第 4 章

機械学習（教師なし学習）

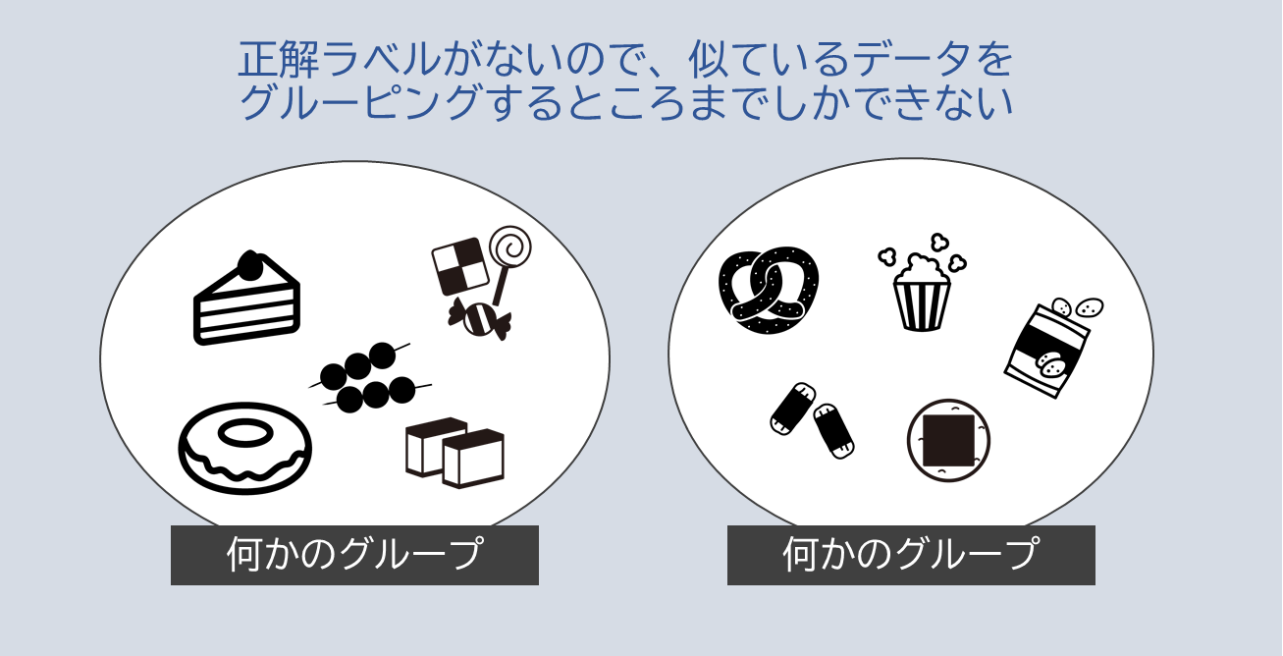
教師あり学習と違って、教師なし学習にはデータの中に正解がありません。正解がないデータはどのような課題解決をしていくのでしょうか。この章では、実践例とともに、手法の種類とそのアルゴリズムについて学んでいきます。

この章のゴール

教師あり学習の基本的な手法と実践例を理解する

教師なし学習とは

教師なし機械学習では過去のデータを何かしらの観点に基づいて似ているデータ同士を分類します。教師がない（正解ラベルがない）状態での分類となりますので、教師なし学習ではこのグルーピングそのものが学習となり、主なタスクとなります。



教師なし学習の特徴は、分類は行えますが、それぞれのグループが持つ意味を明らかにすることはできません。そのため、機械が分類した結果に対して、人が解釈をつける必要があります。

人の目では判断できない大量のデータやグループに対しても機械は分類を行うことができますので、潜在的なクラスターの発見や、異常なデータの発見に役立てることができます。

教師なし学習の手法と活用例（具体的ケースの提示・手法の説明）

教師なし学習の代表的なアルゴリズム

教師なし学習の具体的な手法として、クラスター分析、主成分分析、自己組織化マップ（SOM）などが挙げられます。

主成分分析

多くのデータの中から、ある一定の法則を見つけ出す分析方法です。

たくさんの量的な説明変数の中から、より少ない指標あるいは合成変数と呼ばれる複数の変数を組み合わせたものに要約する手法です。

この要約は「次元の縮約」という表現で呼ばれることもあります。

クラスター分析

クラスター（cluster）とは、英語で「房」「群れ」「かたまり」を意味し、似たものが集まっている状態をいいます。クラスター分析は、大量に集められたデータから、特徴が近い（似ている／距離が近い）データを集めて集団に分ける分析手法です。そして、特徴が近いデータが集まった集団を「クラスター」と呼び、データからいくつかの集団を作ることを「クラスタリング」と呼びます。

自己組織化マップ（SOM）

SOM は、ニューラルネットワークと言われる AI 技術で使用される数理モデルの一種です。煩雑且つ膨大な情報を、人間が瞬時に理解することができるように、傾向や相関関係を自動的に判別し、視覚的に理解できるようにすることができる可視化の手法です。

SOM では、事前の予備知識（正解データ）がなくても、瞬時に高次元のデータをクラスタリングすることができます。この手法は、様々な入力データを与え続けることで、だんだん類似度の近いものが集まり、それらを視覚的に認識することができます。

大量のデータを効率よく圧縮できることが特徴のモデルです。

クラスター分析

「似たもの同士」を集めるクラスター分析

クラスター分析とは、様々な性質をもつデータが大量に集まった中から、特徴が似ているデータを集めていくつかのグループに分類し、データの特性や共通項を把握したり、大量のデータを扱いやすくする分析手法です。クラスター分析は、機械学習の教師なし学習における代表的な手法で、ビジネスの領域においては特に多くの顧客を分類してマーケティング施策を検討する際によく使われています。

Case

世界中の「主食」「間食」とその原材料を用いてクラスタリングし、地図上にマッピングしてみよう

美緒さんと翔真くんは、地理の授業で色々な国の主食を調べてまとめることになりました。
知らない国のごはんを見るのは新鮮で、とても楽しく進めています。

調べていくうちに、美緒さんは「なぜこんなにもたくさんの種類の主食があるんだろう？」
と考えるようになりました。

「地理的に見ると、高温多湿だとお米が多くて、乾燥した地域は小麦なんだ」
「各国の主食は現地の気候に適したものが選ばれているのでは？」

美緒さんはふと、学校で習った機械学習の授業のことを思い出しました。

「そういえば、似た条件のものをグルーピングする「クラスター分析」について 習ったな。
私にもできるかな･･･？」

グループ、セグメントとの違い

「クラスター」とよく比較される分類方法として、「グループ」や「セグメント」があります。

「グループ」や「セグメント」はあらかじめ定められた定義のもとに対象を分類するのに対し、データの特徴から統計的な処理によって分類する「クラスター」は明確に異なります。

	グループ	セグメント	クラスター
よく活用される社会	マスコミを含めて日常語として通用	マーケティング （生物や医学でも）	社会科学全般、考古学、生物学
実在概念なのか、 操作的な概念なのか	調査をしようがしまいが、グループ自体は実在していると想定する	分析者の都合で市場を分割する。 例えば性別や年齢別のクロス集計など	統計学的な基準をデータに適用して全体を分割する操作的な概念

参考文献
Smith,W.R (1956) Product differentiation and market segmentation as alternative marketing strategies. Journal of Marketing, 21. No.1, 3-8

階層的クラスタリングと非階層的クラスタリング

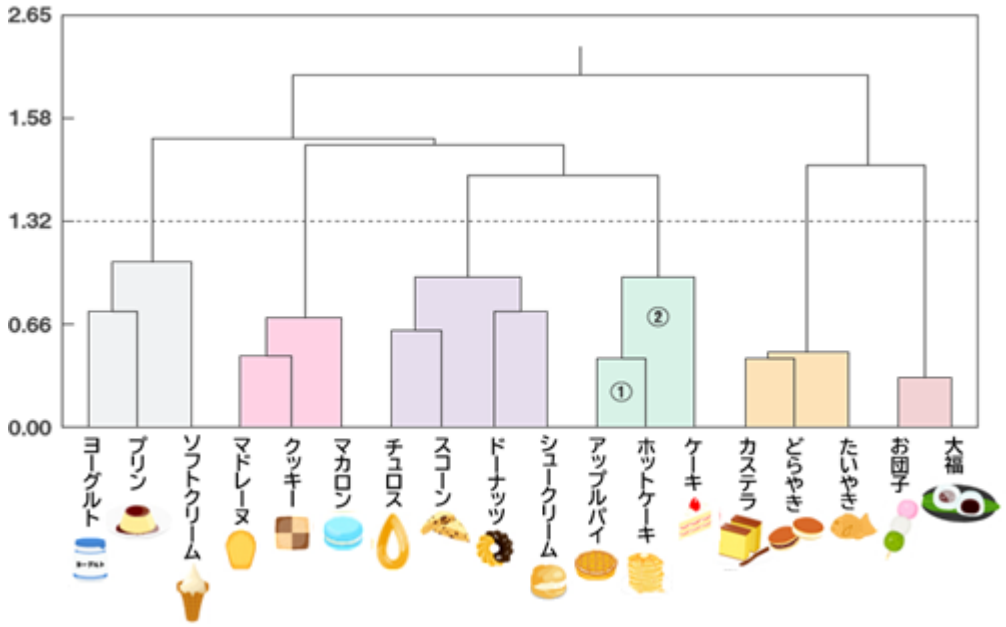
クラスタリングの方法は、大きく 2 つに分けられます。分類対象データが少ない場合は「階層的クラスタリング」、分類対象が多数である場合は「非階層的クラスタリング」を用いることが一般的です。

階層的クラスタリングとは

階層的クラスタリングは、似ている特徴のあるデータをクラスターに順に結合していき、その経過が「デンドログラム」と呼ばれるトーナメント表（樹形図）のような図で視覚化し、データの特徴を把握することのできる分析手法です。図の中で近い場所に位置されるデータ同士は特徴が似ていると解釈します。

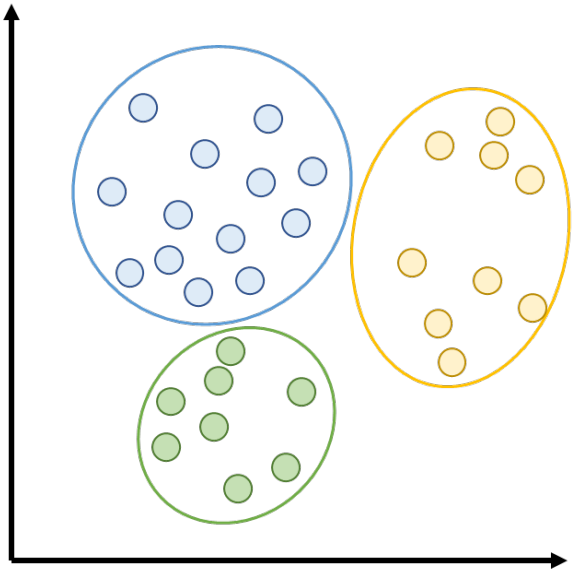
すべてのデータ間でお互いの類似度（距離が近いかどうか）や非類似度（距離が遠いかどうか）を計算します。さらに、似たもの同士を同じクラスターに併合していき、階層型クラスターを形成します。

よく用いられる類似度計算方法（距離の計算方法）はユークリッド距離、その後の結合過程の距離測定方法には様々な方法がありますが、よく使われているのは「ウォード法」です。いずれも全体にデータがバランス良く分類されやすいとされています。



非階層的クラスタリングとは

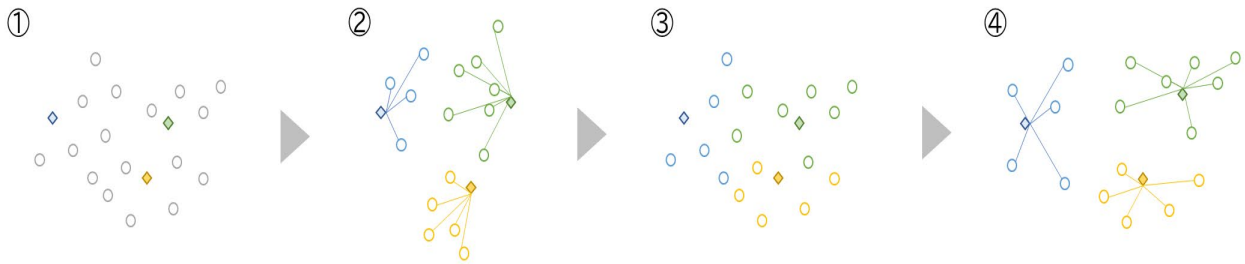
非階層的クラスタリングは、階層的な構造を持たず、いくつのクラスターに分けるかをあらかじめ決めておき、決めた数にデータを分割する手法です（数を決めないで、機械が自動で数を分割してくれる手法も存在します）。ビッグデータなど、処理する件数が膨大な場合にそれらを分類し、データの特徴や共通項を把握する際に非常に有用な手法です。この手法は、全てのデータ間の距離を計算する階層的手法よりも計算量が少なく済むため、ビッグデータを扱うのに重宝されています。



非階層的クラスタリングの代表的手法

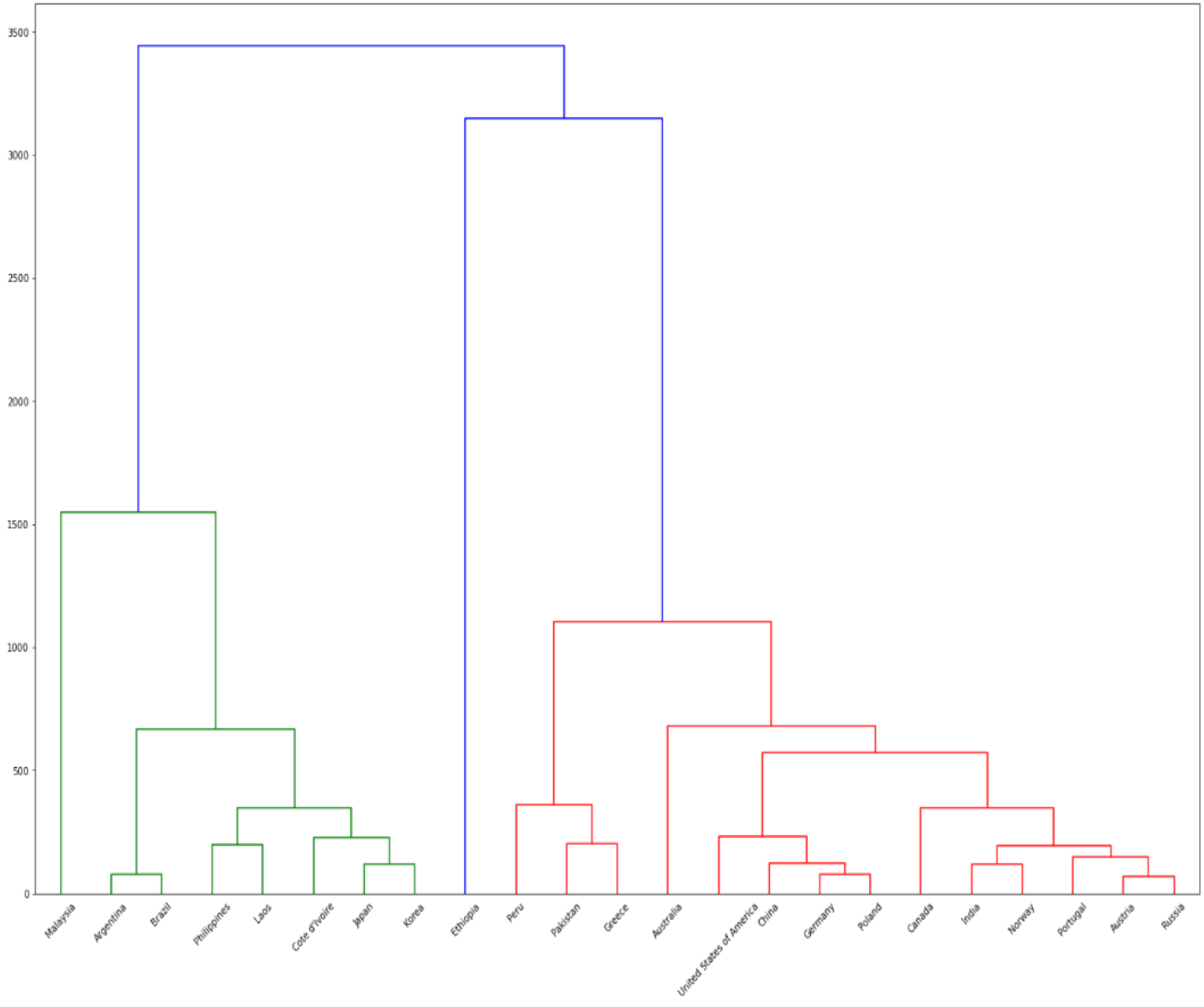
k-means 法（k-平均法）

k-means 法は非階層的クラスタリングの実行方法の一つで、もっとも代表的な手法です。あらかじめ指定された数（例えば k 個）のデータを「プロトタイプ」と指定し、他のそれぞれの個体データ（プロトタイプ以外のもの）を最も近いプロトタイプに割り当てることで最初のクラスターを割り当てます。個体データが割り当てられたら、次は、その時点で所属しているクラスターから、そのクラスターの重心を計算します。さらにその次は、それぞれの個体データが最も近いクラスターに所属を割り当て直します。このように重心の算出と個体データの割り当てを収束するまで計算を実行することにより、データの分割を行います。



インプットデータ

ID	森林の割合	最高気温	最低気温	年間降水量	首都の緯度	首都の経度	高度	平均湿度
1	68.5	26.4	5.2	1529	35.4	139.4	25.2	65
2	23.8	33.2	14.1	768	28.3	77.1	211	54
3	63.7	25.7	-2.4	1429	37.3	127.3	86	68.5
4		26.7	-3.1		39.5	116.3		60



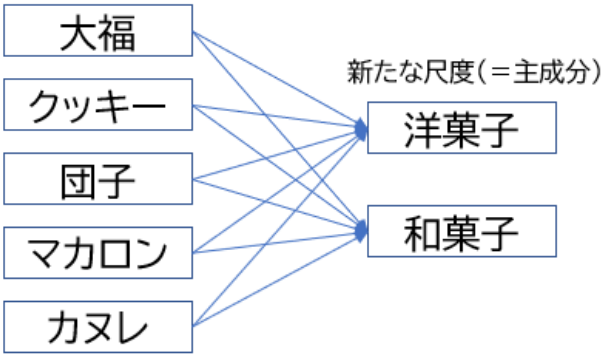
例題で機械学習を実行しよう（Python を用いた機械学習の実行）

結果の解釈

大きく 3 つのクラスターに　分割されています（緑、青、赤の 3 区分）。1 つだけ独立している国があります。これは Etiopia（エチオピア）です。インプットしたデータをよく確認して、どうしてこのエチオピアだけ他の国との結合がされないのかを考えてみましょう。他と異なるデータがあるかもしれません。

主成分分析

主成分分析とは、多数の変数の情報（観測変数）をできるだけ少ない指標や次元（合成変数）で要約する手法です。観測変数を要約した合成変数のことを「主成分」といいます。主成分分析を行うことにより、そのままでは理解しにくいビッグデータを、データの持つ情報をできるかぎり損なわずに全体の雰囲気を可視化し、理解しやすくすることができます。



Case　**主成分分析を使って複数のチャンネルを一つにまとめてみよう！**

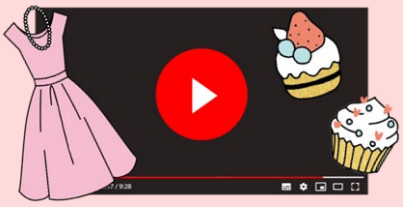
美緒さんは最近、YouTube にはまっています。

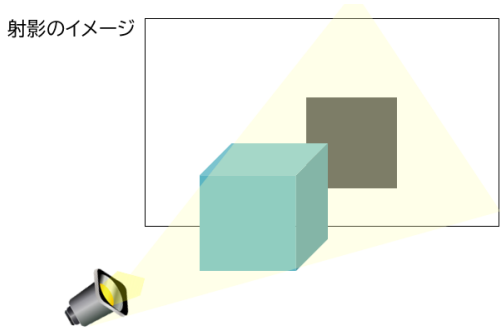
大好きなケーキや洋菓子について、また高校生になって興味を持ち始めたメイクやファッションについても、それらに関連するジャンルの色々なチャンネルを登録していました。

しかし、チャンネル登録をした結果、通知が沢山鳴りすぎて困ってしまいました。そこで、美緒さんは「興味のあるジャンルについてまんべんなく取り上げてくれるいくつかのチャンネルに絞ろう」と考えるようになりました。

「ケーキや洋菓子は食べて感想を言う、まで見たいな。スイーツに限った食レポ？みたいなジャンルがないかな。」「メイクやファッションに興味があるけど、あんまりお金はかけられない。スイーツに関してもそうだな。こういうのを全部ひっくるめたジャンルとか、うまく絞れるようなタグがわかれば！」
美緒さんはふと、学校で習った多変量解析の授業のことを思い出しました。

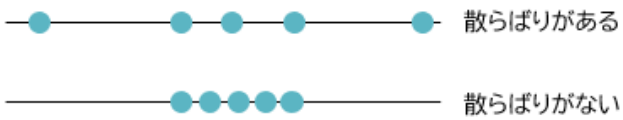
「そういえば、いくつかの要素をまとめて
一つの変数に要約してくれる「主成分分析」について
習ったな。私にもできるかな・・・？」





主成分分析は、データを「射影」してとらえることで、別の角度から捉え、データを扱いやすく変換しています。

また、その「変換」を行う際に「できるだけ情報がたくさん入っているように」変換します。できるだけデータをたくさん入れるためには「データの散らばり」を考慮できると情報を沢山入れることができます。この「データの散らばり」のことを「分散」と言います。



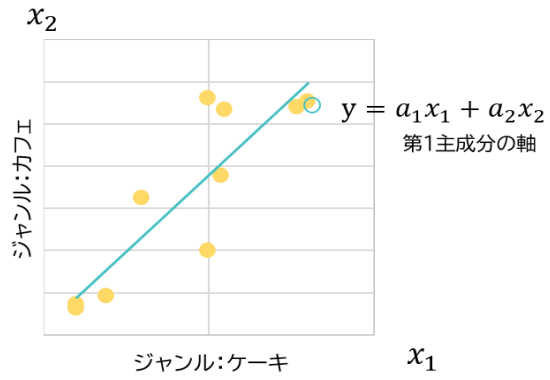
各チャンネルが、下記のようなジャンルの動画をアップロードしているという例で考えてみましょう。

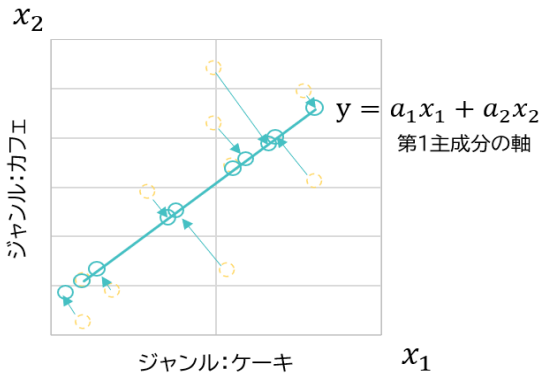
チャンネル No	ケーキ	カフェ	食レポ	スイーツ	食いしん坊	コンビニスイーツ	ファッション	スクールメイク	プチプラ	コスメ
1	55	4	66	66	3	60	44	30	40	45
2	2	0	40	25	50	55	0	0	0	0
3	30	60	40	30	0	0	0	0	0	0
4	30	59	3	44	0	3	20	1	2	44
5	0	0	0	0	0	0	2	20	4	60

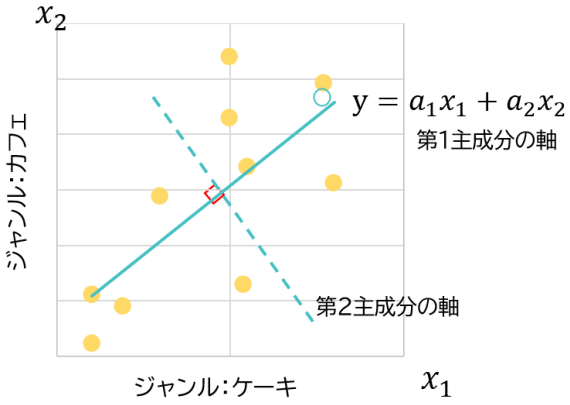
ここからは、説明しやすくするために、ケーキとカフェの関係で考えてみます。

ジャンル：ケーキと、ジャンル：カフェを 2 次元に配置します。ケーキとカフェという 2 つのデータから、新たに「食べ物」という主成分で情報を圧縮します。この 2 つのデータを圧縮して表す関数を考えたときに、この関数は

と表すことができます。この関数を、分散が最大になるように求めると第 1 主成分となります。次に、第 2 主成分は第 1　主成分と直交する軸の中で、軸上に射影した データの分散が最大となる軸を探します。







「分散」を計算してみましょう。

ジャンル:ケーキの登録数	c.平均との差 （偏差）	d. c の二乗 （偏差の二乗）
1		
3		
4		
7		
4		

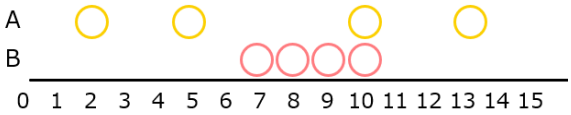
a. データの平均 [　　]　b.データの個数 [　　]

e. d.の合計 [　　]　f. e÷b [　　] ←これが分散

計算できたら、下記の A と B はどちらが分散が大きいかも考えてみましょう。

A(2, 5, 10, 13)

B(7, 8, 9, 10)



主成分の分散は、主成分がもつ情報量を分散として捉え、相関行列 R の固有値問題を解いています。固有値が大きい固有ベクトルほど、データの分散をよく説明している、つまり、データの重要な特徴を捉えていると考えられます。固有値と固有ベクトルは、行列を「線形」に変換する際に特徴を示す指標です。機械学習や統計の手法を実行する際、行列でできているデータを「線形」に置き換えて計算することが多く行われます。固有ベクトルは、データを線形に変換するための行列を掛けても、方向は変わらないベクトルとしてとても便利です。

例題で機械学習を実行しよう（Python を用いた機械学習の実行）

```
[1]  import numpy as np
     import pandas as pd
```

```
# 図やグラフを図示するためのライブラリ
# 日本語表示用ライブラリもインポート
!pip install japanize-matplotlib
import matplotlib.pyplot as plt
import japanize_matplotlib
%matplotlib inline
```

```
Collecting japanize-matplotlib
  Downloading https://files.pythonhosted.org/packages/aa/85/08a4b7fe8987582d99d9bb7ad0ff1ec75439359a7f9690a0dbf2dbf98b15/japanize-matplotlib-1.1.3.tar.gz (4.1MB)
     |████████████████████████████████| 4.1MB 6.0MB/s
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from japanize-matplotlib) (3.2.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib->japanize-matplotlib) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->japanize-matplotlib) (1.3.1)
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.7/dist-packages (from matplotlib->japanize-matplotlib) (1.19.5)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->japanize-matplotlib) (2.8.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->japanize-matplotlib) (2.4.7)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from cycler>=0.10->matplotlib->japanize-matplotlib) (1.15.0)
Building wheels for collected packages: japanize-matplotlib
  Building wheel for japanize-matplotlib (setup.py) ... done
  Created wheel for japanize-matplotlib: filename=japanize_matplotlib-1.1.3-cp37-none-any.whl size=4120276 sha256=6058f35b92b7598998ca7f7cbac38b198a08e88789889bd19
  Stored in directory: /root/.cache/pip/wheels/b7/d9/a2/f907d50b32a2d2008ce5d691d30fb6569c2c93eefcfde55202
Successfully built japanize-matplotlib
Installing collected packages: japanize-matplotlib
Successfully installed japanize-matplotlib-1.1.3
```

```
[3]  import sklearn #機械学習のライブラリ
     from sklearn.decomposition import PCA #主成分分析器
```

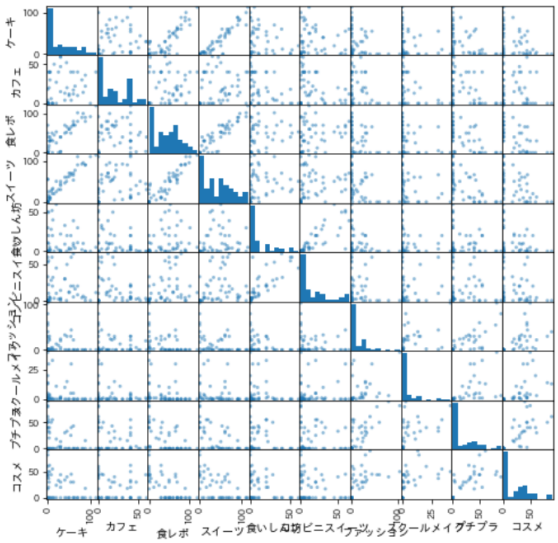
```
[5] df_pca = pd.read_csv("pca.csv",encoding = "shift-jis")
```

```
[6] df_pca.head()
```

	チャンネル名	ケーキ	カフェ	食レポ	スイーツ	食いしん坊	コンビニスイーツ	ファッション	スクールメイク	プチプラ	コスメ
0	チャンネル1	55	4	66	66	3	60	44	30	40	45
1	チャンネル2	2	0	40	25	50	55	0	0	0	0
2	チャンネル3	30	60	40	30	0	0	0	0	0	0
3	チャンネル4	30	59	3	44	0	3	20	1	2	44
4	チャンネル5	0	0	0	0	0	0	2	20	40	60

```
[7] from pandas import plotting
    # plotting.scatter_matrix(df_pca.iloc[:, 1:], figsize=(8, 8), c=list(df.iloc[:, 0]), alpha=0.5)
    from pandas import plotting
    plotting.scatter_matrix(df_pca.iloc[:, 1:], figsize=(8, 8), alpha=0.5)
    plt.show()
```


```
[7] from pandas import plotting
    # plotting.scatter_matrix(df_pca.iloc[:, 1:], figsize=(8, 8), c=list(df.iloc[:, 0]), alpha=0.5)
    from pandas import plotting
    plotting.scatter_matrix(df_pca.iloc[:, 1:], figsize=(8, 8), alpha=0.5)
    plt.show()
```

[8]
```
# 行列の標準化
df_pca_stand = df_pca.iloc[:, 1:].apply(lambda x: (x-x.mean())/x.std(), axis=0)
df_pca_stand.head(3)
```

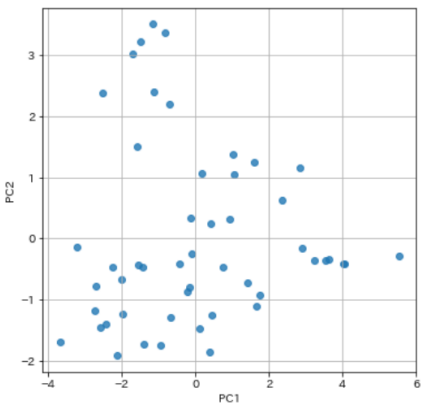
	ケーキ	カフェ	食レポ	スイーツ	食いしん坊	コンビニスイーツ	ファッション	スクールメイク	プチプラ	コスメ
0	0.730517	-0.824120	0.675172	0.680464	-0.568751	2.171877	1.314706	2.588607	0.776056	0.960389
1	-0.928326	-1.037071	-0.098835	-0.524157	2.281066	1.920967	-0.623869	-0.520209	-0.779166	-0.875529
2	-0.051956	2.157193	-0.098835	-0.377252	-0.750654	-0.839043	-0.623869	-0.520209	-0.779166	-0.875529

[9]
```
#主成分分析の実行
pca = PCA()
pca.fit(df_pca_stand)
# データを主成分空間に写像
feature = pca.transform(df_pca_stand)
```

[10]
```
# 主成分得点
pd.DataFrame(feature, columns=["PC{}".format(x + 1) for x in range(len(df_pca_stand.columns))]).head()
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
0	1.059752	1.037929	3.281462	-1.345759	-0.702272	0.558881	1.085325	0.602706	0.003692	-0.005901
1	-0.837499	3.354364	-1.030959	0.165872	0.056898	0.162140	0.079003	-0.182364	0.042608	-0.185159
2	-0.952280	-1.737664	-1.755337	-0.589728	-0.455065	0.662685	-0.185447	0.237100	-0.010275	-0.130298
3	0.384803	-1.852359	-1.311360	-0.810266	-1.075875	-0.351873	0.584331	-0.816635	-0.488922	0.043436
4	2.896199	-0.148181	-0.122281	-0.997403	1.589422	-0.594160	-0.059724	0.150631	0.135430	0.009475

```
# 第一主成分と第二主成分でプロットする
plt.figure(figsize=(6, 6))
plt.scatter(feature[:, 0], feature[:, 1], alpha=0.8)
plt.grid()
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.show()
```



```
[13] # 寄与率
     pd.DataFrame(pca.explained_variance_ratio_, index=["PC{}".format(x + 1) for x in range(len(df_pca_stand.columns))])
```

	0
PC1	0.472086
PC2	0.212715
PC3	0.138437
PC4	0.054656
PC5	0.049795
PC6	0.030128
PC7	0.015340
PC8	0.014318
PC9	0.009140
PC10	0.003386

```
[14] # 累積寄与率を図示する
     import matplotlib.ticker as ticker
     plt.gca().get_xaxis().set_major_locator(ticker.MaxNLocator(integer=True))
     plt.plot([0] + list( np.cumsum(pca.explained_variance_ratio_)), "-o")
     plt.xlabel("Number of principal components")
     plt.ylabel("Cumulative contribution rate")
     plt.grid()
     plt.show()
```

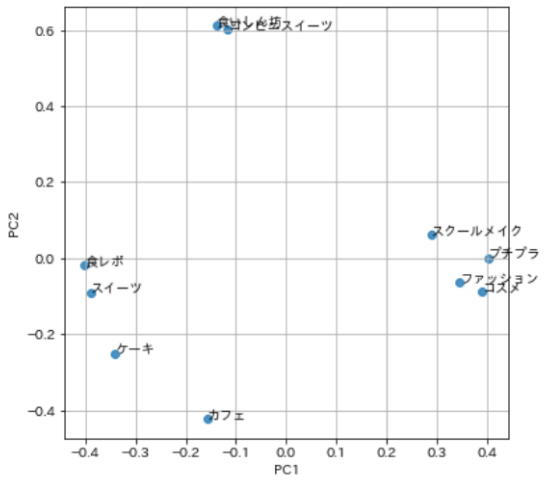
```
[15] # PCA の固有値
     pd.DataFrame(pca.explained_variance_, index=["PC{}".format(x + 1) for x in range(len(df_pca_stand.columns))])
```

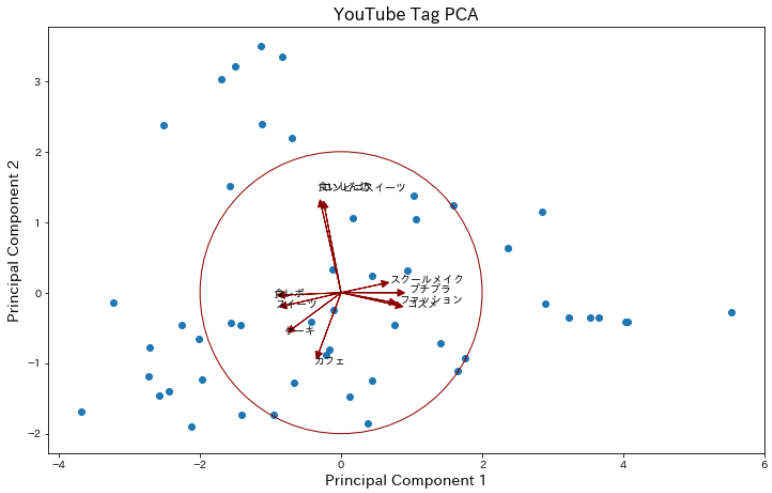
	0
PC1	4.720861
PC2	2.127149
PC3	1.384371
PC4	0.546560
PC5	0.497945
PC6	0.301279
PC7	0.153401
PC8	0.143177
PC9	0.091397
PC10	0.033860

```
[16] # PCA の固有ベクトル
     pd.DataFrame(pca.components_, columns=df_pca.columns[1:], index=["PC{}".format(x + 1) for x in range(len(df_pca_stand.columns))])
```

	ケーキ	カフェ	食レポ	スイーツ	食いしん坊	コンビニスイーツ	ファッション	スクールメイク	プチプラ	コスメ
PC1	-0.340668	-0.157116	-0.400958	-0.388064	-0.139218	-0.116152	0.344557	0.289168	0.401104	0.389620
PC2	-0.252560	-0.421955	-0.017219	-0.089777	0.610583	0.600922	-0.064939	0.062042	-0.000936	-0.088460
PC3	0.452019	-0.354245	0.342630	0.405082	-0.084285	0.110143	0.312388	0.427906	0.253881	0.155955
PC4	0.017224	-0.420758	0.070596	-0.022575	-0.018688	-0.188398	0.415606	-0.705553	0.226274	-0.244124
PC5	0.004090	-0.676407	-0.107237	-0.085147	-0.211588	-0.368155	-0.494561	0.190328	-0.206943	-0.140565
PC6	-0.141463	0.076452	0.005576	-0.147571	-0.143109	0.011549	0.419204	0.399409	-0.196191	-0.746975
PC7	-0.002662	-0.162198	-0.142589	0.001026	-0.555427	0.517413	0.181112	-0.178605	-0.506582	0.236240
PC8	-0.040420	0.057907	0.269086	-0.245377	-0.448469	0.381762	-0.377645	-0.052630	0.562458	-0.231916
PC9	-0.349777	-0.039155	0.782303	-0.317449	0.009187	-0.164206	0.093986	0.034565	-0.241106	0.261041
PC10	-0.686679	-0.013214	0.017006	0.697663	-0.159362	-0.041023	-0.046647	0.002737	0.105004	-0.031407

```
[17] # 第一主成分と第二主成分における観測変数の寄与度をプロット
     plt.figure(figsize=(6, 6))
     for x, y, name in zip(pca.components_[0], pca.components_[1], df_pca.columns[1:]):
         plt.text(x, y, name)
     plt.scatter(pca.components_[0], pca.components_[1], alpha=0.8)
     plt.grid()
     plt.xlabel("PC1")
     plt.ylabel("PC2")
     plt.show()
```

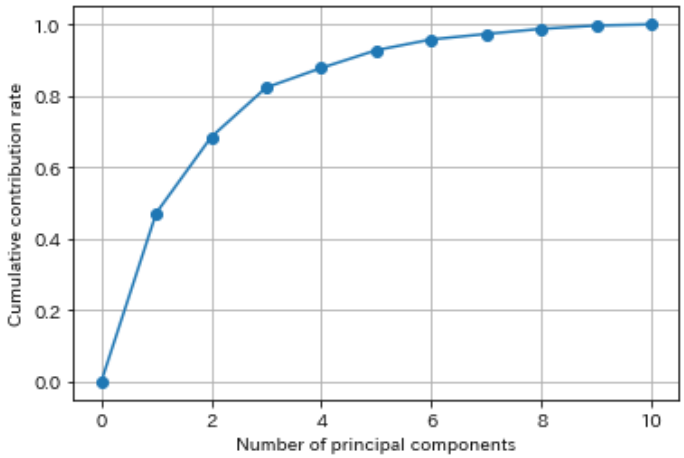




結果の解釈

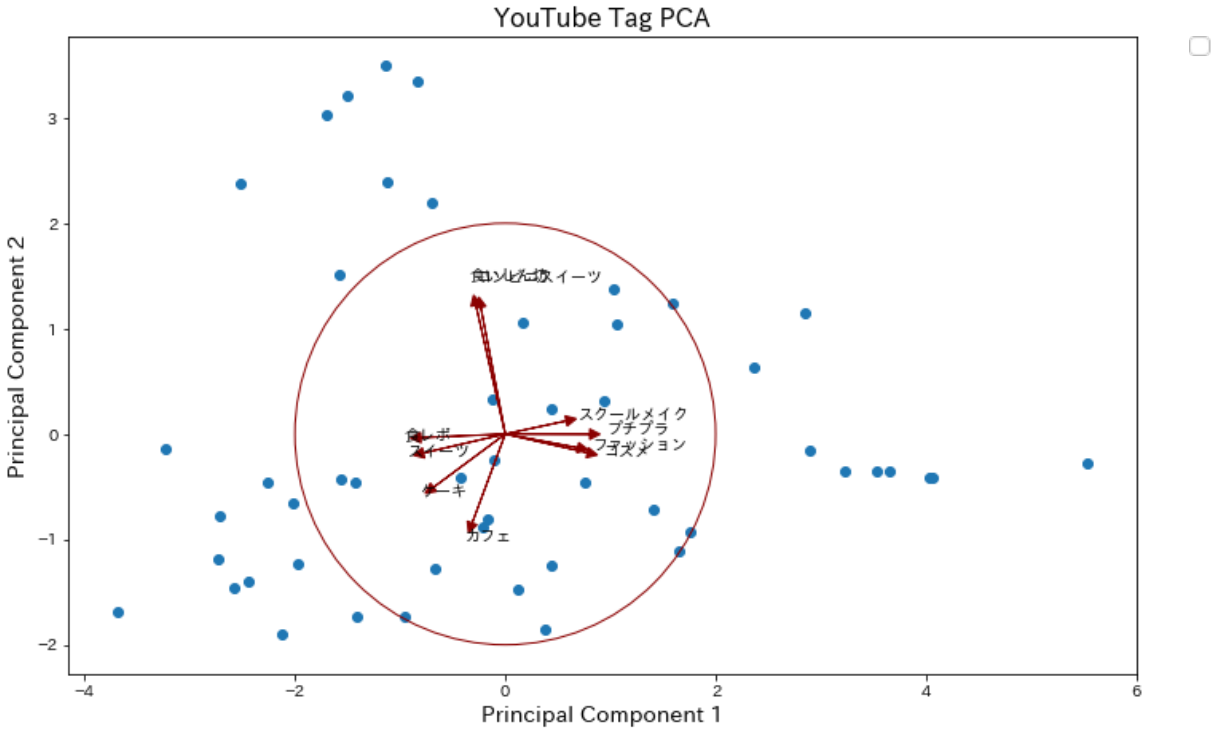
主成分分析を行う目的は、データを「まとめて扱いやすくする」ことでした。たくさんのデータを少しのデータにまとめるのに「いくつの」データにすればよいでしょうか。主成分分析の計算過程に「寄与率」「累積寄与率」が算出されています。この「寄与率」はその主成分が占めるデータの割合を示し、「累積寄与率」は、その累積の値です。

	寄与率
PC1	0.472086
PC2	0.212715
PC3	0.138437
PC4	0.054656
PC5	0.049795
PC6	0.030128
PC7	0.01534
PC8	0.014318
PC9	0.00914
PC10	0.003386



PC1（第１主成分）だけで主成分分析の対象としたデータの 47%、PC2 で 21%、合わせて 68% データを表すことが出来ています。第 3 主成分まで含めると、80%を超えて表すことができます。今回、分析に用いたデータは10の変数がありました。そのため、第10主成分まで計算され、それで100%のデータとなりますが、主成分分析は「データを纏めて扱いやすくする」ことが目的です。8割に達する第3主成分くらいでデータを扱うと良さそうです。

では、10のチャンネルはどのように3つの成分として捉えれば良いでしょうか。第1主成分と第2主成分の得点と、固有ベクトルのプロットを見てみましょう。



同じ方向に固まっている変数を見てみましょう。[食レポ・スイーツ・ケーキ・カフェ]と、[スクールメイク・プチプラ・ファッション・コスメ]、[食いしん坊・コンビニスイーツ]という3方向の矢印が示されています。まとめて表すとしたら、どのような「主成分名」がふさわしいでしょうか？ データサイエンスは、これらを言い表す「言葉」を選ぶ能力も大切です。スイーツ食べ歩き、おしゃれ、家スイーツなどが良いでしょうか。考えてみましょう。

因子分析

因子分析とは

因子分析とは多数の観測変数の背後に潜む少数の潜在因子を想定して理解するための手法です。多数の観測変数に存在する共通因子を仮定し、モデル化して分析する点が主成分分析との違いです。

Case

興味のある SDGs と性格特性は関係がある？

美緒さんと翔真さんのクラスでは、二人一組で SDGs17 の目標に関して、まとめて発表をすることになりました。今日はどのペアがどの目標を担当するか決める話し合いです。

話し合いはなかなか進みませんが、何組かはテーマ決まりました。はきはきしている子らのペアはジェンダー平等について、責任感が強い子らのペアは飢餓について調べると言います。そこで、美緒さんは「もしかしたら、どのテーマに興味を持つかは性格が関係しているかもしれない。」と考えるようになりました。

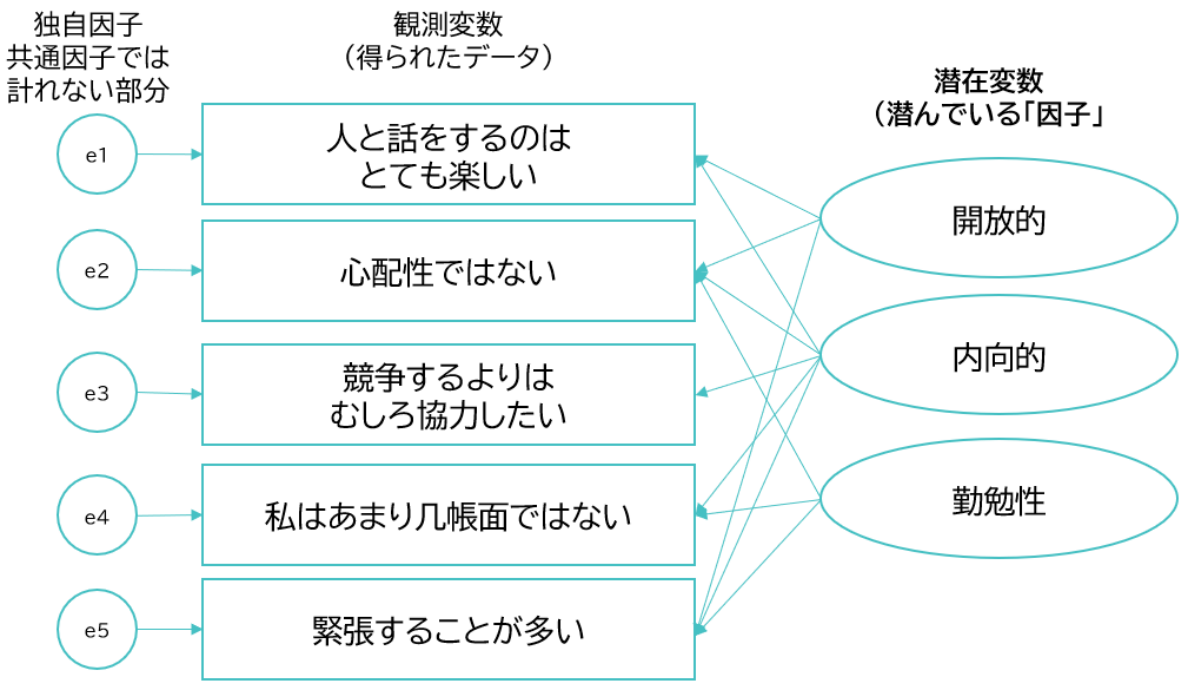
「ある性格の人には、どんな問題が心に響くはずだ、というような紐づけができないかな。」
「そうするには、SDGs の各目標がどのような要素で成り立っているかわかれば。
性格ってでも人それぞれ違うけど、なにか傾向があるはず」

美緒さんは、さっそくクラスの皆に「性格について」と興味のある
SDGsの項目についてのアンケート」をお願いしました。
性格についてはたくさん聞きたかったので、20の質問に
答えてもらいました。すると、項目が多すぎてしまい、せっかくデータがあるのに
何がどうなっているのかわからなくなってしまいました。
「性格って、でも何か共通するものがあるよね。あ！そういえば、
ある問題を説明するより小さな因子を考える「因子分析」について習ったな。私にもできるかな・・・？」

因子分析では、取得できた「データ」変数の背後に共通する原因を見出すことで、複数あるデータを少しのデータで扱えるようにします。データの数が増えれば増えるほど複雑になってしまいますが「共通原因」を考慮させることで、現象を「わかりやすく」捉えられるようになります。

アンケートなどで得られた「データ」は観測変数と呼ばれます。データに潜んでいるかもしれない「因子」は楕円で表します。さらに、共通因子/潜在変数では説明のできない現象も実際には起こり得るので「独自因子」も存在しています。因子分析を行う際は、分析するよりも前の過程で「どのような因子が潜んでいるのか」を考えておくことがとても大切です。

「どのようなことが因子として潜んでいるか」を考える際に、このような図（パス図といいます）を用いて検討します。このとき、結果として得られるデータ、観測変数は四角で、潜んでいる「因子」潜在変数と独自因子は楕円で描きます。矢印の向きは、因子から影響を受けている方向へ矢印を引きます。主成分分析とは矢印の向きが逆であることがポイントです。



要約して総合指標を作成する主成分分析は観測変数が原因で主成分が結果ですが、共通の原因を探る因子分析は因子が原因で観測変数が結果です。

共通因子を抽出するアルゴリズムも複数あり、主因子法、最尤法、最小二乗法などがありますが、「観測変数間の相関構造を説明できるような共通因子と独自因子の分散」を求めることがポイントです。最も一般的なものは最尤法であるとされています。

事前に観測変数内に共通因子がいくつ潜んでいるか？は、データサイエンティストがあらかじめ想定する必要があります。因子数を指定して分析するため、異なる因子数を指定すると結果が異なります。実際には何パターンか試したもののなかから分析の課題に対してふさわしいものを選びます。選ぶ基準としては相関行列の１より大きい固有値の数で決める（カイザー基準）というやり方や、固有値を可視化して（スクリープロットと呼ばれています）　固有値が急に小さくなる手前で決める、などのやり方があります。しかし基準だけで因子数をきめるのではなく、データが取得された背景や過去の知見などから総合的に判断して決定します。

例題で機械学習を実行しよう（Python を用いた機械学習の実行）

```
[1]  # 数値計算ライブラリ
     import numpy as np
     import pandas as pd
     # 可視化ライブラリ
     import matplotlib.pyplot as plt
     %matplotlib inline
```

```
[22] df = pd.read_csv("Factor_SDGs.csv",encoding = "shift-jis")
     df.head()
```

ＩＤ	物事を時間通りに終わらせるようにペースを作るのがかなり得	基本的にすべての人に親切にするように心がけている	人と話をするのはとても楽しい	私は心配性ではない	空想にふけって時間を無駄に使うのは嫌だ	他人と競争するよりはむしろ協力したい	私はあまり几帳面ではない	緊張したりびくびくしたりすることが多い	境応て化す、分気や持に気付ない環にじ変る、自の分気ちはがか	手人対て、い度出しうとある苦なにしはつ態にてまこが	急いでやらなければいけないことがあるのに先延ばしにしてしまう	分力はきいと、かのに決てしとうとある自のでなこを誰他人解しほい思こが	私は生活のテンポが早い	私は情に流されない	私は好奇心が旺盛である	自分の目標を達成するためには努力を惜しまない	ーなよりは、我道行たリダにる	しなた、ちんりるとない悲くっり落込だすこは	理論的なことや抽象的な考えにふけって楽しむことがよくあ	年齢	都道府県	性別	結婚	職業	業種	世帯収	世年	居住形態	子供有無	なはGsつていしたあたSDGにい知てまか	貧困をなくそう
0	3	3	3	3	3	2	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
1	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
2	3	2	3	3	3	3	3	3	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	4	3	3	4	2	2	3	2	3	2	4	2	3	2	3	2	2	2	2	3	2	2	2	2	2	3	2	2	2	2
4	3	3	4	4	3	3	3	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3

```
[23] df=df2
     df.head()
```

▼ データの用意

```
[24] # データの標準化
     # sklearnの標準化モジュールをインポート
     from sklearn.preprocessing import StandardScaler

     # データを変換する計算式を生成
     sc = StandardScaler()
     sc.fit(df)

     # 実際にデータを変換
     z = sc.transform(df)
```

▼ 因子数の推定

```
[25] # sklearnのPCA(主成分分析)クラスをインポート
     from sklearn.decomposition import PCA

     # 主成分分析のモデルを生成
     pca = PCA() # インスタンスを生成・定義
     pca.fit(z)  # 標準化得点データにもとづいてモデルを生成
```

```
PCA(copy=True, iterated_power='auto', n_components=None, random_state=None,
    svd_solver='auto', tol=0.0, whiten=False)
```

```
[26] # 寄与率の取得
     evr = pca.explained_variance_ratio_
     pd.DataFrame(evr,
                  index=["PC{}".format(x + 1) for x in range(len(df.columns))],
                  columns=["寄与率"])
```

	寄与率
PC1	0.257397
PC2	0.177229
PC3	0.085597
PC4	0.068869
PC5	0.050938
PC6	0.042132
PC7	0.039048
PC8	0.033670
PC9	0.031793
PC10	0.028822
PC11	0.026668
PC12	0.023878
PC13	0.021804
PC14	0.019245
PC15	0.018069
PC16	0.017150
PC17	0.015912
PC18	0.015077
PC19	0.014012
PC20	0.012692

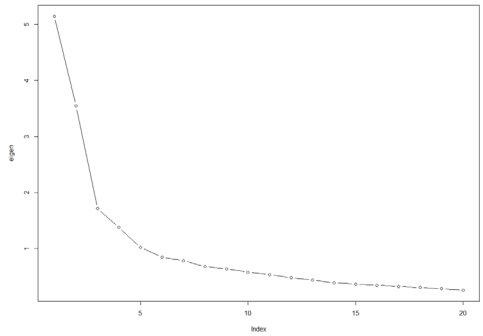
▼ 因子分析

```
# sklearnのFactorAnalysis(因子分析)クラスをインポート
from sklearn.decomposition import FactorAnalysis as FA

# 因子数を指定
n_components=5

# 因子分析の実行
fa = FA(n_components, max_iter=5000) # モデルを定義
fitted = fa.fit_transform(z) # fitとtransformを一括処理

print(fitted)
print(fitted.shape)
```

```
[[ 0.10900651  0.46127867  0.35506488 -0.02764302 -0.58579098]
 [ 3.55142104 -2.4118441   0.11498021 -0.27775706 -0.06780467]
 [ 0.14018723  0.04857657  0.37928002 -0.55123965 -0.16858431]
 ...
 [-0.20502502  0.20893528  0.46789057  0.01755921 -0.44376035]
 [ 0.26410266 -1.25225072 -0.16017214  0.15735426 -0.98829475]
 [-0.39947601 -1.65290582 -0.97952334 -1.38443944  2.00215084]]
(300, 5)
```

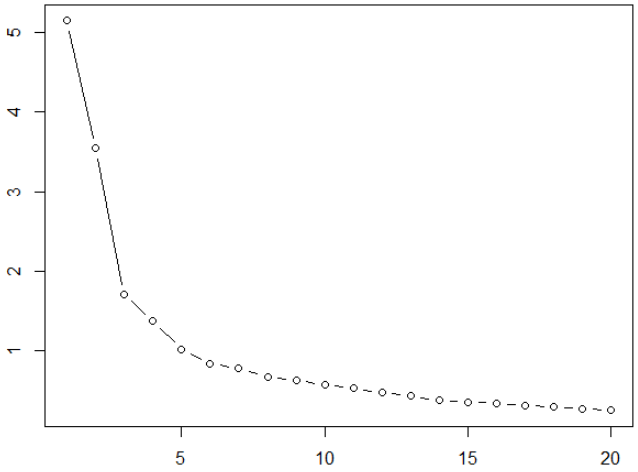
[31]
```
# 因子の解釈
# 変数Factor_loading_matrixに格納
Factor_loading_matrix = fa.components_.T

# データフレームに変換
pd.DataFrame(Factor_loading_matrix,
             columns=["第1因子", "第2因子", "第3因子", "第4因子", "第5因子"],
             index=[df.columns])
```

	第1因子	第2因子	第3因子	第4因子	第5因子
基本的にすべての人に親切にするように心がけている	0.691519	0.117495	0.213516	-0.239118	0.187973
物事を時間通りに終わらせるようにペースを作るのがかなり得意だ	0.618674	0.219078	0.142827	0.209795	0.072856
人と話をするのはとても楽しい	0.736080	0.013359	0.080278	-0.193717	0.013460
私は心配性ではない	0.527128	-0.107450	-0.556706	-0.150303	-0.009217
空想にふけって時間を無駄に使うのは嫌だ	0.404682	0.172081	-0.345373	0.240991	0.340262
他人と競争するよりはむしろ協力したい	0.446062	0.293721	0.271092	-0.345229	0.221687
私はあまり几帳面ではない	-0.137434	0.448931	-0.311400	-0.350794	0.054712
私はあまり陽気ではない	-0.441127	0.578939	-0.068076	0.159030	0.157979
緊張したりびくびくしたりすることが多い	-0.350915	0.555080	0.261186	0.118095	0.124220
環境に応じて変化する、自分の気分や気持ちには気が付かない	-0.075906	0.592986	-0.123402	0.117123	0.132603
苦手な人に対しては、つい態度に出てしまうことがある	-0.258515	0.558467	0.071734	0.099463	-0.155105
急いでやらなければいけないことがあるのに先延ばしにしてしまうことがある	-0.255994	0.523481	-0.267380	-0.395491	-0.171014
私は生活のテンポが早い	0.514650	0.220794	-0.032559	0.236930	-0.109035
自分の力ではできないことを、誰か他の人に解決してほしいと思うことがある	0.005603	0.492890	0.258483	-0.003360	-0.060675
私は好奇心が旺盛である	0.613562	0.090985	0.170937	0.005620	-0.345154
私は情に流されない	0.339612	0.301593	-0.271154	0.437293	-0.161147
自分の目標を達成するためには努力を惜しまない	0.668623	0.130333	0.207516	0.226477	-0.030353
リーダーになるよりは、我が道を行きたい	-0.056173	0.364277	0.182266	-0.122531	-0.234834
悲しくなったり、落ち込んだりすることはない	0.499300	0.205388	-0.415807	0.048007	-0.056296
理論的なことや抽象的な考えにふけって楽しむことがよくある	0.252969	0.385620	0.116427	-0.119105	-0.240326



結果の解釈

例では因子の数を5にしています。まず、因子はいくつにするのが良いでしょうか。スクリープロットといわれる固有値を大きい順に並べたグラフを見てみましょう。



因子の数を決める基準はいくつかありますが、固有値が1よりも大きいものや、急落するあたりで決める方法がよく知られています。1より大きいものは5つあります。急落する手前だと3つです。今回は、1よりも大きい5つに決定します。

では、それぞれの「因子」はどのように解釈すればよいでしょうか。それについては、それぞれの因子がどの観測変数にどのように影響しているかのスコアをみてみましょう。

	第１因子	第２因子	第３因子	第４因子	第５因子
基本的にすべての人に親切にするように心がけている	0.69	0.12	0.21	-0.24	0.19
物事を時間通りに終わらせるようにペースを作るのがかなり得意だ	0.62	0.22	0.14	0.21	0.07
人と話をするのはとても楽しい	0.74	0.01	0.08	-0.19	0.01
私は心配性ではない	0.53	-0.11	-0.56	-0.15	-0.01
空想にふけって時間を無駄に使うのは嫌だ	0.40	0.17	-0.35	0.24	0.34
他人と競争するよりはむしろ協力したい	0.45	0.29	0.27	-0.35	0.22
私はあまり几帳面ではない	-0.14	0.45	-0.31	-0.35	0.05
私はあまり陽気ではない	-0.44	0.58	-0.07	0.16	0.16
緊張したりびくびくしたりすることが多い	-0.35	0.56	0.26	0.12	0.12
環境に応じて変化する、自分の気分や気持ちには気が付かない	-0.08	0.59	-0.12	0.12	0.13
苦手な人に対しては、つい態度に出てしまうことがある	-0.26	0.56	0.07	0.10	-0.16
急いでやらなければいけないことがあるのに先延ばしにしてしまうことがある	-0.26	0.52	-0.27	-0.40	-0.17
私は生活のテンポが早い	0.51	0.22	-0.03	0.24	-0.11
自分の力ではできないことを、誰か他の人に解決してほしいと思うことがある	0.01	0.49	0.26	-0.00	-0.06
私は好奇心が旺盛である	0.61	0.09	0.17	0.01	-0.35
私は情に流されない	0.34	0.30	-0.27	0.44	-0.16
自分の目標を達成するためには努力を惜しまない	0.67	0.13	0.21	0.23	-0.03
リーダーになるよりは、我が道を行きたい	-0.06	0.36	0.18	-0.12	-0.23
悲しくなったり、落ち込んだりすることはない	0.50	0.21	-0.42	0.05	-0.06
理論的なことや抽象的な考えにふけって楽しむことがよくある	0.25	0.39	0.12	-0.12	-0.24

第1因子は「基本的にすべての人に親切にするように心がけている」「人と話をするのはとても楽しい」などのコミュニケーションに関する観測変数に影響を与えています。第2因子は「環境に応じて変化する、自分の気分や気持ちには気が付かない」「私はあまり陽気ではない」「苦手な人に対しては、つい態度に出てしまうことがある」などに影響を与えています。第3因子は、「心配性ではない」に負の符号で影響を与えています。因子の解釈を行う際は、符号にも気をつけながら「潜んでいること」がなにであるかを解釈します。

名付けるとすれば、第1因子は協調性、第2因子は神経性や内向的、第3因子は開放性やおおらかさ、第4因子はマイペースで勤勉、頑固な特性、第5因子は外向性といった名前が考えられそうです。人の性格には「性格5因子」があるとされる理論があります。今回は分析結果から下記のような「因子」を名付けることにします。

第1因子：協調性　第2因子：神経性　第3因子：開放性　第4因子：誠実・統制性　第5因子：外向性

さて、美緒さんが知りたかったのは「性格と SDGs 項目への興味関心」です。アンケートでは、SDGsを知っているかどうか、どの SDGsの項目に興味があるかも聞いています。20の性格についての項目は、5つの因子で表すことが出来ました。20の項目は5つの因子に変換することができます。これを「因子得点」といいます。因子得点と、興味のある SDGsの関係性を把握するのにはどのような方法が良いでしょうか。因子得点は、アンケートの回答者によって高い/低いという数量になっています。SDGsの項目ごとの興味関心は、興味があるかどうかの5段階で聞きました。その関係性を「相関係数」を算出することにより把握してみます。各項目と、SDGsへの興味関心の相関係数は下記のようになりました。

	協調性	神経性	開放性	誠実・統制性	外向性
あなたは SDGs について知っていましたか？	0.22	0.03	0.39	-0.04	0.37
貧困をなくそう	0.22	0.12	0.52	-0.11	0.39
飢餓をゼロに	0.28	0.11	0.56	-0.08	0.37
すべての人に健康と福祉を	0.21	0.07	0.55	-0.11	0.36
質の高い教育をみんなに	0.29	0.11	0.56	-0.10	0.41
ジェンダー平等を実現しよう	0.21	0.05	0.53	-0.05	0.32
安全な水とトイレを世界中に	0.20	0.12	0.52	-0.16	0.35
エネルギーをみんなにそしてクリーンに	0.19	0.10	0.53	-0.12	0.34
働きがいも経済成長も	0.27	0.08	0.58	-0.14	0.33
産業と技術革新の基盤をつくろう	0.29	0.08	0.58	-0.16	0.35
人や国の不平等をなくそう	0.19	0.11	0.57	-0.10	0.34
住み続けられるまちづくりを	0.20	0.12	0.57	-0.13	0.37
つくる責任　つかう責任	0.27	0.09	0.62	-0.09	0.38
気候変動に具体的な対策を	0.13	0.06	0.52	-0.08	0.31
海の豊かさを守ろう	0.13	0.10	0.51	-0.13	0.37
陸の豊かさも守ろう	0.13	0.10	0.51	-0.10	0.38
平和と公平をすべての人に	0.14	0.10	0.54	-0.09	0.38

パートナーシップで目標を達成しよう	0.35	0.08	0.62	-0.10	0.41

「あなたは SDGs について知っていましたか？」という項目に対しては、開放性の因子、外向性の因子とゆるやかな相関が認められます。また「開放性」はどの SDGsの項目においても相関が認められます。誠実・統制性と名付けた第 4 因子は、ほぼどの項目とも相関が認められません。この因子には「私は情に流されない」「空想にふけって時間を過ごしてしまうのは嫌だ」という観測変数に影響を与えていました。マイペースで頑固な性格といえる因子であったことは、SDGs へ興味関心は低いのでしょうか。ここで「人の性格因子」について考えてみましょう。因子は各アンケートの項目（観測変数）に影響を与えていますが、因子得点は人によって様々なパターンがあります。

協調性の因子が高く、神経内向性の高い人もいれば、協調性の因子は高いが、神経内向性は低い人もいるでしょう。アンケートの 20 項目から 5 つの因子は見出すことができましたが、その 5 つの因子も人によって様々な組み合わせが存在します。また、このデータは、本書のために実際に 300 人にアンケート調査を実施したものから算出しています。300 人の「5つの因子」には、様々な組み合わせパターンがあるはずです。それらをさらに分類するためには「因子得点」を用いたクラスター分析なども検討できます。300 人と人数が多いので、非階層型クラスター分析を用いるのが良いでしょう。

Try：因子得点を用いて、300 人の人の「性格因子によるクラスター」を作成してみよう

例：因子得点を用いたクラスター分析結果（非階層型クラスター分析　K-means 法、ユークリッド距離）

＜表：クラスター別 因子得点の重心＞

	Cluster1	Cluster2	Cluster3	Cluster4
協調性	0.05	-1.66	-0.15	1.14
神経性	0.58	-0.88	0.38	-1.15
開放性	0.38	-0.95	-0.67	0.84
誠実・統制性	-0.11	-0.91	0.85	-0.50
外向性	0.89	-0.40	-0.94	-0.08
人数	105	33	80	54

因子を用いてアンケートに回答された方たちを「似ている性格因子」で 4 つのクラスターに部類することが出来ました。Cluster1 は外向性が高く開放的な方たちのようです。Cluser2 はすべての因子が低い傾向ですが、特に協調性が低く、内向的な方たちのようです。Cluster3 は、誠実・当生成と神経性が高く開放性や外向性の低い、こちらも内向的な方たちですが Cluster2の方たちと比べると協調性の因子は低くありません。

Cluster4 の方たちは、協調性と開放性が高く、誠実・統制性は高くない方たちが集まっています。では、このクラスター別に SDGsへの興味関心を見てみましょう。下記の表は、所属するクラスターごとに SDGsの認知度を「よく知っている/知っている」と回答した割合と「興味がある・少し興味がある」と回答した人の割合です。

	Cluster1	Cluster2	Cluster3	Cluster4
人数	n=105	n=33	n=80	n=54
因子の特徴	外向性・神経性が高い	協調性・開放性が低い	誠実・統制性が高く外向性が低い	協調性・開放性が高い
あなたは SDGs について知っていましたか？	57.1%	15.2%	22.5%	50.0%
貧困をなくそう	19.0%	6.1%	5.0%	16.7%
飢餓をゼロに	20.0%	3.0%	5.0%	13.0%
すべての人に健康と福祉を	16.2%	3.0%	3.8%	11.1%
質の高い教育をみんなに	17.1%	3.0%	1.3%	13.0%
ジェンダー平等を実現しよう	21.9%	6.1%	11.3%	20.4%
安全な水とトイレを世界中に	16.2%	0.0%	3.8%	9.3%
エネルギーをみんなにそしてクリーンに	17.1%	3.0%	5.0%	9.3%
働きがいも経済成長も	21.9%	0.0%	5.0%	20.4%
産業と技術革新の基盤をつくろう	19.0%	0.0%	2.5%	14.8%
人や国の不平等をなくそう	21.0%	0.0%	3.8%	13.0%
住み続けられるまちづくりを	14.3%	3.0%	3.8%	9.3%
つくる責任　つかう責任	18.1%	0.0%	1.3%	11.1%
気候変動に具体的な対策を	20.0%	6.1%	6.3%	9.3%
海の豊かさを守ろう	14.3%	6.1%	3.8%	9.3%
陸の豊かさも守ろう	14.3%	3.0%	2.5%	9.3%
平和と公平をすべての人に	17.1%	0.0%	3.8%	11.1%
パートナーシップで目標を達成しよう	19.0%	0.0%	5.0%	13.0%

外向性や開放性の高いクラスターである Cluster1、Cluster4は SDGs の認知も高く、それぞれの項目における興味関心度合いも高い結果となっています。また、協調性の低い Cluster2では認知度も興味関心度合いも低い結果となりました。ただし、この結果についてそのまま高い・低いを論じるのはあくまで参考です。というのは、Cluster2の方たちは回答者が33名であり、このような性格因子を持つ人達すべてを代表するわけではなく、ごく少数であるためです。また、この33名の方々も性別や年代、職業も様々であることが考えられます。

データサイエンスを駆使しデータから何かを見出すには、様々な視点から分析を行うことが求められます。そして、今回このクラスター分析に用いた K-means 法には「クラスターの分類結果が初期値の設定によって異なる」という問題もあります。機械学習の手法に潜む「癖や特徴」を理解することは、実際に分析を行ううえで欠かせません。それらを理解したうえで機械学習の手法を選択できるようになるためには、分析する課題（テーマ）に対する理解や、手法そのものに対する理解が求められるということを認識しておきましょう。

アソシエーション分析

アソシエーション分析は、ビジネスのマーケティングでよく使用される手法で、膨大なデータの中から統計的なパターンや、意味のある関連性を抽出するデータマイニング手法です。たくさんある商品の中から、商品間の関連性や同時性を見つけるのに有益といわれます。

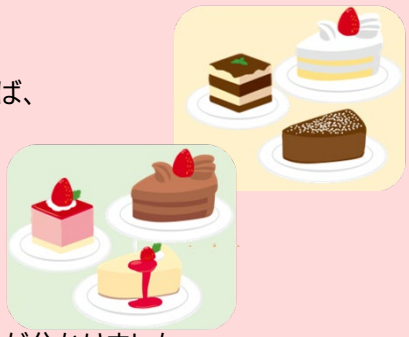
Case

一緒に買われるケーキはどれ？

美緒さんはある日、接客中にお客さんに聞かれました。
「こんなにチョコレート系のケーキばかり買う人、なかなかいないかしら？」
その時は「ええっと･･･そんなことないですよ。」とお茶を濁しましたが、そのときから、
本当はどうなのかずっと気になっています。

確かに、いろいろなジャンルのケーキをバランス良く買っていく人が多い気がするけど、
似た味のケーキばかり買う人もいないわけじゃない。フルーツ系ばかり買う人もいるし、
チーズケーキが好きなお客様も何人か覚えてる。
どの組み合わせが多いかなんて考えたこともなかったわ･･･。

美緒さんはショッピングサイトでよく見る「これを買った人はこれも気になっています。」
という機能を思い出しました。



良く買われる商品の組み合わせを知っていれば、接客のときに
「これとこれを一緒に買われる人が多いですよ。」なんておすすめすれば、
たくさん買ってもらえるかもしれない。

「人気の組み合わせランキングを作って店頭の黒板に書いたら、
通りすがりの人も見てくれるかもしれないわ。」

調べてみると、アソシエーション分析という手法があり
それらは、購買行動における商品間の関連性を見出す分析であることが分かりました。
「アソシエーション分析をしたら売上がさらにアップするかも！私にもできるかな･･･？」

「この商品を買った人は、こちらの商品も買っています」

インターネットの通販サイトなどで、「この商品を買った人はこんな商品も買っています」という表示を見たことはないでしょうか？　これらは、私たちの行動履歴やサイト内の購買データなどに基づいて、現在みている商品に関連したものをおすすめする「レコメンデーション」という仕組みです。レコメンデーションの構築にはアソシエーション分析が用いられています。

3 つの値:信頼度・支持度・リフト値

“If(もしこうであるなら)、then(こうなる)”というような相関ルールのことをアソシエーションルールといいます。データベースからこのアソシエーションルールを抽出するには何らかの評価指標が必要ですが、よく用いられている指標には信頼度・支持度・リフトなどがあります。

・信頼度(confidence)

信頼度とは、例えば、A を買った人のうち、どれくらいの人が B も買ったかという割合のことです。
例:ショートケーキを買った人のうち、どれくらいの人がプリンも買ったか?

ショートケーキを購入した人:100 人、プリンを買った人 60 人、両方を買った人:30 人では、30÷100 で信頼度は 0.3 です。

・支持度(support)

支持度とは、例えば、A　と B がどのくらい一緒に売れているかという割合のことです。この指標が大きいほど全体の中でそのルールが出現する確率が高くなります。一般的に知られている支持度は一緒に売れた数を、全体の数で割るものです。

ショートケーキもプリンも両方を買った人:30 人、お店に来たすべての人が 200 人であるとき、30÷200 で支持度は 0.15 です。

・リフト(lift)

リフト値とは、例えば、A と B を一緒に買った人の割合は、全体の中で B を買った人の割合よりどれだけ多いかを倍率で示したもののことです。リフト値が低い場合、A と B の関連性にあまり意味はなく、一般的にリフト値が1以上の場合は有効なルールとみなされています。
例:ショートケーキとプリンを一緒に買った人の割合は、全体の中でプリンを買った人の割合の何倍である

例題で機械学習を実行しよう（Python を用いた機械学習の実行）

▾ アソシエーション分析

```
[21] df = pd.read_csv("association.csv", index_col=0)
     df.head()
```

CustomerCD	PurchaseDate	Item	Amount	Number
Customer0000208	2019/10/12	Macaroons	530	2
Customer0000422	2019/11/22	Macaroons	530	1
Customer0000422	2020/5/19	Macaroons	530	2
Customer0000675	2019/11/6	Macaroons	530	3
Customer0000675	2019/10/8	Rare cheesecake	380	3

ここからデータを加工していきます。

```
[22] # CustomerCDとItemをキーに商品個数を集計する
     w1 = df.groupby(['CustomerCD', 'Item'])['Number'].sum()

     w1.head()
```

```
CustomerCD       Item
Customer0000001  Chocolate cake      8
                 Financier          2
                 Gateau chocolat    4
                 Macaroons          8
                 Pudding           12
Name: Number, dtype: int64
```

```
[23] # 商品番号を列に移動 (unstack関数の利用)
     w2 = w1.unstack().reset_index().fillna(0).set_index('CustomerCD')

     w2.head()
```

Item	Apple Pie	Chocolate cake	Eclair	Financier	Gateau chocolat	Macaroons	Pudding	Rare cheesecake	Shortcake	cookie	cream puff	jelly
CustomerCD												
Customer0000001	0.0	8.0	0.0	2.0	4.0	8.0	12.0	3.0	40.0	0.0	0.0	4.0
Customer0000003	0.0	0.0	4.0	0.0	0.0	1.0	0.0	0.0	12.0	0.0	0.0	0.0
Customer0000005	0.0	1.0	2.0	0.0	0.0	9.0	12.0	1.0	73.0	0.0	5.0	10.0
Customer0000009	0.0	6.0	0.0	0.0	3.0	5.0	0.0	5.0	31.0	0.0	2.0	0.0
Customer0000015	20.0	13.0	6.0	14.0	8.0	1.0	3.0	0.0	61.0	0.0	3.0	0.0

[24]
```
# 集計結果が正の場合True、0の場合Falseとする
basket_df = w2.apply(lambda x: x>0)

basket_df.head()
```

Item	Apple Pie	Chocolate cake	Eclair	Financier	Gateau chocolat	Macaroons	Pudding	Rare cheesecake	Shortcake	cookie	cream puff	jelly
CustomerCD												
Customer0000001	False	True	False	True	True	True	True	True	True	False	False	True
Customer0000003	False	False	True	False	False	True	False	False	True	False	False	False
Customer0000005	False	True	True	False	False	True	True	True	True	False	True	True
Customer0000009	False	True	False	False	True	True	False	True	True	False	True	False
Customer0000015	True	True	True	True	True	True	True	False	True	False	True	False

モデル構築

アプリオリ分析という手法で、Support（支持度）を求め、Supportの値が高いリストを抽出します。

[25]
```
# ライブラリの読み込み
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules

# アプリオリによる分析
freq_items1 = apriori(basket_df, min_support = 0.06, use_colnames = True)

# Supportが高い順に表示
freq_items1.sort_values('support', ascending = False).head()
```

	support	itemsets
8	0.839688	(Shortcake)
6	0.269637	(Pudding)
1	0.241617	(Chocolate cake)
18	0.206247	(Pudding, Shortcake)
0	0.188792	(Apple Pie)

Confidence（信頼度）やLift（リフト）を求めます。リフトは、Support（支持度）が、あまり売れていない商品をどう扱うかのための指標であったのと逆に、売れ筋商品をどう扱うかという指標です。

今回は、リフト値が高い関係を抽出しています。

[26]
```
# アソシエーションルールの抽出
a_rules1 = association_rules(freq_items1, metric = "lift", min_threshold = 1)

# リフト値が高い順にソート
a_rules1 = a_rules1.sort_values('lift', ascending = False).reset_index(drop=True)

a_rules1.head()
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Apple Pie)	(Chocolate cake)	0.188792	0.241617	0.091870	0.486618	2.014006	0.046254	1.477230
1	(Chocolate cake)	(Apple Pie)	0.241617	0.188792	0.091870	0.380228	2.014006	0.046254	1.308882
2	(Apple Pie)	(Chocolate cake, Shortcake)	0.188792	0.186495	0.069821	0.369830	1.983052	0.034612	1.290929
3	(Chocolate cake, Shortcake)	(Apple Pie)	0.186495	0.188792	0.069821	0.374384	1.983052	0.034612	1.296655
4	(Chocolate cake)	(Apple Pie, Shortcake)	0.241617	0.162150	0.069821	0.288973	1.782139	0.030643	1.178367

結果の解釈

アプリオリアルゴリズムを用いたアソシエーション分析の結果、下記のような購買パターンを発見することができました。

antecedents	consequents	antecedent support	consequent support	support	confidence	lift
(Apple Pie)	(Chocolate cake)	0.19	0.24	0.09	0.49	2.01
(Chocolate cake)	(Apple Pie)	0.24	0.19	0.09	0.38	2.01
(Apple Pie)	(Chocolate cake, Shortcake)	0.19	0.19	0.07	0.37	1.98
(Chocolate cake, Shortcake)	(Apple Pie)	0.19	0.19	0.07	0.37	1.98
(Chocolate cake)	(Apple Pie, Shortcake)	0.24	0.16	0.07	0.29	1.78
(Apple Pie, Shortcake)	(Chocolate cake)	0.16	0.24	0.07	0.43	1.78
(Shortcake)	(Eclair)	0.84	0.14	0.14	0.16	1.15
(Eclair)	(Shortcake)	0.14	0.84	0.14	0.97	1.15
(Rare cheesecake)	(Shortcake)	0.10	0.84	0.09	0.95	1.13
(Shortcake)	(Rare cheesecake)	0.84	0.10	0.09	0.11	1.13
(Macaroons)	(Shortcake)	0.09	0.84	0.08	0.95	1.13
(Shortcake)	(Macaroons)	0.84	0.09	0.08	0.10	1.13
(cream puff)	(Shortcake)	0.10	0.84	0.09	0.94	1.12
(Shortcake)	(cream puff)	0.84	0.10	0.09	0.11	1.12
(jelly)	(Shortcake)	0.09	0.84	0.08	0.92	1.10
(Shortcake)	(jelly)	0.84	0.09	0.08	0.10	1.10
(Financier)	(Shortcake)	0.08	0.84	0.07	0.90	1.07
(Shortcake)	(Financier)	0.84	0.08	0.07	0.09	1.07
(Shortcake)	(Apple Pie)	0.84	0.19	0.16	0.19	1.02
(Apple Pie)	(Shortcake)	0.19	0.84	0.16	0.86	1.02
(Shortcake)	(Gateau chocolat)	0.84	0.11	0.09	0.11	1.02
(Gateau chocolat)	(Shortcake)	0.11	0.84	0.09	0.86	1.02

今回はリフトが 1 よりも大きいもののみ算出しています。Confidence(信頼度)が最も高いのは、エクレアとショートケーキ、次いでレアチーズケーキとショートケーキ、シュークリーム(Cream puff)とショートケーキです。Support が最も高いのは 0.16 のショートケーキとアップルパイ同士の組み合わせです。Lift がもっとも高いのはアップルパイとショートケーキです。今回の学習では、お客さんごとの購入を1つの算出の単位としました。一緒に買われているものを算出するには、レシート単位で算出する方法も検討できそうです。アソシエーション分析は「一緒に購入した」単位を何に設定するかによって、見いだせる結果が異なることも抑えておきましょう。

数量化理論(III 類、IV 類)

数量化理論については「教師あり学習の手法と活用例」の数量化 I 類・II 類における解説で述べたように、数量化 I 類から IV 類がよく使われますが、ここでは数量化 III 類と数量化 IV 類の説明をします。

数量化 I 類と II 類が、機械学習において教師あり学習に分類されるのに対し、数量化 III 類と IV 類は目的変数がないデータを分析するので、「教師なし学習」に分類され、多変量解析では要約の手法にあたります。

数量化 III 類の分析アルゴリズムは、目的変数が量的変数であるコレスポンデンス分析に対応し、数量化 IV 類は MDS（多次元尺度構成法）に対応しています。

表：要約の手法と活用例

変数	**手法**
量的変数	主成分分析、因子分析、クラスター分析
	・白米の消費量と身長の関係を調べる ・各国の輸出額と GDP の関係を調べる
質的変数	**数量化 III 類**、コレスポンデンス分析、**数量化 IV 類**、MDS（多次元尺度構成法）
	・好きな音楽のジャンルとファッションの関係を調べる ・性別・年代と好きな YouTube チャンネルの関係をプロットする

注）数量化 IV 類は量的変数にも対応しています。

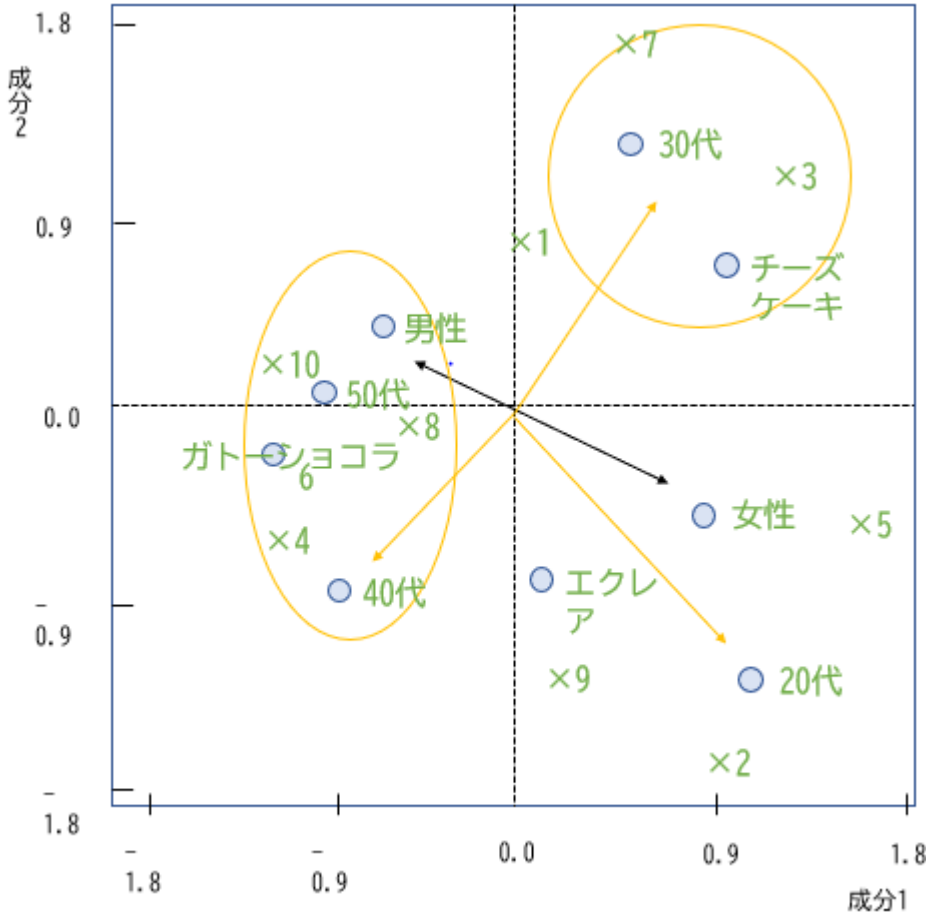
数量化 III 類は、以下の表のように何人かの属性データと好きなスイーツのデータがあった時に、まずはそのデータをカテゴリーデータに変換し、その嗜好パターンからAさんとBさんの好みが似ているか似ていないかという類似度とスイーツ P とスイーツQの好みの類似度を計算することでポジショニングマップを作成することができます。

表　変換前のデータ（2）

No.	性別	年代	好きなお酒
1	男性	30代	エクレア
2	女性	20代	エクレア
3	女性	30代	チーズケーキ
4	男性	40代	ガトーショコラ
5	女性	20代	チーズケーキ
6	男性	40代	ガトーショコラ
7	男性	30代	チーズケーキ
8	男性	50代	エクレア
9	女性	40代	エクレア
10	男性	50代	ガトーショコラ

表： 変換後のデータ(2)

No.	女性	男性	20代	30代	40代	50代	ガトーショコラ	チーズケーキ
1	0	1	0	1	0	0	0	0
2	1	0	1	0	0	0	0	0
3	1	0	0	1	0	0	0	1
4	0	1	0	0	1	0	1	0
5	1	0	1	0	0	0	0	1
6	0	1	0	0	1	0	1	0
7	0	1	0	1	0	0	0	1
8	0	1	0	0	0	1	0	0
9	1	0	0	0	1	0	0	0
10	0	1	0	0	0	1	1	0



図：ポジショニングマップ

具体的な類似性の計算式はここでは述べませんが、イメージとしては、図の変換後のデータの表側と表頭をうまく並べ替えることで、似た者同士がなるべく近くに配置されるような組み合わせを考えようという発想です。

表　数量化 III 類の計算アルゴリズムイメージ

No.	20代	チーズケーキ	女性	30代	エクレア	40代	ガトーショコラ	男性
8	0	0	0	0	1	0	0	1
10	0	0	0	0	0	0	1	1
6	0	0	0	0	0	1	1	1
4	0	0	0	0	0	1	1	1
1	0	0	0	1	1	0	0	1
9	0	0	1	0	1	1	0	0
7	0	1	0	1	0	0	0	1
3	0	1	1	1	0	0	0	0
5	1	1	1	0	0	0	0	0
2	1	0	1	0	1	0	0	0

数量化 IV 類は、類似度のデータからポジショニングマップを作る時に使われます。類似度データは、表の類似度マトリクスと言われる相関行列表形式のデータで、表側と表頭が同じ項目が並んでいて、その項目同士の類似度がアンケートデータの 5 段階評価のような順序尺度（質的変数）や相関係数の比尺度（量的変数）で表わされています。

表：　類似度マトリクス

	1	2	3	4	5	6	7	8	9	10
1エクレア	10									
2ゼリー	9	10								
3モンブラン	6	7	10							
4シュークリーム	7	9	8	10						
5フィナンシェ	5	6	8	8	10					
6マカロン	2	3	6	3	6	10				
7レアチーズケーキ	2	3	5	4	7	6	10			
8クッキー	1	2	4	3	5	5	9	10		
9ショートケーキ	1	1	2	1	3	3	7	8	10	
10ガトーショコラ	2	3	3	4	5	2	5	5	4	10

類似度マトリクスデータが与えられた時に、似ているものが近くに、似ていないものが遠くにプロットされるようなポジショニングマップを作ることができます。

図： ポジショニングマップ