第4回　Pythonによるデータ加工処理の基礎（Pandas）

松尾・岩澤研究室
Matsuo-Iwasawa Lab UTokyo

講師：嵐　大樹

2024/10/29

自己紹介

嵐　大樹（あらし　ひろき）

所属

奈良県立医科大学医学部医学科2年

受講歴

GCI 2023 winter

DL基礎講座 2024

LLM講座2024（受講中）

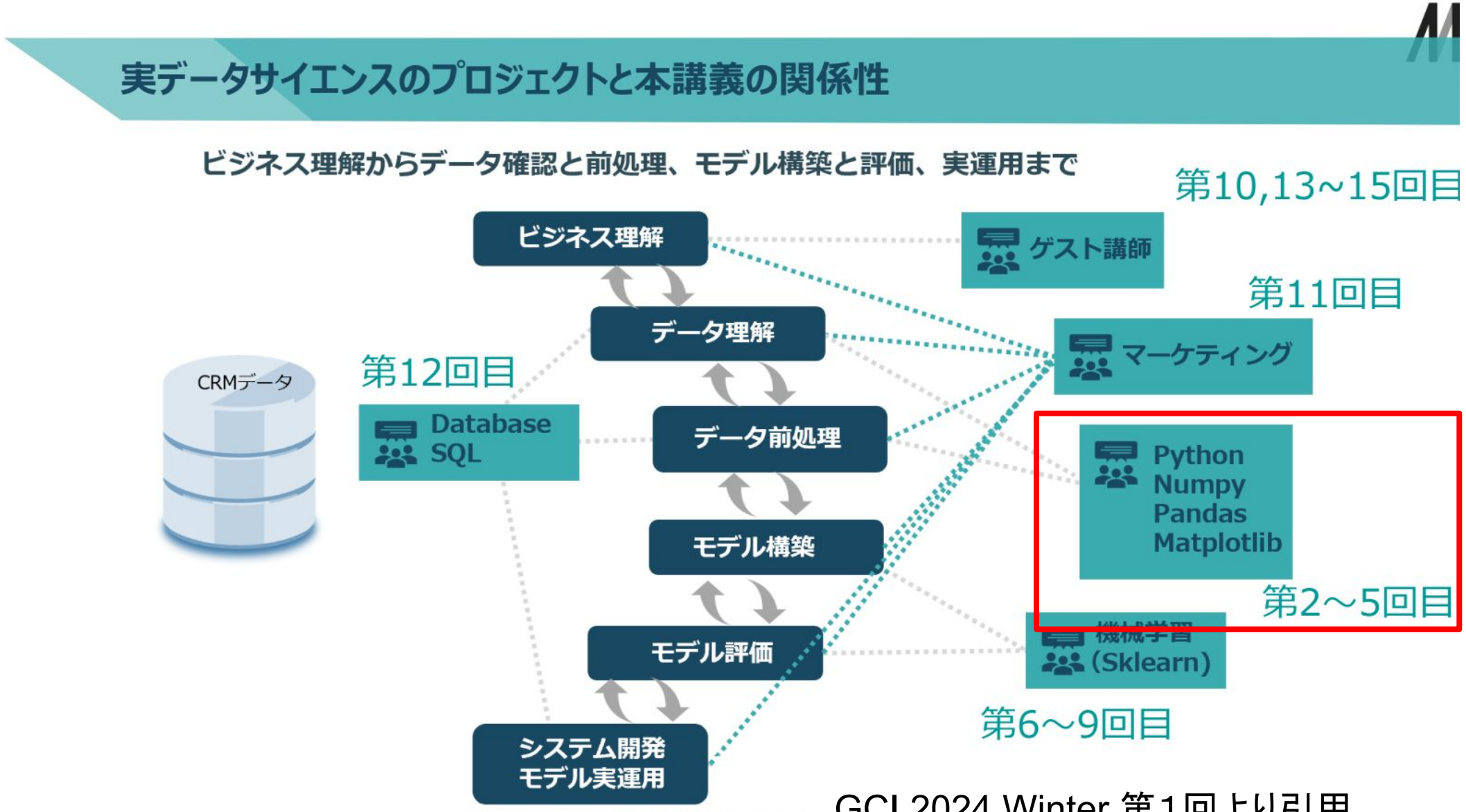
一言

私自身、GCI受講時はほとんど初学者であり苦労したので、その経験を活かし、どんな方でも消化しやすいような講義を心がけたいと思います。

本講義の構成

- イントロダクション
  - pandasの概要
- 演習形式の講義
  - データの読み込み
  - データの加工処理
- コンペ提出の操作デモ

本講義の位置付け



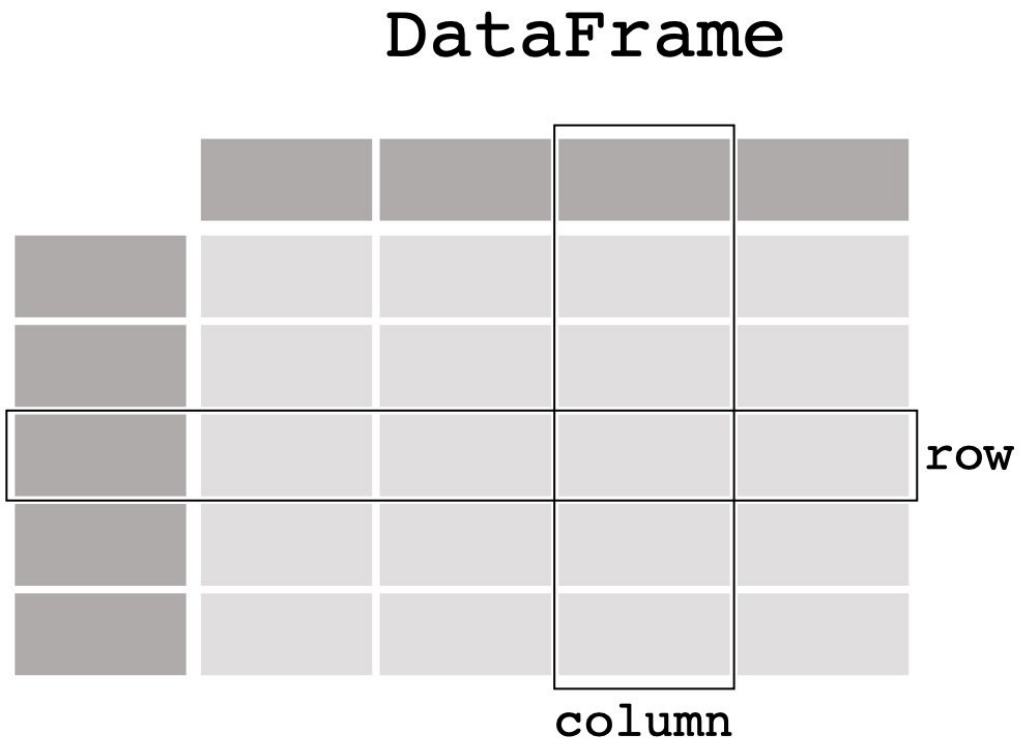
GCI 2024 Winter 第1回より引用

pandasとは？

- 主にDataFrame形式のデータの加工処理を行うためのライブラリ

DataFrame形式のデータとは？

- Excelのような二次元の表形式のデータ
- 例えば、様々な人について、それぞれの特徴をまとめたデータ



DataFrame (2次元)

		1列目	2列目	3列目	4列目
		ID	City	Birth_year	Name
1行目	0	100	Tokyo	1990	Hiroshi
2行目	1	101	Osaka	1989	Akiko
3行目	2	102	Kyoto	1992	Yuki
4行目	3	103	Hokkaido	1997	Satoru
5行目	4	104	Tokyo	1982	Steve

DataFrame形式のデータの加工処理とは？

- 様々にあり、その一部を今回の講義で1つ1つ学びます
- 例えば、ある人たちのいくつかの特徴のデータだけを抜き出したり、いくつかのデータセットを結合したり、など

DataFrame (2次元)

		1列目	2列目	3列目	4列目
		ID	City	Birth_year	Name
1行目	0	100	Tokyo	1990	Hiroshi
2行目	1	101	Osaka	1989	Akiko
3行目	2	102	Kyoto	1992	Yuki
4行目	3	103	Hokkaido	1997	Satoru
5行目	4	104	Tokyo	1982	Steve

pandasを学ぶ意義

- AIモデルはデータを元に学習する
- AIモデルをより「賢く」するためには、よくデータを見て、よりよいデータに整えてから、AIモデルに与える必要がある

⇨ よくデータを見る、よりよく整えるためには、加工処理が不可欠

演習で扱う加工処理の一覧

加工処理の種類	実装のイメージ	重要度のイメージ
ライブラリの読み込み	import ~~ as ~~	☆☆☆
DataFrameへの変換	DataFrame()	☆
データの読み込み	pd.read_csv()	☆☆☆
データの概観	df.info(), df.describe()	☆☆☆
データの選択と代入	df.loc(), df.iloc, df.at	☆
特定の条件のデータの抽出	df[df['~~'] == ~~]	☆☆☆
値のソート	df.sort_values()	☆☆
nan（null）の判定	df.isnull().sum()	☆
データの結合	pd.merge, pd.concat	☆☆☆
データの削除	df.drop()	☆☆
重複データの除去	df.drop_duplicates()	☆
マッピング処理	df['~~'].map(~~)	☆☆
ビン分割	pd.cut(~~, ~~)	☆☆
データの集約とグループ演算	df.groupby('~~')	☆☆
欠損データの扱い方	df.fillna(~~)	☆☆☆

演習中に意識するべきこと

- 全てを理解して覚えようとする必要はありません

⇨まずはざっくりとイメージを掴むことに努めてください

⇨最終的に調べて使えるようになることが目指すべきところです

- その上で、特によく使うものは「特に重要」と言うので、そこに関してはしっかり聞いてほしいです

本日の目標

- pandasとは何かを説明できるようになる
- pandasを学ぶ意義について説明できるようになる
- pandasを用いて様々なデータの加工処理が可能となることを具体的な実感を持ってイメージできるようになる
- pandasを用いた特に重要な加工処理について、その実装方法を理解する
- コンペの予測結果を提出できるようになる

Pandasのデータ構造

Pandasでよく使われるデータ構造は **Series** と **DataFrame**

Series（1次元）

		ID
1行目	0	100
2行目	1	101
3行目	2	102
4行目	3	103
5行目	4	104

DataFrame（2次元）

		1列目 ID	2列目 City	3列目 Birth_year	4列目 Name
1行目	0	100	Tokyo	1990	Hiroshi
2行目	1	101	Osaka	1989	Akiko
3行目	2	102	Kyoto	1992	Yuki
4行目	3	103	Hokkaido	1997	Satoru
5行目	4	104	Tokyo	1982	Steve

Pandasのデータ構造

DataFrameはNumpyの二次元配列に行ラベル・列ラベルがついたもの
行ラベルを index ・列ラベルを columns という

Series（1次元）

		ID
1行目	0	100
2行目	1	101
3行目	2	102
4行目	3	103
5行目	4	104

index　values

DataFrame（2次元）

		1列目	2列目	3列目	4列目
		ID	City	Birth_year	Name
1行目	0	100	Tokyo	1990	Hiroshi
2行目	1	101	Osaka	1989	Akiko
3行目	2	102	Kyoto	1992	Yuki
4行目	3	103	Hokkaido	1997	Satoru
5行目	4	104	Tokyo	1982	Steve

columns

index　values

データの抽出

データを抽出する方法は、**何を取得するか・何を指定するか** によって使い分ける

	ラベル指定	インデックス指定
SeriesやDataFrameを取得する	**loc** df.loc[0:3, ['ID','Birth_year']]	**iloc** df.iloc[0:4, [0,2]]
単独の要素を取得する	**at** df.at[2,'Birth_year']	**iat** df.iat[2, 2]

loc df.loc[0:3, ['ID','Birth_year']]

	ID	City	Birth_year	Name	Score
0	100	Tokyo	1990	Hiroshi	0
1	101	Osaka	1989	Akiko	10
2	102	Kyoto	1992	Yuki	20
3	103	Hokkaido	1997	Satoru	30
4	104	Tokyo	1982	Steve	40

iloc df.iloc[0:4, [0,2]]

	ID	City	Birth_year	Name	Score
0	100	Tokyo	1990	Hiroshi	0
1	101	Osaka	1989	Akiko	10
2	102	Kyoto	1992	Yuki	20
3	103	Hokkaido	1997	Satoru	30
4	104	Tokyo	1982	Steve	40

at df.at[2,'Birth_year']

	ID	City	Birth_year	Name	Score
0	100	Tokyo	1990	Hiroshi	0
1	101	Osaka	1989	Akiko	10
2	102	Kyoto	1992	Yuki	20
3	103	Hokkaido	1997	Satoru	30
4	104	Tokyo	1982	Steve	40

iat df.iat[2, 2]

	ID	City	Birth_year	Name	Score
0	100	Tokyo	1990	Hiroshi	0
1	101	Osaka	1989	Akiko	10
2	102	Kyoto	1992	Yuki	20
3	103	Hokkaido	1997	Satoru	30
4	104	Tokyo	1982	Steve	40

データの結合（merge）

内部結合 では，**両方のデータにキーが存在する場合に結合** する

df1

	id	city	birth_year	name
0	100	Tokyo	1990	Hiroshi
1	101	Osaka	1989	Akiko
2	102	Kyoto	1992	Yuki
3	103	Hokkaido	1997	Satoru
4	104	Tokyo	1982	Steeve
5	106	Tokyo	1991	Mituru
6	108	Osaka	1988	Aoi
7	110	Kyoto	1990	Tarou
8	111	Hokkaido	1995	Suguru
9	113	Tokyo	1981	Mitsuo

df2

	id	math	english	sex	index_num
0	100	50	90	M	0
1	101	43	30	F	1
2	102	33	20	F	2
3	105	76	50	M	3
4	107	98	30	M	4

【キーの指定】
- on, left_on, right_on,
  left_index, right_index引数で指定可能
- 指定なしの場合，
  両方のデータに共通のカラムがキーとなる

pd.merge(df1, df2)

id（キー）がdf1・df2ともに
存在する行のみ結合される

	id	city	birth_year	name	math	english	sex	index_num
0	100	Tokyo	1990	Hiroshi	50	90	M	0
1	101	Osaka	1989	Akiko	43	30	F	1
2	102	Kyoto	1992	Yuki	33	20	F	2

データの結合（merge）

左外部結合 では，**左のデータのキーをもとに結合** する

df1

	id	city	birth_year	name
0	100	Tokyo	1990	Hiroshi
1	101	Osaka	1989	Akiko
2	102	Kyoto	1992	Yuki
3	103	Hokkaido	1997	Satoru
4	104	Tokyo	1982	Steeve
5	106	Tokyo	1991	Mituru
6	108	Osaka	1988	Aoi
7	110	Kyoto	1990	Tarou
8	111	Hokkaido	1995	Suguru
9	113	Tokyo	1981	Mitsuo

df2

	id	math	english	sex	index_num
0	100	50	90	M	0
1	101	43	30	F	1
2	102	33	20	F	2
3	105	76	50	M	3
4	107	98	30	M	4

pd.merge(df1, df2, how = ‘left‘)

左のデータ（df1）の
情報量は失われない

	id	city	birth_year	name	math	english	sex	index_num
0	100	Tokyo	1990	Hiroshi	50.0	90.0	M	0.0
1	101	Osaka	1989	Akiko	43.0	30.0	F	1.0
2	102	Kyoto	1992	Yuki	33.0	20.0	F	2.0
3	103	Hokkaido	1997	Satoru	NaN	NaN	NaN	NaN
4	104	Tokyo	1982	Steeve	NaN	NaN	NaN	NaN
5	106	Tokyo	1991	Mituru	NaN	NaN	NaN	NaN
6	108	Osaka	1988	Aoi	NaN	NaN	NaN	NaN
7	110	Kyoto	1990	Tarou	NaN	NaN	NaN	NaN
8	111	Hokkaido	1995	Suguru	NaN	NaN	NaN	NaN
9	113	Tokyo	1981	Mitsuo	NaN	NaN	NaN	NaN

データの結合（merge）

完全外部結合 では，**どちらかのデータにキーがあれば結合** する

df1

	id	city	birth_year	name
0	100	Tokyo	1990	Hiroshi
1	101	Osaka	1989	Akiko
2	102	Kyoto	1992	Yuki
3	103	Hokkaido	1997	Satoru
4	104	Tokyo	1982	Steeve
5	106	Tokyo	1991	Mituru
6	108	Osaka	1988	Aoi
7	110	Kyoto	1990	Tarou
8	111	Hokkaido	1995	Suguru
9	113	Tokyo	1981	Mitsuo

df2

	id	math	english	sex	index_num
0	100	50	90	M	0
1	101	43	30	F	1
2	102	33	20	F	2
3	105	76	50	M	3
4	107	98	30	M	4

どちらのデータも
情報量は失われない

```
pd.merge(df1, df2, how = 'outer')
```

	id	city	birth_year	name	math	english	sex	index_num
0	100	Tokyo	1990.0	Hiroshi	50.0	90.0	M	0.0
1	101	Osaka	1989.0	Akiko	43.0	30.0	F	1.0
2	102	Kyoto	1992.0	Yuki	33.0	20.0	F	2.0
3	103	Hokkaido	1997.0	Satoru	NaN	NaN	NaN	NaN
4	104	Tokyo	1982.0	Steeve	NaN	NaN	NaN	NaN
5	106	Tokyo	1991.0	Mituru	NaN	NaN	NaN	NaN
6	108	Osaka	1988.0	Aoi	NaN	NaN	NaN	NaN
7	110	Kyoto	1990.0	Tarou	NaN	NaN	NaN	NaN
8	111	Hokkaido	1995.0	Suguru	NaN	NaN	NaN	NaN
9	113	Tokyo	1981.0	Mitsuo	NaN	NaN	NaN	NaN
10	105	NaN	NaN	NaN	76.0	50.0	M	3.0
11	107	NaN	NaN	NaN	98.0	30.0	M	4.0

データの結合（concat）

縦結合では，**キーを指定せずにデータを積み上げて結合**　する

df1

	id	city	birth_year	name
0	100	Tokyo	1990	Hiroshi
1	101	Osaka	1989	Akiko
2	102	Kyoto	1992	Yuki
3	103	Hokkaido	1997	Satoru
4	104	Tokyo	1982	Steeve
5	106	Tokyo	1991	Mituru
6	108	Osaka	1988	Aoi
7	110	Kyoto	1990	Tarou
8	111	Hokkaido	1995	Suguru
9	113	Tokyo	1981	Mitsuo

df3

	id	city	birth_year	name
0	117	Chiba	1990	Suguru
1	118	Kanagawa	1989	Kouichi
2	119	Tokyo	1992	Satochi
3	120	Fukuoka	1997	Yukie
4	125	Okinawa	1982	Akari

pd.concat([df1,df3])

	id	city	birth_year	name
0	100	Tokyo	1990	Hiroshi
1	101	Osaka	1989	Akiko
2	102	Kyoto	1992	Yuki
3	103	Hokkaido	1997	Satoru
4	104	Tokyo	1982	Steeve
5	106	Tokyo	1991	Mituru
6	108	Osaka	1988	Aoi
7	110	Kyoto	1990	Tarou
8	111	Hokkaido	1995	Suguru
9	113	Tokyo	1981	Mitsuo
0	117	Chiba	1990	Suguru
1	118	Kanagawa	1989	Kouichi
2	119	Tokyo	1992	Satochi
3	120	Fukuoka	1997	Yukie
4	125	Okinawa	1982	Akari

共通カラムで
積み上げる

データの結合（concat）

横結合では，カラムを無視して**インデックスをもとに結合**する

df1

	id	city	birth_year	name
0	100	Tokyo	1990	Hiroshi
1	101	Osaka	1989	Akiko
2	102	Kyoto	1992	Yuki
3	103	Hokkaido	1997	Satoru
4	104	Tokyo	1982	Steeve
5	106	Tokyo	1991	Mituru
6	108	Osaka	1988	Aoi
7	110	Kyoto	1990	Tarou
8	111	Hokkaido	1995	Suguru
9	113	Tokyo	1981	Mitsuo

df2

	id	math	english	sex	index_num
0	100	50	90	M	0
1	101	43	30	F	1
2	102	33	20	F	2
3	105	76	50	M	3
4	107	98	30	M	4

pd.concat([df1, df2], axis=1)

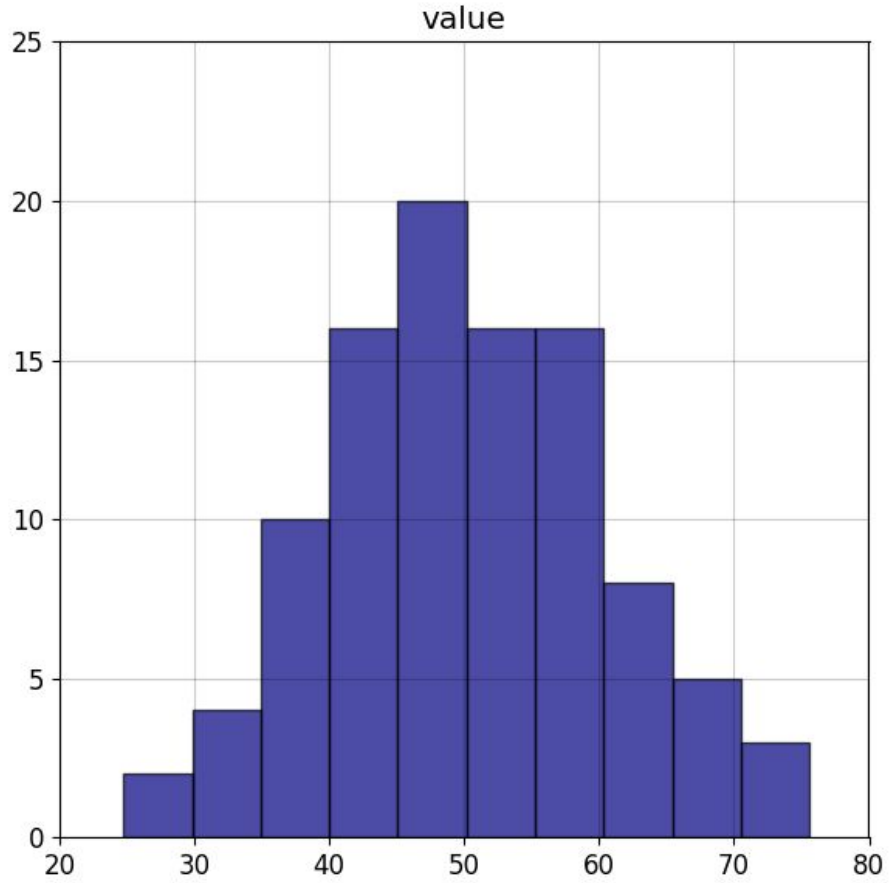
不足しているデータは
欠損値となる

	id	city	birth_year	name	id	math	english	sex	index_num
0	100	Tokyo	1990	Hiroshi	100	50.0	90.0	M	0.0
1	101	Osaka	1989	Akiko	101	43.0	30.0	F	1.0
2	102	Kyoto	1992	Yuki	102	33.0	20.0	F	2.0
3	103	Hokkaido	1997	Satoru	105	76.0	50.0	M	3.0
4	104	Tokyo	1982	Steeve	107	98.0	30.0	M	4.0
5	106	Tokyo	1991	Mituru	NaN	NaN	NaN	NaN	NaN
6	108	Osaka	1988	Aoi	NaN	NaN	NaN	NaN	NaN
7	110	Kyoto	1990	Tarou	NaN	NaN	NaN	NaN	NaN
8	111	Hokkaido	1995	Suguru	NaN	NaN	NaN	NaN	NaN
9	113	Tokyo	1981	Mitsuo	NaN	NaN	NaN	NaN	NaN

ビン分割

ビン分割を行うときは、データを**等間隔または等個数で分割** する

data = DataFrame({“value”:np.random.normal(50,10,100)}) # 正規分布に従う乱数



cut()：データを 等間隔 で分割

pd.cut(data[“value”], bins=5)

	(24.702, 34.927]	(34.927, 45.101]	(45.101, 55.275]	(55.275, 65.449]	(65.449, 75.624]
count	6	26	36	24	8

qcut()：データを 等個数 で分割

pd.qcut(data[“value”], q=5)

	(24.752, 42.199]	(42.199, 46.881]	(46.881, 53.138]	(53.138, 58.002]	(58.002, 75.624]
count	20	20	20	20	20

欠損データと異常値の取り扱いの基礎

欠損値や異常値は、なんらかの方法で**処理する必要**がある

欠損値：欠損している値のこと

- 列や行ごと除去する
- 別の値で置き換える

異常値：異常な値のこと

- 列や行ごと除去する
- 別の値で置き換える

	ID	City	Birth_year	Name
0	100	Tokyo	3000	Hiroshi
1	101	Osaka	1989	Akiko
2	102	Kyoto	1992	Yuki
3	103	NaN	1997	Satoru
4	104	Tokyo	1982	Steve