

cs-12.プログラミング・データサイエンス・AIの融合

(コンピューターサイエンス)

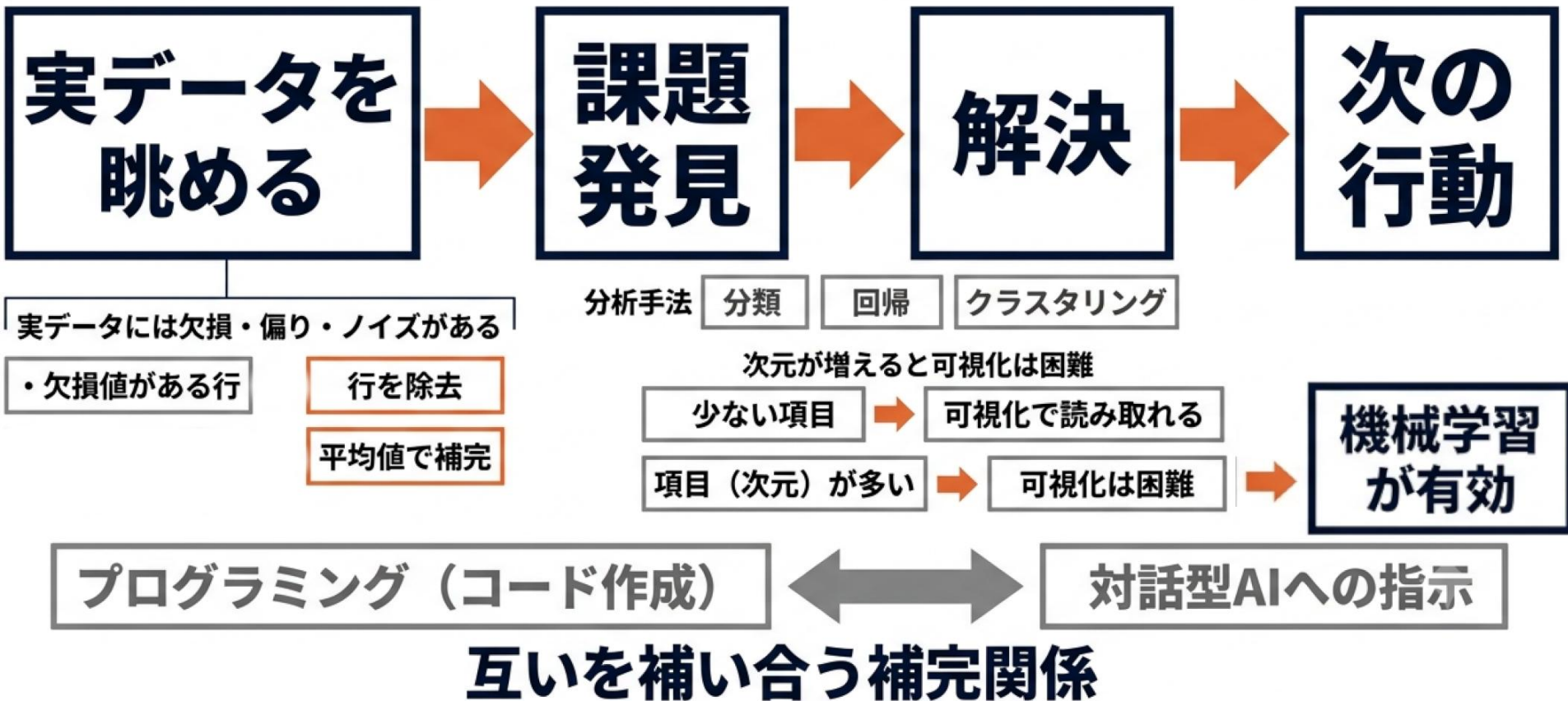
金子邦彦



「コンピュータサイエンス」第12回の内容



データ分析は『眺める→課題発見→解決→次の行動』の流れで進む



データ分析を、目的→手法→比較→実践の4ステップで学ぶ

1

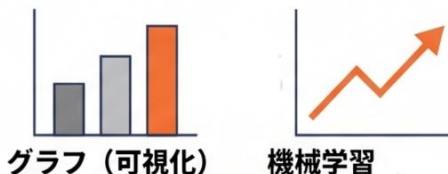
データ分析 とは何か

なぜ・何のために
分析するのかを理解する

2

分析手法の基礎

可視化と機械学習を身近なテーマで学ぶ



3

アプローチ の比較

自分で
コードを
書く

VS

生成AIに
任せる

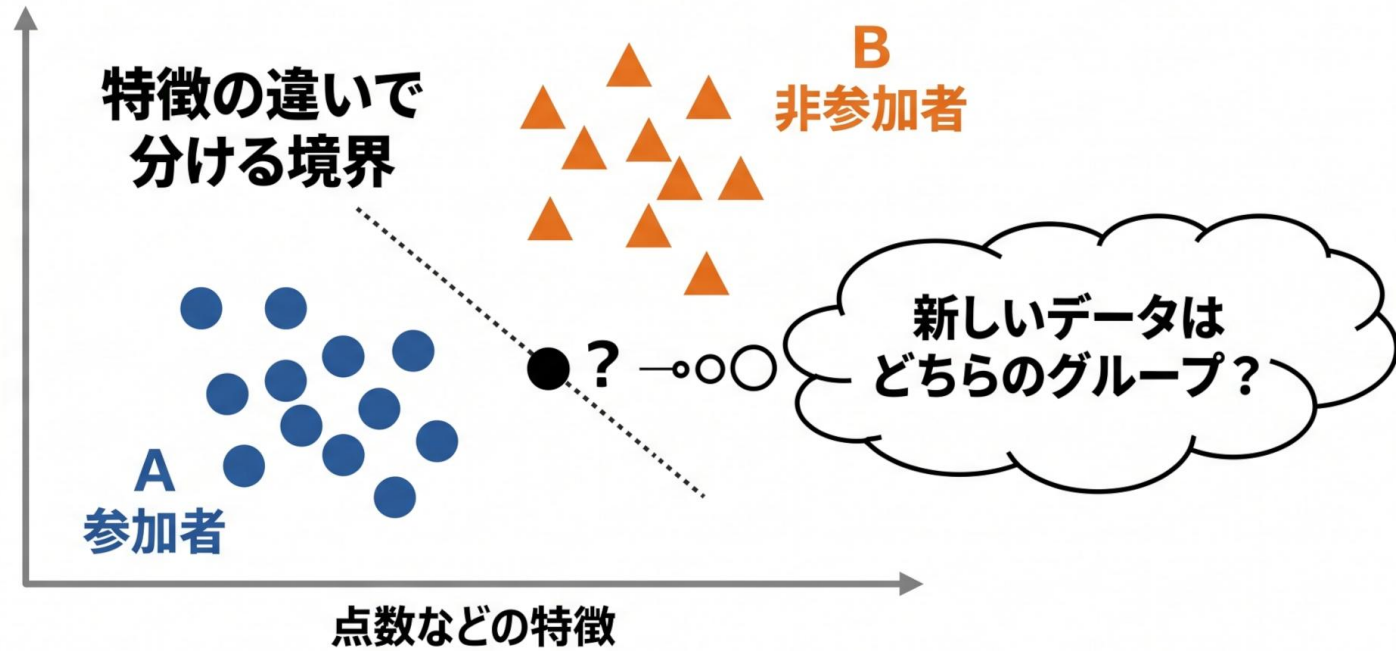
4

実行環境 で体験

オンライン実行環境で
実際にコードを動かす



分類とは、特徴の違いを手がかりに、新しいデータがどちらのグループに属するかを見分ける手法



本資料の流れ

- 12.1 分類済みデータを眺める →
- 12.2 機械学習で分類・回帰を行う

データはすべて架空

講習会Aの点数・売上・生活習慣アンケート（12人分）はすべて架空。手法は基本的なものに限定

検証は省略

本来は学習用データと検証用データを分けるのが原則。本資料では簡単のため省略

機械学習

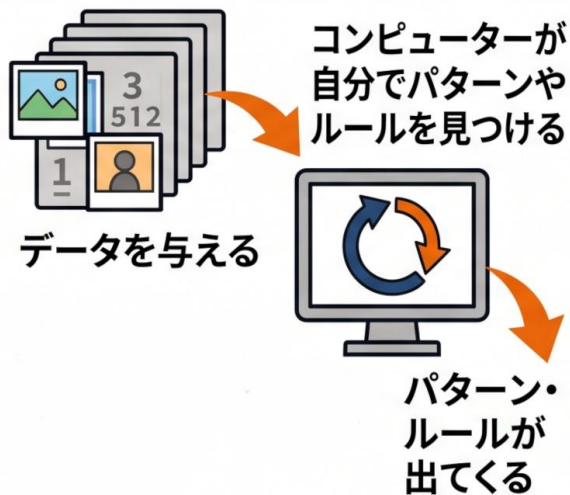
機械学習とは、ルールを教えるのではなく、データからコンピューター自身にパターンやルールを見つけさせる仕組み

これまでのやり方



人が答え方を全部教える

機械学習



人はデータを渡す

与えるのは“データ”

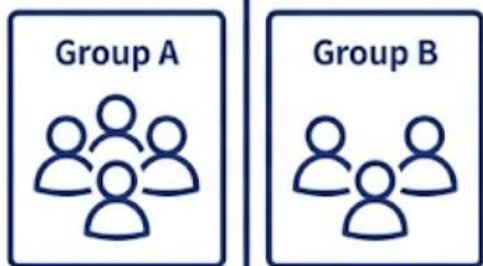


機械学習によるデータ分析



機械学習は『分類』『回帰』『クラスタリング』の3つの型で身近な問題を解く

違いの発見 = 分類



講習会を受けた人／受けていない人

講習会Aを受けた人と受けていない人で、アンケート回答の傾向にどんな違いがあるかを見つける

将来の予測 = 回帰

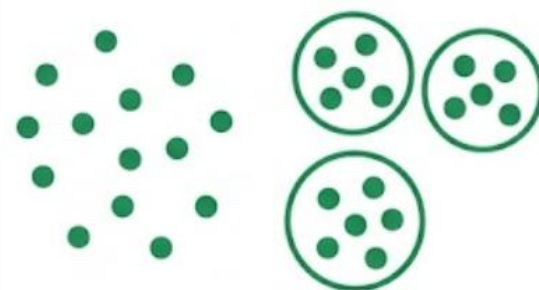
(変数と変数の関係の推定)



過去2年の売上 → 来年を予測

ある店の過去2年分の月別売上から、来年の売上の推移を機械に予測させる

自動グループ分け = クラスタリング



似た回答者どうしを自動でまとめる

生活習慣アンケートの回答者を、似た者どうしのグループに分ける

オンラインの pythononline.dev の使用

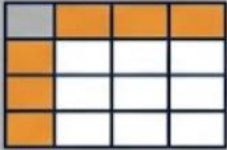


Python の各種ライブラリを使用。Pythononline.dev なら特別な準備（インストールなど）なしで、インポートだけで使える

URL: <https://pythononline.dev/>

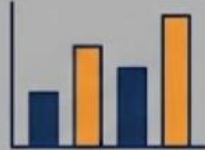
pandas

表データの処理



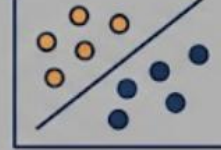
matplotlib

グラフを描く



scikit-learn

機械学習



OneCompiler に最初から用意済み
→ 準備不要ですぐ使える

Python



これら3つは学習の本筋ではないので一度だけ確認し、以降は意識せず読み進めてOK

グラフの表示

- `plt.show()` で表示する
`plt.show()` だけでOK
- 行わないこと) :
描画バックエンド指定
(Agg等) / `savefig`
でのファイル保存 /
`plt.figure()` での細かな設定

日本語の扱い

- 軸ラベルと凡例は英語で書く
軸・凡例は英語
- 理由：日本語フォント設定ライブラリが不要になり、簡単だから

`random_state=0`

- コード中に登場する定型
毎回同じ結果にする
- 意味：結果を毎回同じにするための決まり文句

※ 本筋の理解には影響しません。今はこの3点だけ押さえれば十分です。

12-1. データ分析

データ分析

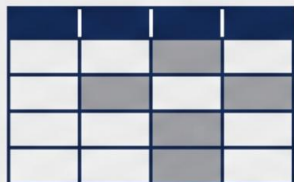


データ分析とは、実データから課題を見つけ、解決して“次の行動”につなげること

【実データの特徴】

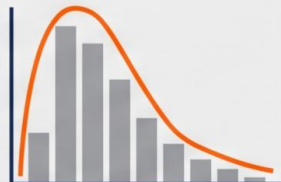
実データは“そのままでは分析できない”ことがある

欠損



アンケートの未回答・記録漏れ

偏り



集め方による傾向の偏り

ノイズ

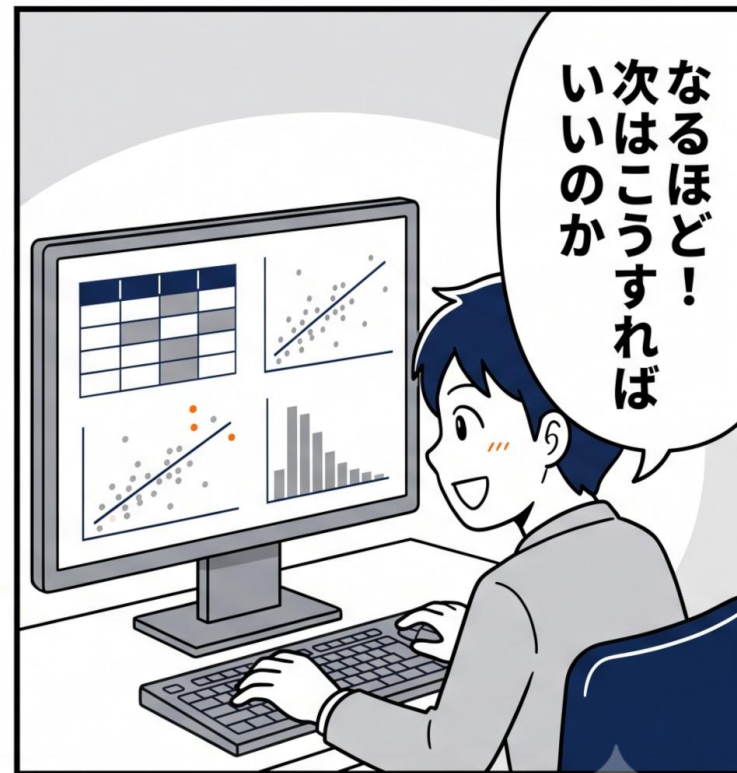


記入ミス・測定誤差

【分析の流れ】

いきなり機械学習ではなく、まず“全体を眺める”ことが第一歩

- ① 実データを眺める
- ② 課題を発見する
(傾向・問題点・改善点に気づく)
- ③ 課題を解決する
(分析結果を根拠に解決策を導く)
- ④ 次の行動につながる



1: 受講者と
非受講者の違い

2: 来年の売上
を予測

実データは、そのままでは分析できないことがある生のデータ

整えたデータ

○	○	○
○	○	○
○	○	○

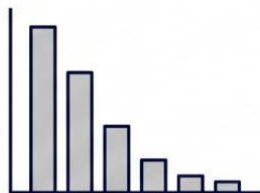
実データの3つの特徴

欠損



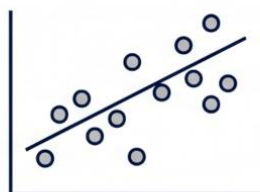
値が抜けている
(未回答・記録漏れ)

偏り



特定の傾向にかたよる
(集め方の影響)

ノイズ

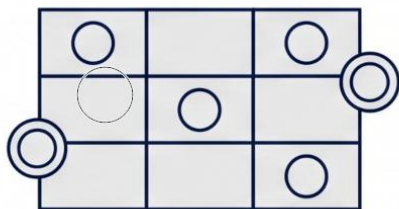


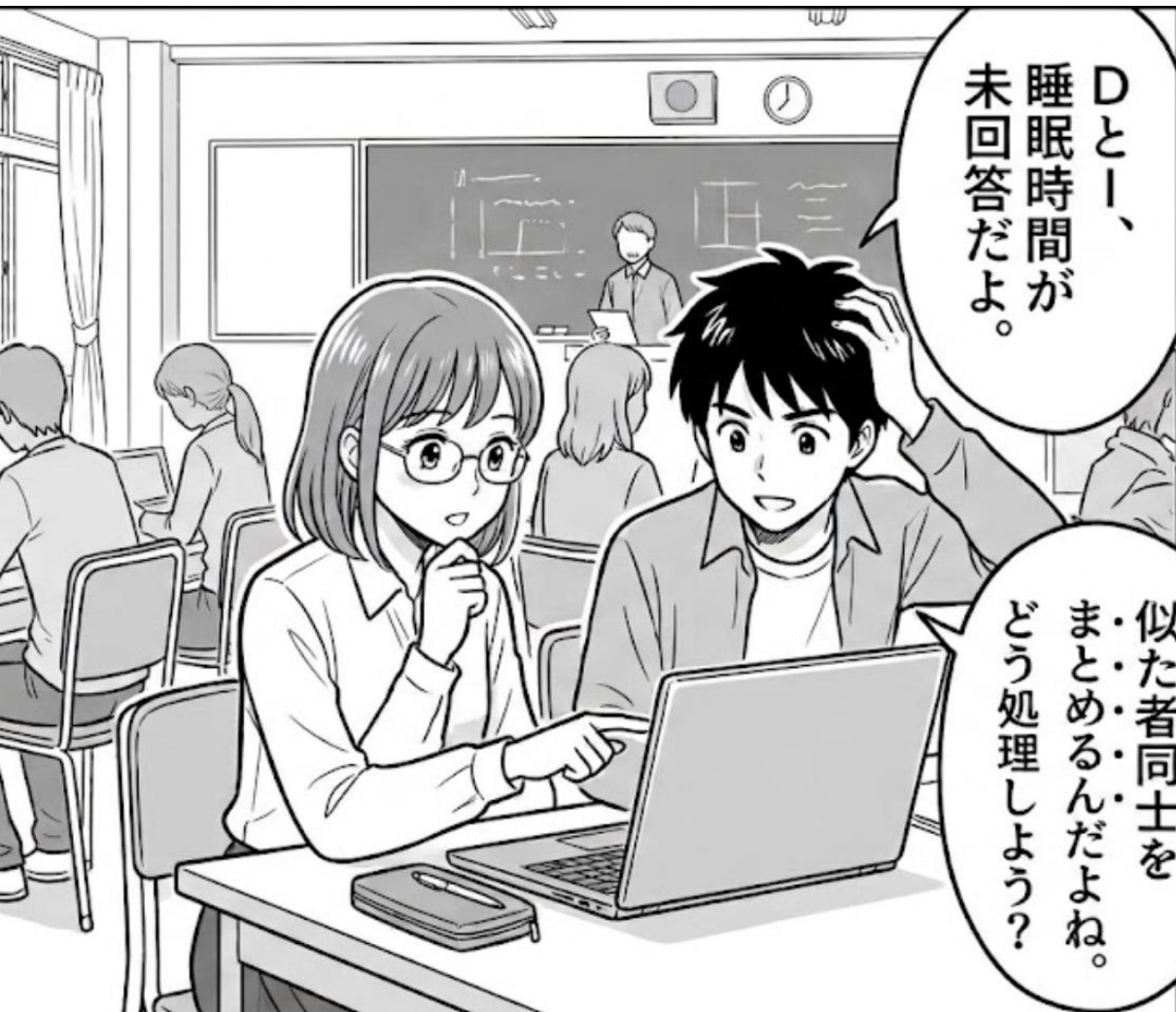
本来の値からのずれ
(記入ミス・測定誤差)

そのままでは
分析できない
ことがある

この前提を
知ることが、
以降の学習
すべての出発点

実データ (実際に観測・
収集した生のデータ)





生活習慣アンケート (12人分)

	A	B	C
▲	回答者	運動回数	睡眠時間
1			
2	A	5	7.5
3	B	1	5.0
4	C	6	8.0
5	D	0	未回答
6	E	3	6.5
7	F	7	7.0
8	G	2	5.5
9	H	0	4.5
10	I	4	未回答
11	J	6	7.5
12	K	1	6.0
12	L	3	6.5

欠損値への対処方針



アンケートの未回答（欠損）への2つの対処：行の除去か平均で補完

回答者	運動回数	睡眠時間
A	5	7.5
B	1	5.0
C	6	8.0
D	0	未回答
E	3	6.5
F	7	7.0
G	2	5.5
H	0	4.5
I	4	未回答
J	6	7.5
K	1	6.0
L	3	6.5

DとIの睡眠時間が欠損（2件）

方針1



方針2



方針1：欠損の行を除去

未回答のDとIを丸ごと外す

残り10人（ABCDEFGHIJKL）で進める

方針2：平均値で補完

他の10人の平均6.4時間をD・Iに入れる
 $(7.5+5.0+8.0+6.5+7.0+5.5+4.5+7.5+6.0+6.5) \div 10 = 6.4$

D → 6.4 / I → 6.4 で12人全員を残す

睡眠時間の欠損データ 2つの対処法

① 行を除去 (10人)

回答者	運動回数	睡眠時間
A	5	7.5
B	1	5.0
C	6	8.0
E	3	6.5
F	7	7.0
G	2	5.5
H	0	4.5
J	6	7.5
K	1	6.0
L	3	6.5

② 平均で補完 (12人)

回答者	運動回数	睡眠時間
A	5	7.5
B	1	5.0
C	6	8.0
D	0	6.4
E	3	6.5
F	7	7.0
G	2	5.5
H	0	4.5
I	4	6.4
J	6	7.5
K	1	6.0
L	3	6.5

欠損値への対処：Python での対処



生活習慣アンケート12人分、睡眠時間に2件の欠損 (D・I)。対処は「行の除去」か「平均で補完」の2通り

回答者	運動回数	睡眠時間
A	5	7.5
B	1	5.0
C	6	8.0
D	0	未回答
E	3	6.5
F	7	7.0
G	2	5.5
H	0	4.5
I	4	未回答
J	6	7.5
K	1	6.0
L	3	6.5

pandas では未回答は NaN として扱われる

① 行を除去

欠損のある D・I を丸ごと外す → 残り10人で進める

A B C E F G H J K L (10人)

```
df_dropped = df.dropna()
```

元の df は変更しない

② 平均で補完

D・I の睡眠時間に他10人の平均 6.4 を入れる → 12人全員を残す

平均 = $(7.5 + 5.0 + 8.0 + 6.5 + 7.0 + 5.5 + 4.5 + 4.5 + 7.5 + 6.0 + 6.5) \div 10 = 6.4$

```
df.isnull().sum()  
df["睡眠時間"] = df["睡眠時間"].fillna(df["睡眠時間"].mean())
```

D・I に 6.4 が入る

データ分析の主な3つの型

分類

グループごとの
違いを見つける

メールの
スパム判定



顧客の
解約予測



回帰

過去の数値から
先を予測する

住宅価格の予測



売上予測



クラスタリング

似た者どうしを
自動でまとめる

顧客セグメンテーション



グループ A グループ B グループ C

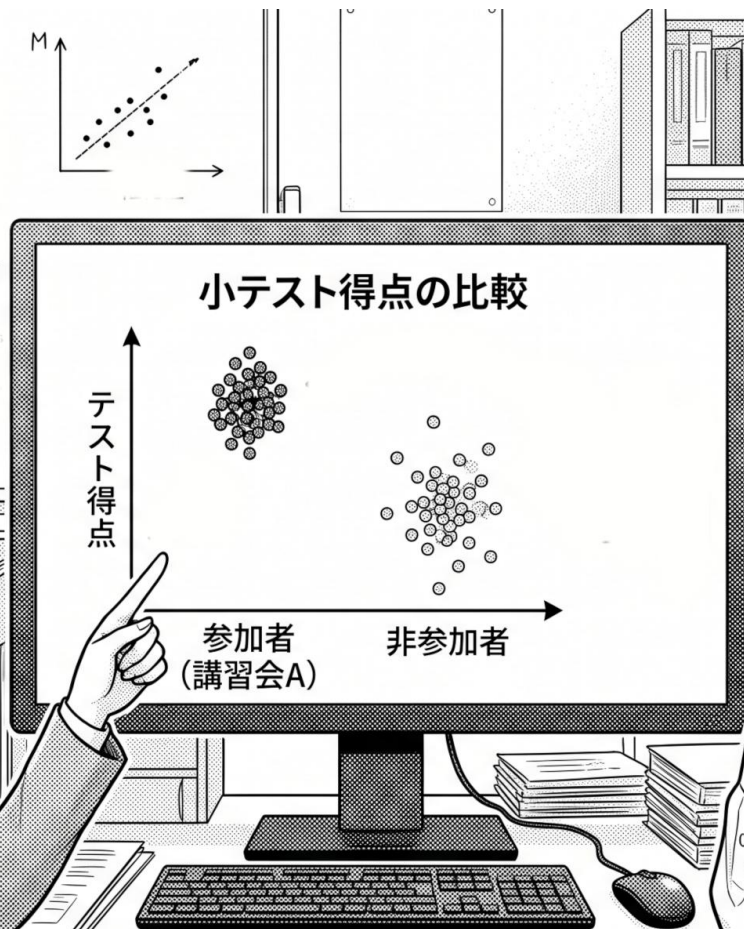
レコメンデーション



分析の型は
いろいろ



講習会Aの参加者と非参加者で、
小テストの点を比べたのか。



はい。参加者の方が高得点。
次回の募集に活かしましょう。

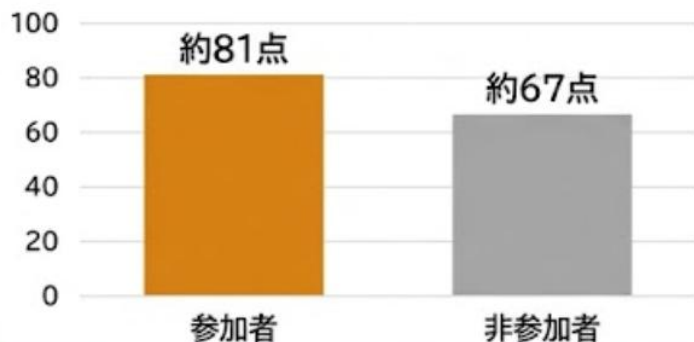
分類（グループごとの違いを見つける）



講習会Aの参加者と非参加者で小テストの点を比較し、次回募集に生かす

観察するデータ(小テストの点数)

グループ	回答者	点数	グループ	回答者	点数
参加者	A	85	非参加者	G	70
	B	78		H	65
	C	92		I	72
	D	60		J	55
	E	88		K	68
	F	81		L	74



傾向

参加者の方が平均点が高い(参加者~81点 / 非参加者~67点)。講習会には効果がありそう

問題点

参加しても伸びない人がいる(例:参加者Dは60点)。効果には個人差がある

改善の余地

次回募集に生かす(例:点数が伸び悩む層向けの補習枠を設ける・非参加者に効果を示して参加を促す)

ここでは全体の傾向を眺めている。本来の『分類』は、自動での分類（2つ以上のグループの特徴の違いを手がかりに、新しいデータがどちらに属するかを見分ける）である

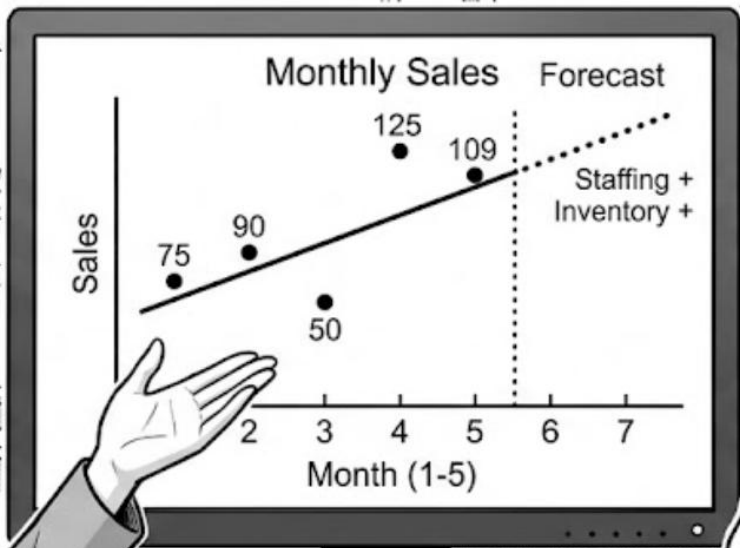
はい。
 上昇傾向なので、
 来月は人員と仕入れを
 増やす提案です。

$$= x^2 + 2t + C \quad y = n - ovt$$

$$\frac{d^2}{dt^2} = \frac{1}{\partial \pi} \frac{\partial v}{\partial x} = \frac{n}{2} = 2/x + \frac{1}{at} + \left(\frac{y}{x} + \frac{x}{a}\right)$$

$$= r^2 \quad f = r(1a - 1/t)^2$$

$$\frac{kx^2}{2} + \frac{dv}{dt} = \frac{6}{-1}$$



5ヶ月の売上データから、
 回帰で予測したのか。

回帰（過去の数値から先を予測する）



まず実際のデータを読み、そこから傾向・問題点・改善の余地に気づく

観察するデータ(売上・アンケートなど)

月	数値
1月	75
2月	90
3月	50
4月	125
5月	109



傾向

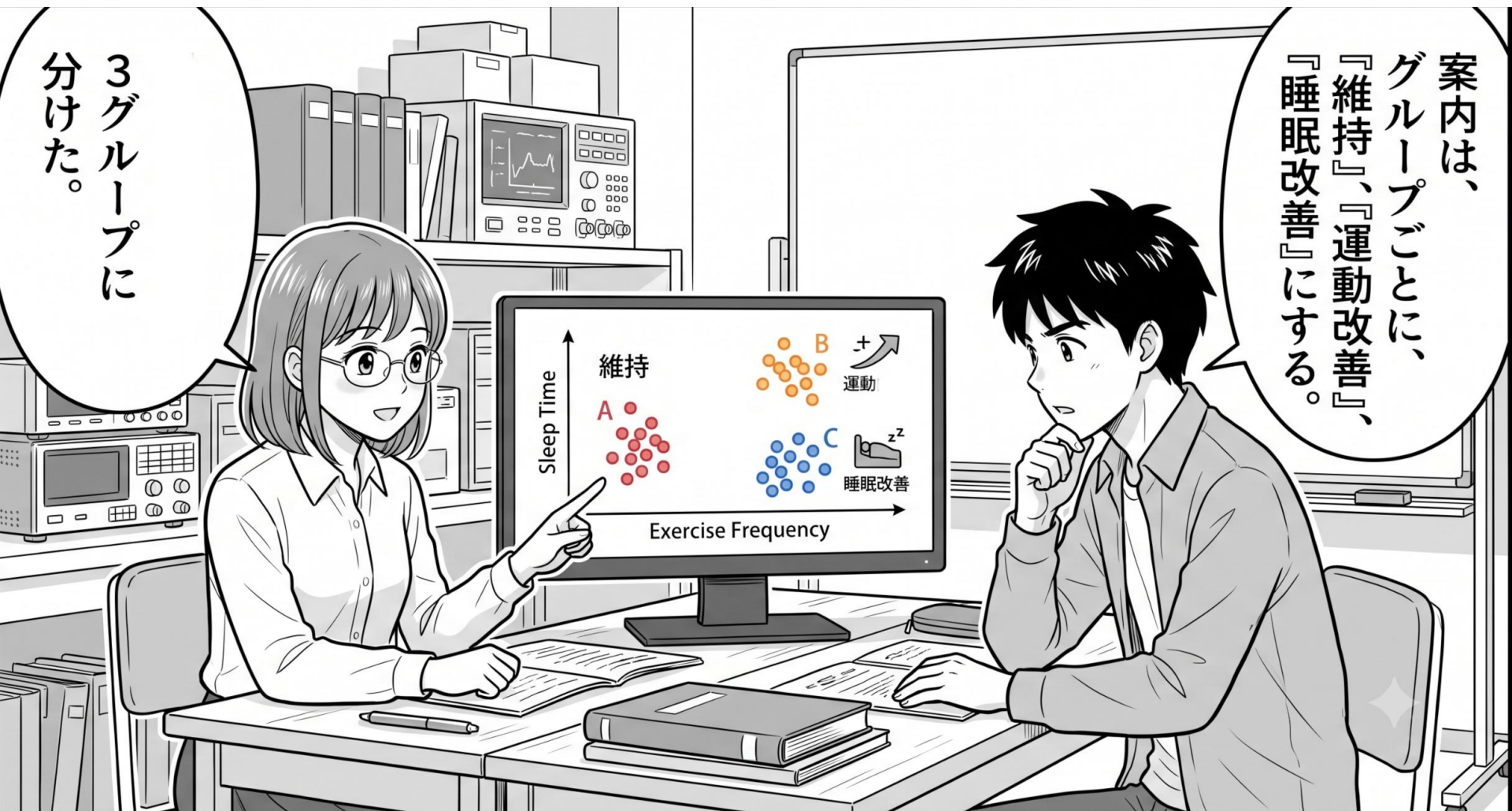
全体がどちらに向かっているかが見える
(例:売上は上昇傾向にある)

問題点

見落としていた弱点に気づく
(例:特定の月だけ落ち込んでいる)

改善の余地

打ち手を計画できる
(例:上昇傾向なら人員と仕入れを増やし、
売上予測から適切な仕入れと人員を計画する)



クラスタリング：似た者同士を自動でまとめる



クラスタリング：12人を3グループに分けて対応を考える

補完済みデータ(12人分)

回答者	運動回数	睡眠時間
A	5	7.5
B	1	5.0
C	6	8.0
D	0	6.4
E	3	6.5
F	7	7.0
G	2	5.5
H	0	4.5
I	4	6.4
J	6	7.5
K	1	6.0
L	3	6.5

維持グループ

運動も睡眠も十分(例:A・C・E・F・I・J・L)。現状維持を案内

運動改善グループ

睡眠は取れているが運動が少ない(例:B・D・H・K)。運動習慣づくりを提案

睡眠改善グループ

睡眠時間が短い(例:B・G・H)。睡眠の質・時間の改善を案内

データを用いて課題を解決する



データ分析は"課題の発見"から始まり、必ず"次の行動"につながる

課題を見つける

機械に分析させる

分かること
(根拠)

次の行動に
活かす

	課題	分析	分かること	行動
分類 (違いの発見)	講習会Aの効果を 知りたい	受講者と非受講者の 回答傾向を機械に 見つけさせる	両者の回答傾向 の違い	次の募集案内の 改善に活かす
回帰 (将来の予測)	来年の売上を 見通したい	過去2年の月別売上 から来年の推移を 予測させる	来年の売上の 見込み	仕入れや人員の 計画に活かす
クラスタリング (自動グループ分け)	回答者の特徴を つかみたい	アンケート回答者を 似た者どうしに 分けさせる	似た者どうしの グループ	グループごとの 対応・案内に活かす

12-2. データ分析手法の基礎

数値の表を眺めるだけでは傾向はつかみにくい。
図にすると目で直感的に把握できる。

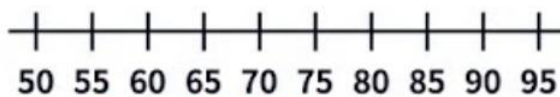
表（数値のまま）

参加者	非参加者
85	70
78	65
92	72
60	55
88	68
81	74

傾向の違いが
読み取りにくい

点数の分布

可視化



点数

- = 参加者
- = 非参加者



分かること

参加者は
高い側、
非参加者は
低い側に分布

ただし中間には
重なりもある

可視化の実現：Python による実現

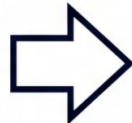


Pythonでデータをグラフに変換すると、点数の傾向が目で確認できる。

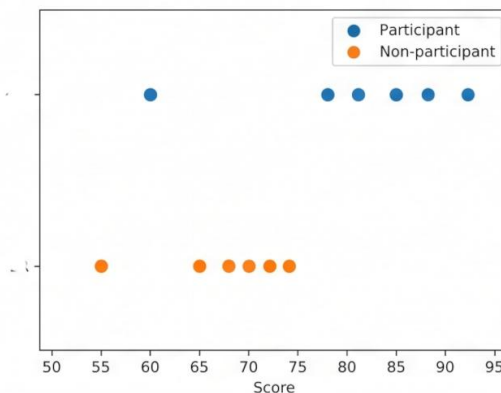
1. コードを書く

```
import matplotlib.pyplot as plt
plt.scatter(参加者["点数"], [1]*6,
            label="Participant")
plt.scatter(非参加者["点数"], [0]*6,
            label="Non-participant")
plt.xlabel("Score")
plt.legend()
plt.show()
```

実行



2. グラフが出る



3. 傾向が読める

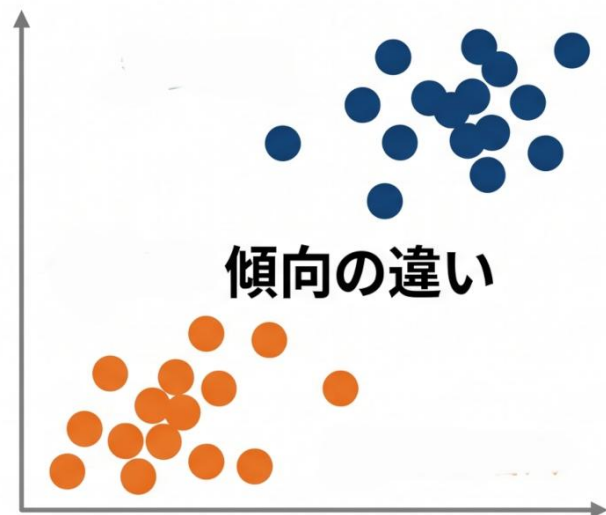
**参加者は高い側、
非参加者は低い
側に分布**

可視化の限界



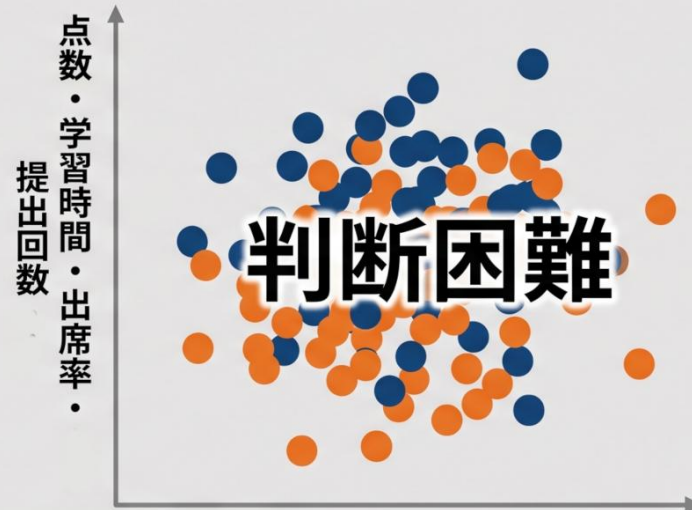
可視化は万能ではない — 目で見えて判断できるのは項目1~2個までという限界がある

項目が1~2個なら目で判断できる



項目が増えると...

項目が増えると目では判断できない



項目が1個（点数だけ） = 目で見えて判断できる

人間が目では総合判断できるのは項目1~2個まで。
項目が増えると可視化だけでは傾向をつかめない

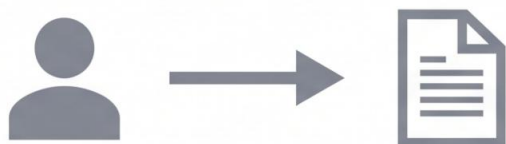
→ パターン・ルールを見つける作業そのものをコンピュータに任せる = 機械学習へ

機械学習のメリット



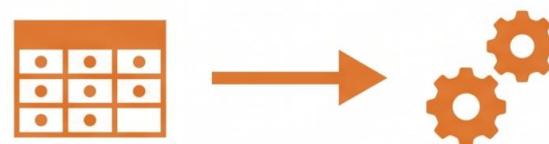
従来のプログラム

人間がルールを書いて教える



機械学習

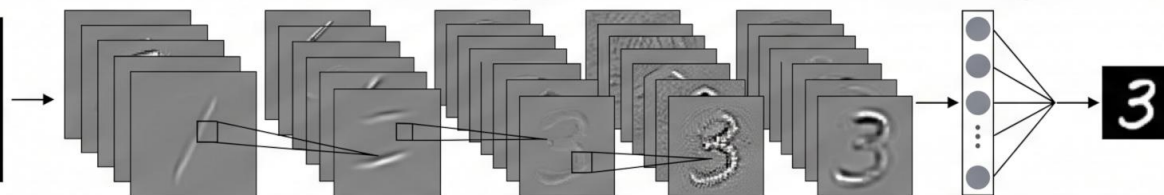
データを与えコンピューターがパターン・ルールを見つける



入力画像



入力画像



出力結果

ルールを人間が
書かなくてよい

人が気づきにくい
パターンも発見

学習した傾向を
未知データにも適用

判定・予測・分類を
自動化できる



機械学習の主なタスク

分類

回帰

クラスタリング

その他

学習した傾向を未知のデータにも適用できる (汎化)

① 機械学習の流れ

データ → 学習(fit) → モデル → 判定・予測(predict) → 結果

② "学習"と"判定・予測"

fit でコンピュータは分類基準などのパターンやルールを見つける。
predict で使用。

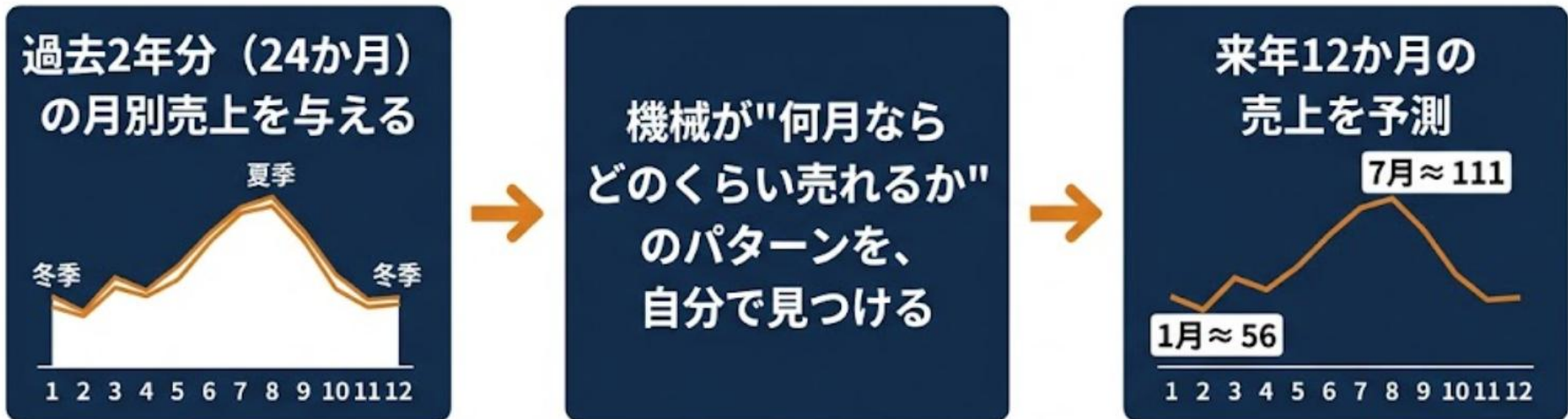
③ 使用するPythonのライブラリ

pandas: データ準備・管理
scikit-learn: 学習、判定・予測

その他、Pythonには多数の機械学習用ライブラリがある

過去のデータから来年を予測する (回帰)

fit で学習し、predict で答えさせる — この使い方は変わらない



"夏に売れる"と人間が教えたのではない。

コンピューターが24個の数値から、パターンを見つける

"どちらのグループか"を当てるのではなく、今回は"数値そのもの"を予測する (=回帰)

結果はデータの質と量に左右される

学習データに偏りがあると判定を誤る

過学習：訓練データに合いすぎ未知データで外す

**精度は100%ではなく必ず検証が必要
（「未知のデータ」を用いて検証）**

機械学習の実現：Python の利用



pandas : データを準備する

scikit-learn : 学習と判定を行う



※学習に使ったデータでの一致は12/12。丸暗記でも当たるため、本当の力は新しいデータで確かめる=これが"検証"

**例：小テストの点数から、参加者・非参加者を"見分けられるか"を調べる
→ "傾向"を、見分けにくい人 (例：参加者Dは60点) を映す**

コンピューターの結果を、図にして確かめる

結果は、こう返ってくる

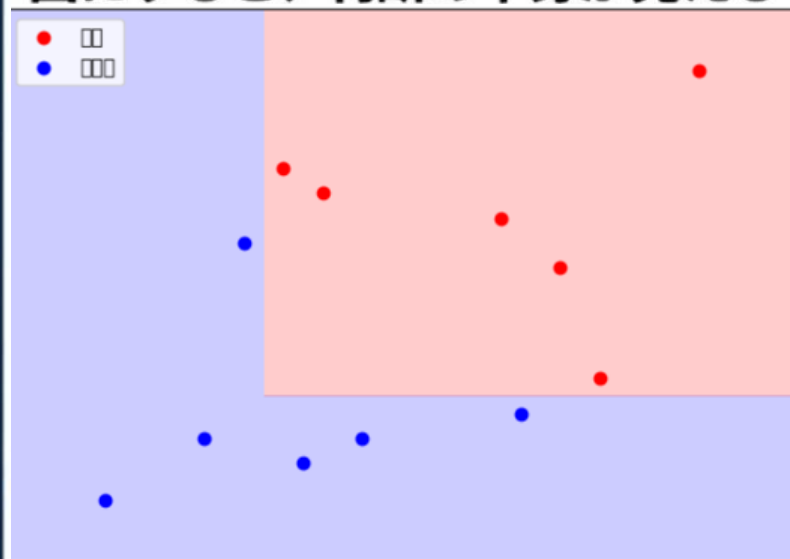
機械学習による分類の場合
分類結果は A
分類結果は B

このままでは、コンピューターが
"何をどう判断したのか""信用してよいの"
人間には分からない

図にする



図にすると、判断の中身が見える



= その結論が妥当そうか確認できる

これは、分析の"前"にデータを眺める図ではなく、分析の"後"に、出てきた結論を人間が確かめるための図

予測が信用できそうか、図で確かめる

予測は、こう返ってくる

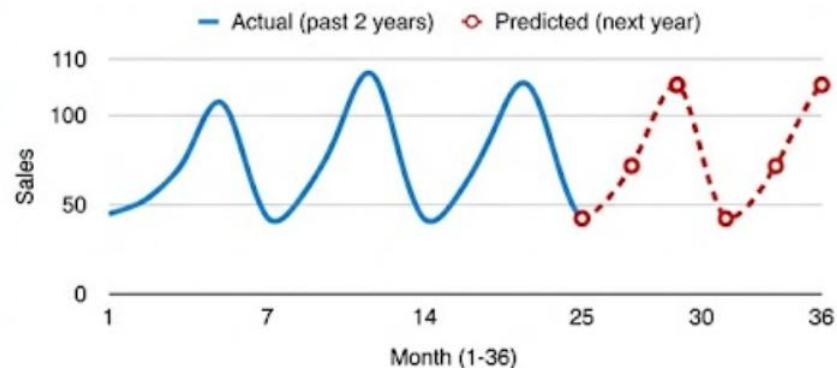
次年の12ヶ月予測数値値
56, 59, 66, 73, 81, 96, 111,
106, 89, 76, 63, 58

数値の表を見ているだけでは、
この予測を信用してよいか分からない

図にする



1本の時間軸に並べると、確認も気づきも得られる



- 赤い予測が、過去の山と谷の形をそのまま引き継いでいる
= 季節のパターンを学習し、来年も同じ形と結論づけた、と分かる
- △ 毎年少しずつ伸びている分は、予測に反映されていないのでは、という限界にも気づける

"コンピューターに結論を出させる → 図にして人間が確かめる" はセット。図で確かめずにそのまま信用しない。

12-3. プログラミングによる分析 と対話型AIによる分析

自分でコードを組み立てて実行する

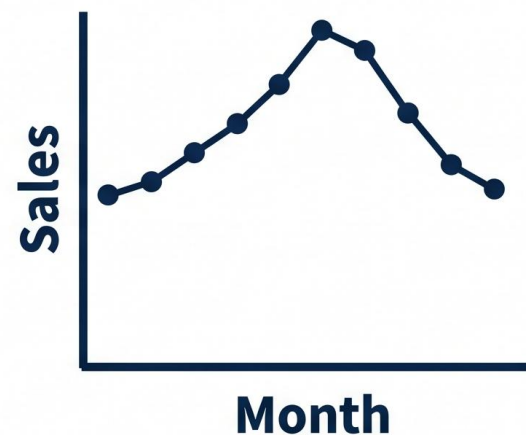
自分でコードを書く

対話型AI（ChatGPTなど）の助けを借りることもある

Pythonコード

```
import matplotlib.pyplot as plt  
売上 = [56, 59, 66, 73, 81, 96, 111,  
        106, 89, 76, 63, 58]  
plt.plot(range(1,13), 売上, marker="o")  
plt.xlabel("Month"); plt.ylabel("Sales")
```

実行



対話型AIによる分析

自然言語で指示 (プロンプト)

人間の指示



対話型AIによる実行



グラフ

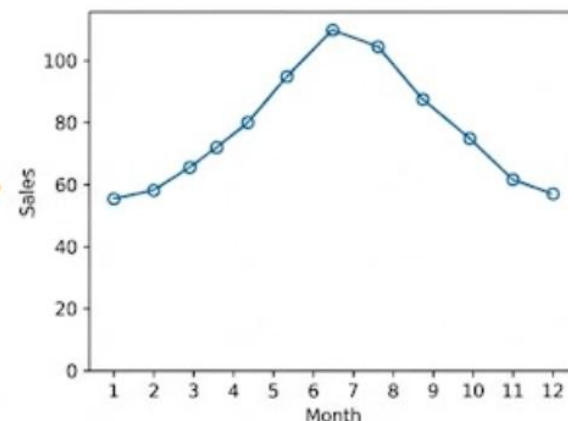
月別売上

[55, 58, 65, ..., 57]
の折れ線グラフを作って



対話型AI

数秒~数十秒で返答



対話型AIが自動でコード生成、実行を行っている場合がある

自分で組み立てる vs 対話型AIに任せる — 5つの観点で比べる

観点	自分でコードを組み立てる	対話型AIに任せる
手順	自分で記述 (AIの補完も併用)	自然言語で指示する
速さ	書けるまで慣れが必要	素早く結果を得られる
再現性	高い (同じコード=同じ結果)	同じ指示でも毎回変わりうる
理解の深さ	一行ずつ理解しながら進める	中身を理解しないとブラックボックス化しやすい
検証・応用	自力で誤りを直し、拡張できる	誤りの修正には結局プログラミングの知識が役立つ

両者は対立ではなく、補完し合う。
 分けるのは"AIを使うかどうか"ではなく、"結果をどうコントロールするか"

AIに任せれば速い。ただし出力の正しさを検証し、目的に合わせて直す・広げるには基礎が不可欠 — "コンピューターの出力は、人間が確認・検証してから使う"

演習



PythonOnline.dev の使い方 — ブラウザだけでPythonを実行する

自分のPCへのインストール（環境構築）は不要。ブラウザだけで始めることができる



URL: <https://pythononline.dev/>

pythononline.dev の使い方



- ブラウザ上で Python を記述・実行できるオンライン実行環境
- pandas・matplotlib・scikit-learn はインポートで読み込み

【使用手順】

- pythononline.dev のURL

<https://pythononline.dev/>

- **コードセルの中身をすべて削除。各演習の「コード」をコピーして貼り付け**
- 「**Run All**」をクリック
- セルの下に print の結果やグラフ（画像）が表示されることを確認

演習 1 . 回帰による将来予測 — 予測を図で確かめる



- OneCompiler を使用

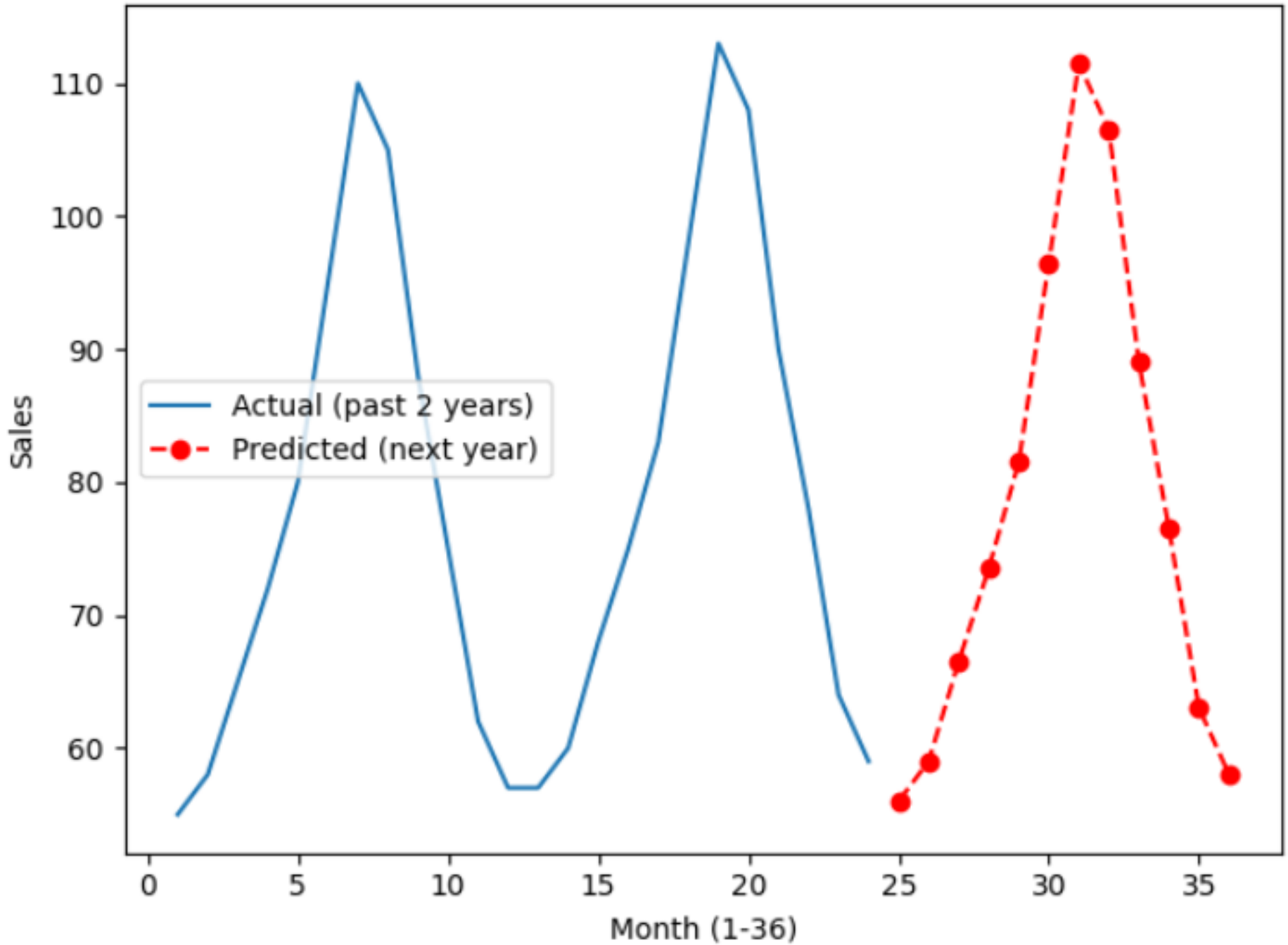
<https://pythononline.dev/>

- **前のコードは削除。下のコードを張り付けて、「Run All」をクリック**

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor
```

```
df = pd.DataFrame({
    "月": list(range(1, 13)) * 2,
    "売上": [55, 58, 65, 72, 80, 95, 110, 105, 88, 75, 62, 57,
            57, 60, 68, 75, 83, 98, 113, 108, 90, 78, 64, 59],
})
model = DecisionTreeRegressor(random_state=0)
model.fit(df[["月"]], df["売上"])
予測 = model.predict(pd.DataFrame({"月": list(range(1, 13))}))
```

```
plt.plot(range(1, 25), df["売上"], label="Actual (past 2 years)") # 実績 (1~24か月目)
plt.plot(range(25, 37), 予測, "r--o", label="Predicted (next year)") # 予測 (25~36か月目)
plt.xlabel("Month (1-36)")
plt.ylabel("Sales")
plt.legend()
plt.show()
```



演習 2. データの読み込みと欠損の補完



- OneCompiler を使用

<https://pythononline.dev/>

- 前のコードは削除。下のコードを張り付けて、「Run All」をクリック

```
import pandas as pd
```

```
df = pd.DataFrame({
    "回答者": ["A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L"],
    "運動回数": [5, 1, 6, 0, 3, 7, 2, 0, 4, 6, 1, 3],
    "睡眠時間": [7.5, 5.0, 8.0, None, 6.5, 7.0, 5.5, 4.5, None, 7.5, 6.0, 6.5],
})
```

```
print("補完前の未回答数:¥n", df.isnull().sum())
df["睡眠時間"] = df["睡眠時間"].fillna(df["睡眠時間"].mean()) # 平均6.4時間で埋まる
```

補完前の未回答数:

回答者 0

運動回数 0

睡眠時間 2

dtype: int64

	回答者	運動回数	睡眠時間
0	A	5	7.5
1	B	1	5.0
2	C	6	8.0
3	D	0	6.4
4	E	3	6.5
5	F	7	7.0
6	G	2	5.5
7	H	0	4.5
8	I	4	6.4
9	J	6	7.5
10	K	1	6.0
11	L	3	6.5

演習 3. 分類による判定、機械学習の結果の可視化



- OneCompiler を使用

<https://pythononline.dev/>

- 前のコードは削除。下のコードを張り付けて、「Run All」をクリック

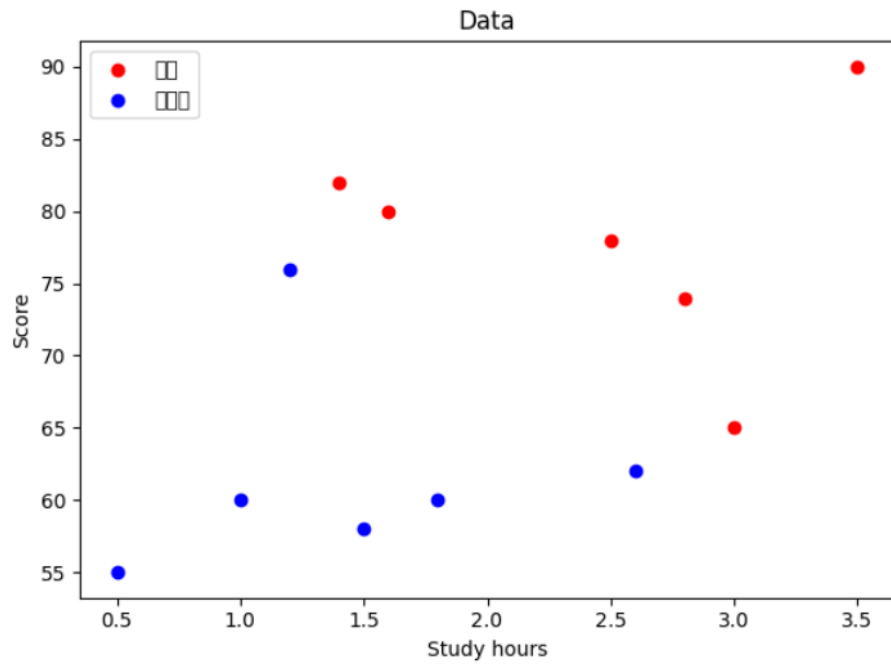
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier

# 1. データの用意
df = pd.DataFrame({
    "学習時間": [2.5, 3.0, 1.6, 3.5, 2.8, 1.4, 1.0, 0.5, 1.5, 2.6, 1.2, 1.8],
    "テスト点数": [78, 65, 80, 90, 74, 82, 60, 55, 58, 62, 76, 60],
    "講習会A": ["受講"]*6 + ["未受講"]*6,
})

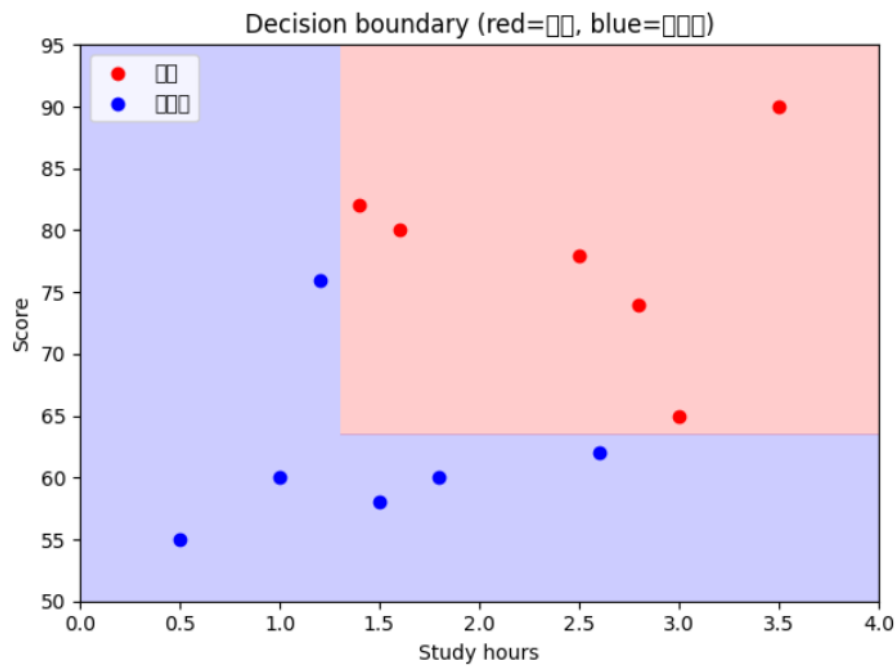
# 2. 実データの散布図: 受講/未受講を色分けして描く
for 判定, 色 in [("受講", "red"), ("未受講", "blue")]:
    部分 = df[df["講習会A"] == 判定]
    plt.scatter(部分["学習時間"], 部分["テスト点数"], color=色, label=判定)
plt.xlabel("Study hours")
plt.ylabel("Score")
plt.title("Data")
plt.legend()
plt.show()

# 3. 学習 (fit)
model = DecisionTreeClassifier(random_state=0)
model.fit(df[["学習時間", "テスト点数"]], df["講習会A"])

# 4. 決定境界の可視化: 平面全体を格子点で埋め、各点の判定を背景色で塗り分ける
x_min, x_max = 0, 4
y_min, y_max = 50, 95
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 300),
                     np.linspace(y_min, y_max, 300))
格子 = pd.DataFrame({"学習時間": xx.ravel(), "テスト点数": yy.ravel()})
領域 = model.predict(格子)
領域の色 = np.where(領域 == "受講", 1, 0).reshape(xx.shape)
plt.contourf(xx, yy, 領域の色, levels=1, colors=["blue", "red"], alpha=0.2)
for 判定, 色 in [("受講", "red"), ("未受講", "blue")]:
    部分 = df[df["講習会A"] == 判定]
    plt.scatter(部分["学習時間"], 部分["テスト点数"], color=色, label=判定)
plt.xlabel("Study hours")
plt.ylabel("Score")
plt.title("Decision boundary (red=受講, blue=未受講)")
plt.legend()
plt.show()
```



[Download image](#)



演習 4 . クラスタリング

- OneCompiler を使用



<https://pythononline.dev/>

- **前のコードは削除。下のコードを張り付けて、「Run All」をクリック**

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# 1. データの用意（第1章と同じ、未回答を含む実データ）
df = pd.DataFrame({
    "回答者": ["A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L"],
    "運動回数": [5, 1, 6, 0, 3, 7, 2, 0, 4, 6, 1, 3],
    "睡眠時間": [7.5, 5.0, 8.0, None, 6.5, 7.0, 5.5, 4.5, None, 7.5, 6.0, 6.5],
})

# 2. 前処理: 未回答を平均値で補完する
print("補完前の未回答数:¥n", df.isnull().sum())
df["睡眠時間"] = df["睡眠時間"].fillna(df["睡眠時間"].mean()) # 平均6.4時間で埋まる

# 3. 機械学習: 似た回答の人同士を、自動で2グループに分けさせる
model = KMeans(n_clusters=2, random_state=0)
df["グループ"] = model.fit_predict(df[["運動回数", "睡眠時間"]])
print(df)
print("¥nグループごとの平均:¥n", df.groupby("グループ")[["運動回数", "睡眠時間"]].mean())

# 4. 結果の可視化: コンピューターの分けたグループを色で確認する
for g, 色 in [(0, "blue"), (1, "red")]:
    部分 = df[df["グループ"] == g]
    plt.scatter(部分["運動回数"], 部分["睡眠時間"], color=色, label=f"Group {g}")
plt.xlabel("Exercise per week")
plt.ylabel("Sleep hours")
plt.legend()
plt.show()
```

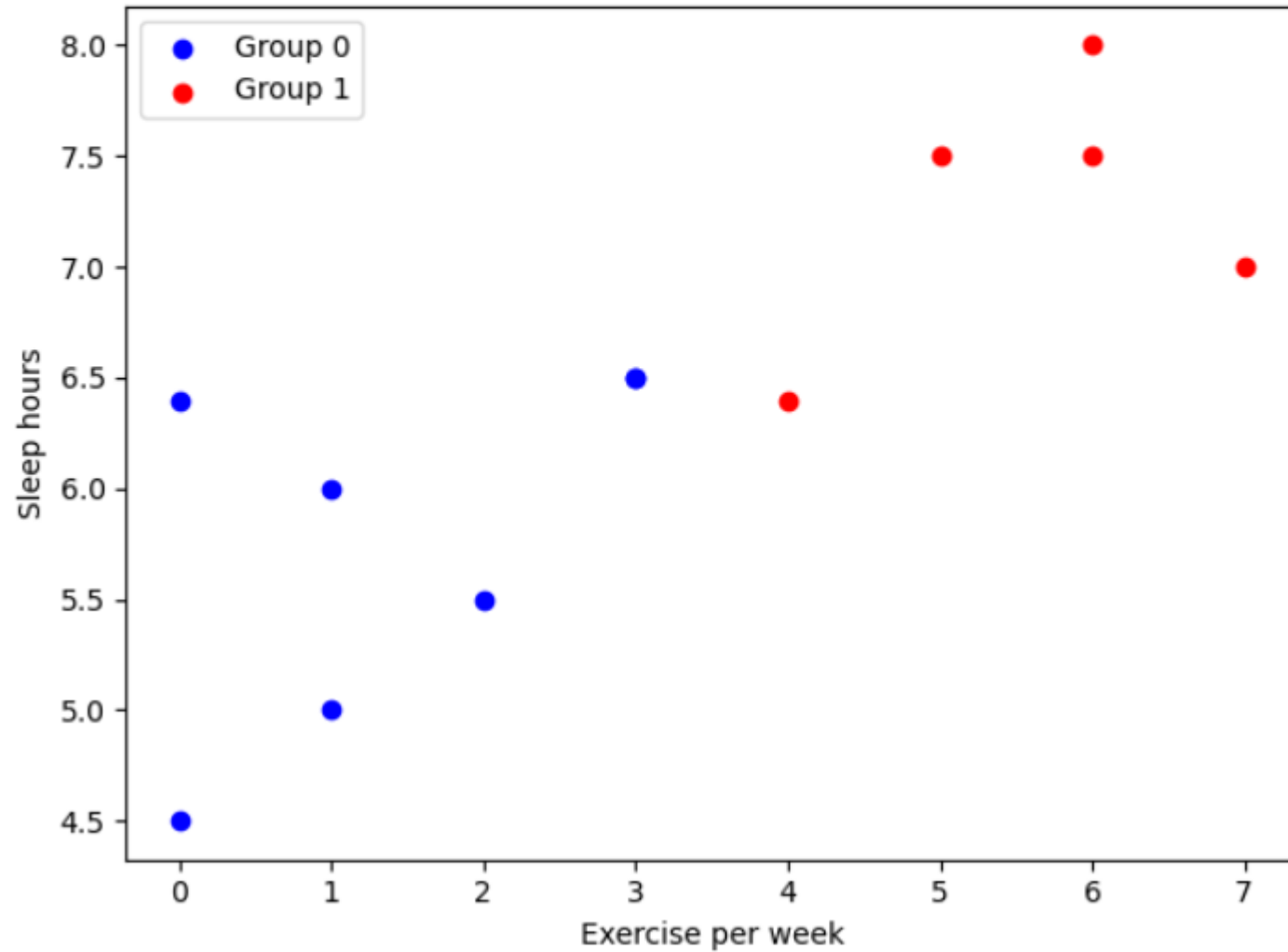
グループごとの平均:

運動回数 睡眠時間

グループ

0	1.428571	5.771429
1	5.600000	7.280000

↓ Download image



演習 5 . 回帰による将来予測



- OneCompiler を使用

<https://pythononline.dev/>

- 前のコードは削除。下のコードを張り付けて、「Run All」をクリック

```
import matplotlib.pyplot as plt
```

```
売上 = [55, 58, 65, 72, 80, 95, 110, 105, 88, 75, 62, 57]
```

```
plt.plot(range(1, 13), 売上, marker="o") # plot で折れ線グラフを描く
```

```
plt.xlabel("Month")
```

```
plt.ylabel("Sales")
```

```
plt.show()
```

対話型AI（例：Microsoft Copilot, <https://portal.office.com>

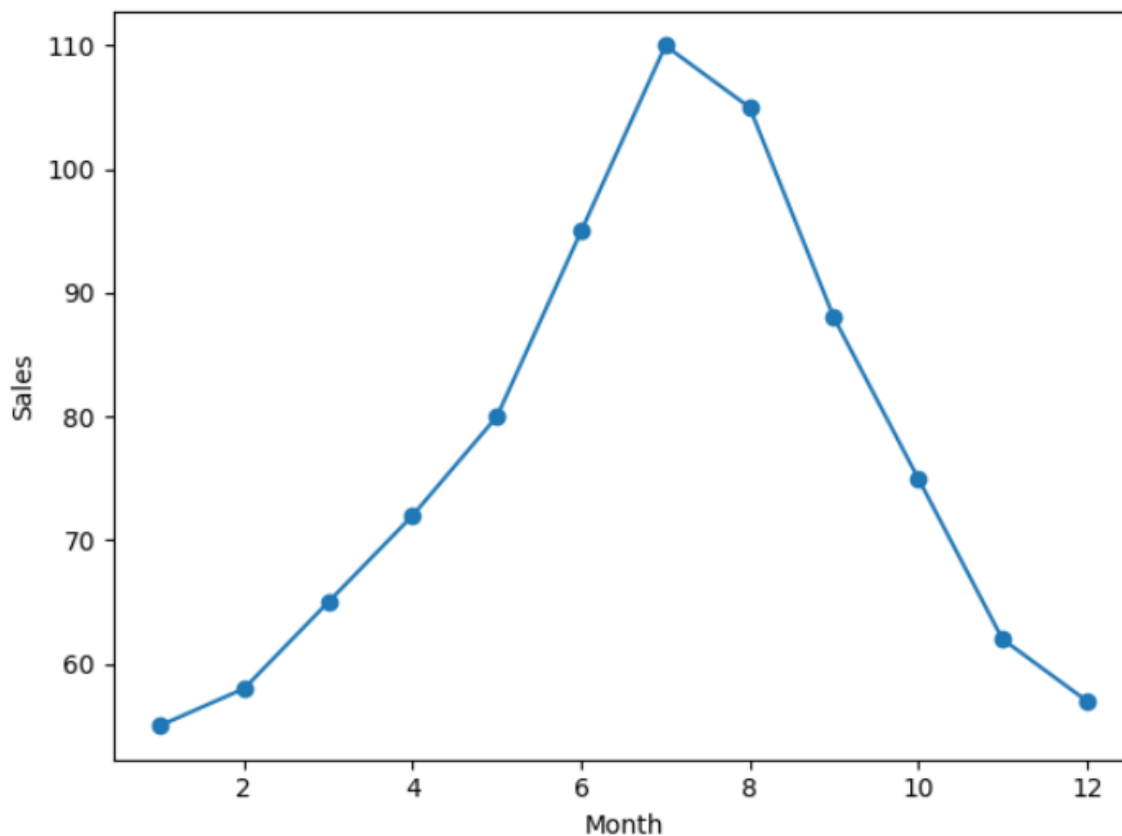
にアクセス。大学のアカウントでサインイン）への指示（プロンプト）

```
import matplotlib.pyplot as plt
```

```
売上 = [55, 58, 65, 72, 80, 95, 110, 105, 88, 75, 62, 57]  
plt.plot(range(1, 13), 売上, marker="o") # plot で折れ線グラフを描く  
plt.xlabel("Month")  
plt.ylabel("Sales")  
plt.show()
```

Click **Run** or press **shift + ENTER** to run code

[Download image](#)





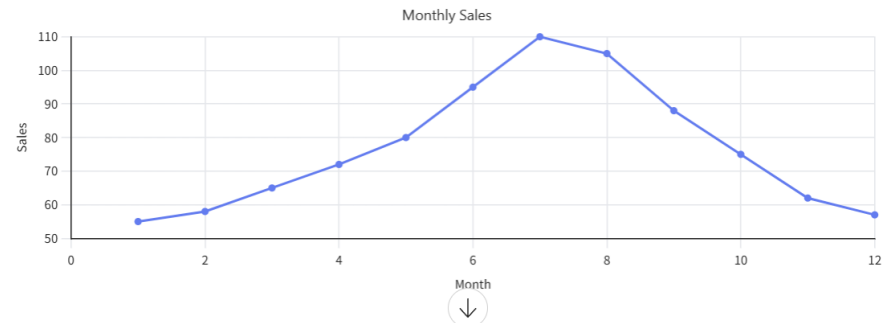
Copilot

コーディングと実行

月別売上 [55, 58, 65, 72, 80, 95, 110, 105, 88, 75, 62, 57] の折れ線グラフを表示してください。

```
import matplotlib.pyplot as plt
months=list(range(1,13))
sales=[55,58,65,72,80,95,110,105,88,75,62,57]
plt.figure(figsize=(7,4))
plt.plot(months,sales,marker='o')
plt.title('Monthly Sales')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.xticks(months)
plt.grid(True)
plt.tight_layout()
file='monthly_sales_line_chart.png'
plt.savefig(file)
plt.close()
print(file)
```

月別売上の折れ線グラフです。



売上は1月の55から増加し、7月に最大110を記録した後、年末に向けて減少しています。ピークは7月、最小は1月です。

展開して、思考過程（この場合にはAIが作成したコード）を表示

Copilot

コーディングと実行

月別売上の折れ線グラフです。



売上は1月の55から増加し、7月に最大110を記録した後、年末に向けて減少しています。ピークは7月、最小は1月です。