複数の特徴量で予測

- STEP0：事前準備
- STEP1：データの用意
- STEP2：学習モデルの選択
- STEP3：データを入れて学習させる
- STEP4：傾きと切片を求める
- STEP5：予測を行う
- STEP6：モデルの評価

コード9-12 model_dm10の傾きと切片を求める

```
print(model_dm10.coef_)  
print(model_dm10.intercept_)
```

e+00 : $\times 10^0$
e+01 : $\times 10^1$
e-01 : $\times 10^{-1}$

➡ [-3.63612242e-02 -2.28596481e+01 5.60296209e+00 1.11680799e+00
-1.08999633e+00 7.46450456e-01 3.72004715e-01 6.53383194e+00
6.84831250e+01 2.80116989e-01]

-334.5671385187877

傾きは特徴量ごとに出るので、10個出力

➡ 回帰式に当てはめる

$$y = -334.57 + (-0.036 \times \text{Age}) + (-22.85 \times \text{Sex}) + (5.60 \times \text{BMI}) + (1.12 \times \text{BP}) + (-1.09 \times \text{TC}) + (0.75 \times \text{LDL}) + (0.37 \times \text{HDL}) + (6.53 \times \text{TCH}) + (68.5 \times \text{LTG}) + (0.28 \times \text{GLU})$$

線形回帰分析：複数の特徴量で予測

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習させる
STEP4：傾きと切片を求める
STEP5：予測を行う
STEP6：モデルの評価

コード9-13 複数の特徴量を入れたモデルの評価を行う

```
print(model_dm10.score(x_dm, y_dm))
```

→ 0.5177484222203498

(再掲) `print(model_bmi.score(x_bmi, y_dm))`
→ 0.3439237602253802

今回は一つの特徴量(bmi)より10個の特徴量を入れて予測する方が予測能が向上した(1に近いほど予測能が高いと言える)

→ 特徴量が多いほどyを予測できる性能が高かった
一方、多ければ多いほど必ず予測性能が高くなるとも限らない

→ 機械学習では予測性能を高めることが非常に重要な目的

演習9：課題

Webclassで課題を提出してください。締め切りは**2024/01/31 23:59**まで

- (1)dmデータの特徴量データからage, sex, bmi, s6(血糖値)の4つを抽出してx_dm4を作成してください(コードを記載してください)
- (2)(1)の特徴量を使用して、線形回帰モデルをmodel_dm4と名前をつけて作成し、重回帰を行ってください。model_dm4の切片を求めて記載してください。 (出力結果)
- (3)(2)の決定係数 R^2 を回答してください。 (出力結果)