第4章の内容

1. 回帰分析による予想
2. 決定木による予想
3. クラスター分析
   – その1 –
   – その2 –

161

1.回帰分析による予想

回帰分析とは

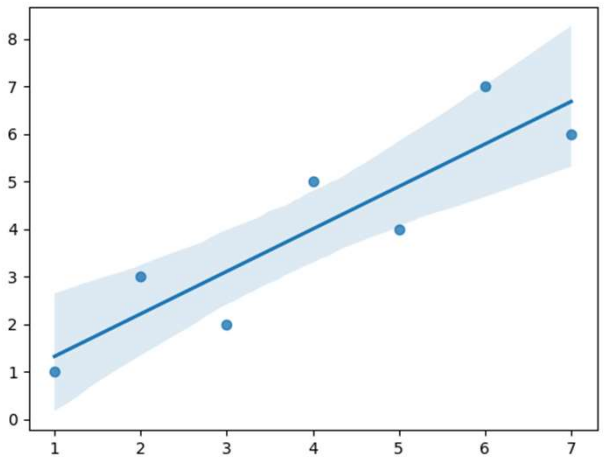
- 因果関係が疑われる複数の変数を使って，ある変数から他の変数の値を予測する手法
- 原因となる変数のことを説明変数，結果となる変数のことを応答変数という
- 説明変数と応答変数はそれぞれ独立変数と従属変数という
- 説明変数が数量データであるモデルを用いた分析手法
  （説明変数が複数あるものを重回帰分析と呼ぶ）

162

1.回帰分析による予想

回帰分析

```
import seaborn as sns
x = [1, 2, 3, 4, 5, 6, 7]
y = [1, 3, 2, 5, 4, 7, 6]
sns.regplot(x=x, y=y)
```
散布図



163

1.回帰分析による予想

回帰分析

```
!pip install pingouin
```
Pingouin（ピングイン）というパッケージをインストール

```
import pingouin as pg
pg.corr(x, y)
```

R:相関係数

	n	r	CI95%	p-val	BF10	power
pearson	7	0.892857	[0.43, 0.98]	0.006807	8.961	0.854032

p値

95%信頼区間

```
pg.linear_regression(x, y)
```

	names	coef	se	T	pval	r2	adj_r2	CI[2.5%]	CI[97.5%]
0	Intercept	0.428571	0.900680	0.475831	0.654258	0.797194	0.756633	-1.886700	2.743843
1	x1	0.892857	0.201398	4.433293	0.006807	0.797194	0.756633	0.375147	1.410568

傾き　　p値　R^2（相関係数の二乗）　95%信頼区間

164

1.回帰分析による予想

回帰分析と結果の可視化

既知のデータ
sales_data.csv

	A	B
1	temperature	sales
2	4.7	507
3	5.4	416
4	11.5	607
5	17	746
6	19.8	894
7	22.4	1021
8	28.3	1506
9	28.1	1443
10	22.9	861
11	19.1	640
12	14	492
13	8.3	537
14	5.6	494
15	7.2	423
16	10.6	542
17	13.6	667
18	20	1000
19	21.8	991
20	24.1	1236
21	28.4	1513
22	25.1	996
23	19.4	724
24	13.1	531
25	8.5	584
26	7.1	510
27	8.3	482
28	10.7	610

予測対象データ
sales_future.csv

	A	B
1	temperature	
2	12.6	
3	18.7	
4	10.2	
5	15.1	
6	20.5	
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		
25		
26		
27		

sales_future

一般社団法人　日本アイスクリーム協会
（https://www.icecream.or.jp/about/）と
国土交通省気象庁
（https://www.data.jma.go.jp/obd/stats/etrn/view/monthly_s3.php?prec_no=44&block_no=47662）のデータを合わせて作成した

sales_data.csvに基づいて回帰分析を行い，
sales_future.csvに対して売上予測

165

1.回帰分析による予想

回帰分析と結果の可視化

```
from google.colab import files
uploaded = files.upload()
```

ファイルの読み込み等

```
import pandas as pd

df = pd.read_csv('sample_sales_data.csv')
df_test = pd.read_csv('sample_sales_future.csv')
```

166

1.回帰分析による予想

回帰分析と結果の可視化

```
x_name = "temperature"        xとyに分離
y_name = "sales"
x = df[x_name]
y = df[y_name]

import statsmodels.api as sm   回帰分析の実行
model = sm.OLS(y, sm.add_constant(x))
result = model.fit(disp=0)
print(result.summary())
```

求めたいもの（目的変数）が「y」で，その計算に使うもの（説明変数）が「x」
「add_constant」は切片を使う場合に指定
つまり，y=ax + b のbの項を使う場合に付ける

167

1.回帰分析による予想

回帰分析と結果の可視化

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  sales   R-squared:                       0.832
Model:                            OLS   Adj. R-squared:                  0.829
Method:                 Least Squares   F-statistic:                     287.1
Date:                Fri, 17 Nov 2023   Prob (F-statistic):           3.96e-24
Time:                        08:27:53   Log-Likelihood:                -380.86
No. Observations:                  60   AIC:                             765.7
Df Residuals:                      58   BIC:                             769.9
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        155.0243     44.387      3.493      0.001      66.175     243.874
temperature   41.4521      2.447     16.943      0.000      36.555      46.349
==============================================================================
Omnibus:                        0.628   Durbin-Watson:                   1.324
Prob(Omnibus):                  0.731   Jarque-Bera (JB):                0.706
Skew:                           0.048   Prob(JB):                        0.703
Kurtosis:                       2.477   Cond. No.                         44.5
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

coefとconstが交わった所が切片
「155.0243」

R-squaredがR^2値
「0.832」

coefとtemperatureが交わった所が傾き「41.4521」

アイスクリームの売上をy,
気温をx,
y = 41.45x + 155.02
という関係があるっぽい

168

1.回帰分析による予想

回帰分析と結果の可視化（未知のデータの推測）

```
y_result=[]
for i in range(len(df_test.index)):
    y_tmp = result.params.const + result.params[x_name] *
df_test[x_name][i]
    y_result.append(y_tmp)
print(y_result)
```

Xの傾きが格納

切片が格納

```
[677.3210908376445, 930.1790552581247, 577.8359900820456, 780.9514041247264, 1004.7928808248239]
```

169

1.回帰分析による予想

回帰分析と結果の可視化（未知のデータの推測）

```
df_test_y = pd.DataFrame(y_result,columns=["y"])
df_result = pd.concat([df_test,df_test_y],axis=1)
df_result.to_csv("result.csv")
df_result
```

	temperature	y
0	12.6	677.321091
1	18.7	930.179055
2	10.2	577.835990
3	15.1	780.951404
4	20.5	1004.792881

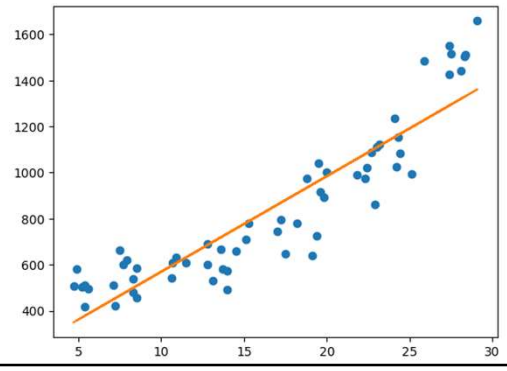
読み込んだファイル（sales_future.csv）に計算結果（y列）を追記し，
csvファイルとして出力することも可能

170

1.回帰分析による予想

回帰分析と結果の可視化（プロット）

```
import matplotlib.pyplot as plt
plt.plot(x, y, 'o')
plt.plot(x, result.params.const+result.params[x_name]*x)
plt.show()
```



そこそこ良い感じの分析精度

171

2.決定木による予想

決定木とは（Decision Tree Analysis:DCA）

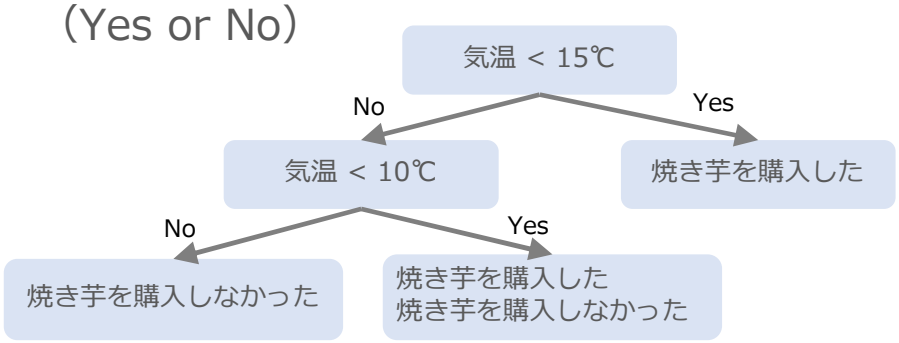
- データから分類・判別のために作られる決定木と呼ばれる樹形図を作成し，予測や検証をする分析（機械学習の1つ）
- 購買情報やアンケート結果等のさまざまなデータに対して実施することが可能
- 目的変数の予測や目的変数に影響している因子の検証等に活用

172

2.決定木による予想

決定木分析と回帰分析の違い

- どちらも目的変数を予想するモデル、ただプロセスが違う
- 計算式などを使わずにシンプルな分岐のみで予想
  （Yes or No）



173

2.決定木による予想

流れ

- 事前準備
- 決定木の前処理
  - 説明変数（x）と目的変数（y）に分割
  - ダミー変数処理（文字列→数値）
  - 学習用-テスト用に分割
- 決定木モデルの作成と予想
  - パラメータ：max_depth（最大深度）
- 決定木の可視化（plot_tree）
  - ツリーの見方
  - パラメータ：説明変数の名前（feature_names）
  - パラメータ：目的変数の名前（class_names）
  - パラメータ：色（filled）

174

2.決定木による予想

事前準備（タイタニック号データ）

```
import seaborn as sns
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import tree
df = sns.load_dataset('titanic')
df.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

175

2.決定木による予想

事前準備（タイタニック号データ）

survived	生存フラグ
pclass	チケットクラス
sex	性別
age	年齢
sibsp	兄弟・配偶者の数
parch	親・子の数
fare	料金

embarked	出航地
class	チケットクラス
who	性別
adult_male	成人男性かどうか
deck	乗船していたデッキ
embark_town	出航地名
alive	生存
alone	一人だったか

176

2.決定木による予想

決定木の前処理：説明変数（x）と目的変数（y）に分割

```
df_x = df[['sex','pclass','fare']]
df_y = df['survived']
```

事前に定義したタイタニック号のデータの変数（df）から，説明変数（df_x）と目的変数（df_y）に分ける

決定木の可視化を見やすくするために，性別・チケットクラス・運賃に限定

177

2.決定木による予想

決定木の前処理：ダミー変数処理（文字列→数値）

```
df_x = pd.get_dummies(df_x, drop_first=True)
```

drop_first カラム数を減らすパラメータ

性別のカラムに「male / female」の文字列が格納されているので，数値列に変換
get_dummies

	pclass	fare	sex_male
0	3	7.2500	1
1	1	71.2833	0
2	3	7.9250	0
3	1	53.1000	0
4	3	8.0500	1

178

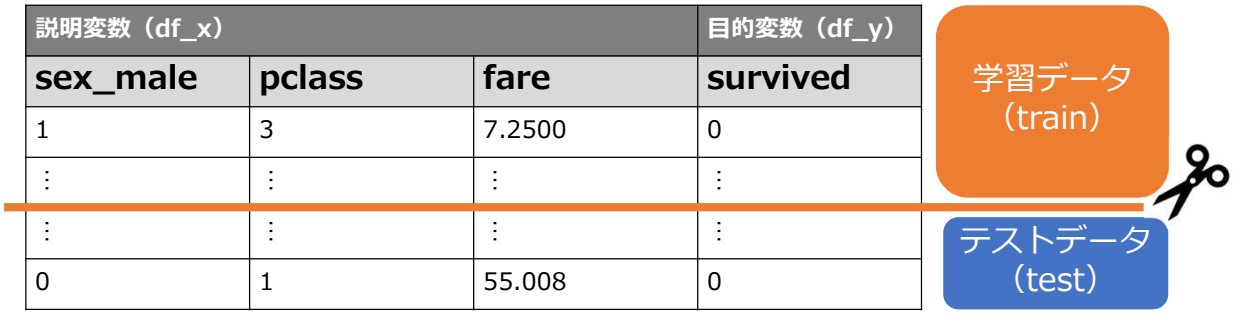
2.決定木による予想

決定木の前処理：学習用-テスト用に分割

```
from sklearn.model_selection import train_test_split
train_x, test_x, train_y, test_y =
train_test_split(df_x,df_y,random_state=1)
```

学習用とテスト用にデータを分割
train_test_split

説明変数（df_x）			目的変数（df_y）
sex_male	**pclass**	**fare**	**survived**
1	3	7.2500	0
⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮
0	1	55.008	0



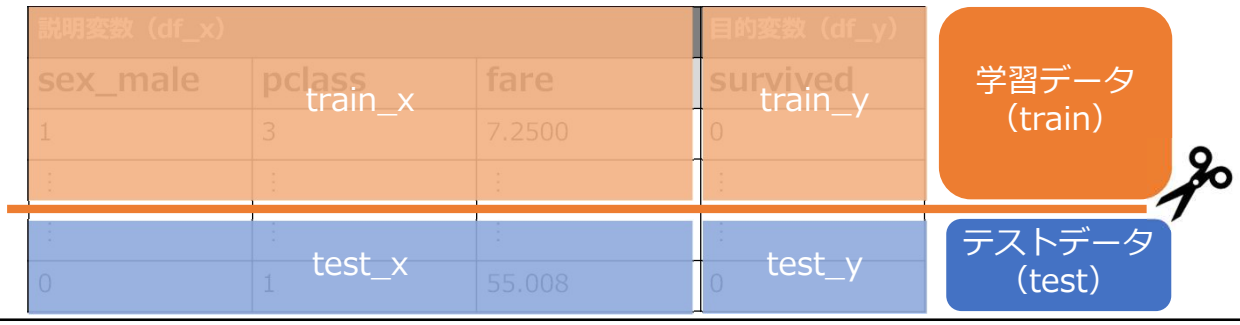
179

2.決定木による予想

決定木の前処理：学習用-テスト用に分割

```
from sklearn.model_selection import train_test_split
train_x, test_x, train_y, test_y =
train_test_split(df_x,df_y,random_state=1)
```

学習用とテスト用にデータを分割
train_test_split



180

2.決定木による予想

決定木モデルの作成と予測

```
from sklearn import tree
model = tree.DecisionTreeClassifier(max_depth=2, random_state=1)
```

決定木のモデルを作成
scikit-learnというpython用機械学習ライブラリの中から決定木モデルのtreeを拝借
sklearn.tree.DecisionTreeClassifierというクラスに決定木が実装されている
max_depth：分類技の最大深さ
random_state：学習時の乱数シード，常に同じ結果を得たい場合は適当な整数を指定

181

2.決定木による予想

決定木モデルの作成と予測

```
model.fit(train_x, train_y)
```

fitメソッドで学習を行う

fit(x, y)：特徴量 x, クラス y を教師データとして学習する

182

2.決定木による予想

決定木モデルの作成と予測

model.predict(test_x)　　モデルに対してpredictfitで予想を実施

predict (x)：特徴量 x に対するクラスの予想結果を返す

```
array([1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 0])
```

183

2.決定木による予想

決定木モデルの作成と予測

model.score(test_x,test_y)　　scoreメソッドでスコア（正解率）を算出

score (x, y)：決定係数を出力，予想値 xと正解値 yの相関を測る

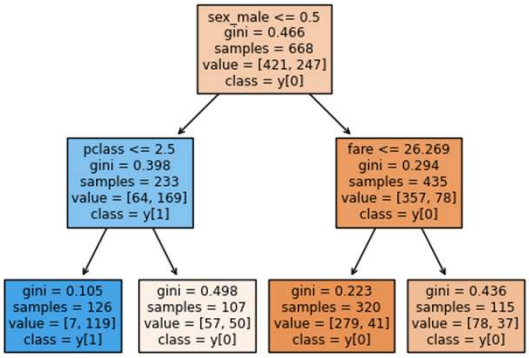
```
0.7533632286995515
```

184

2.決定木による予想

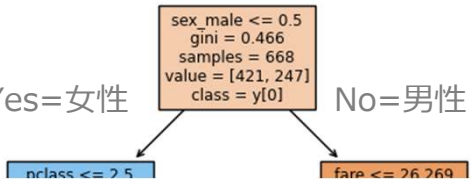
決定木の可視化（plot_tree）

```
from sklearn.tree import plot_tree
plot_tree(model, feature_names=train_x.columns, class_names=True,
filled=True)
```



性別（男性=1）が0.5より小さい

Yes=女性　No=男性

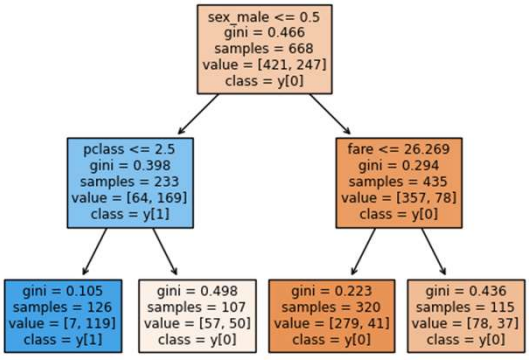


185

2.決定木による予想

決定木の可視化（plot_tree）

```
from sklearn.tree import plot_tree class_name：目的変数の名前
plot_tree(model, feature_names=train_x.columns, class_names=True,
filled=True)                                          filled：色
```



gini（ジニ係数）：不純度
各葉（ノード）にどれくらい間違いが含まれているのか
不純度が低い（正しい）と値は「0」
不純度が高い（間違い）と「1」に近づく

色合い：濃いほど目的変数に近い
（ジニ係数も低い）
sample：ノードの個数
value：その条件に当てはまる数

186

3.クラスター分析

クラスター分析とは（クラスタリング）

- 個々のデータから似ているデータ同士をグルーピングする分析手法
- クラスタ（群）ごとに分けることで，データの中身を理解しやすくしたり，群ごとに施策を行うことができる

187

3.クラスター分析

流れ

- 事前準備
- k-meansの前処理
  - データを限定する
  - 欠損値を処理する
  - ダミー変数処理（文字列→数値）
  - データを標準化する
- k-meansのクラスタリングを実行
- クラスタリングの結果を確認
  - 主成分分析でグラフ化

188

3.クラスター分析

事前準備（タイタニック号データ）

```
import seaborn as sns
import pandas as pd
df = sns.load_dataset('titanic')
df.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

189

3.クラスター分析

事前準備（タイタニック号データ）

survived	生存フラグ
pclass	チケットクラス
sex	性別
age	年齢
sibsp	兄弟・配偶者の数
parch	親・子の数
fare	料金

embarked	出航地
class	チケットクラス
who	性別
adult_male	成人男性かどうか
deck	乗船していたデッキ
embark_town	出航地名
alive	生存
alone	一人だったか

190

3.クラスター分析

k-meansの前処理：データを限定する

```
columns = ['survived','pclass','sex','age','fare']
df = df[columns]
```

生存・チケットクラス・性別・年齢・運賃に絞り込む

191

3.クラスター分析

k-meansの前処理：欠損値を処理する

```
df.isnull().sum()
```
isnull と sum の組み合わせ

年齢に177件の欠損を確認，これを今回は平均で補う

```
survived    0
pclass      0
sex         0
age       177
fare        0
dtype: int64
```

```
mean = df['age'].mean()
df['age'] = df['age'].fillna(mean)
```
fillna を使う

192

3.クラスター分析

k-meansの前処理：ダミー変数処理（文字列→数値）

```
df = pd.get_dummies(df, drop_first=True)
df.head()
```

isnull と sum の組み合わせ

データに文字列が含まれているのでダミー変数処理を行う

	survived	pclass	age	fare	sex_male
0	0	3	22.0	7.2500	1
1	1	1	38.0	71.2833	0
2	1	3	26.0	7.9250	0
3	1	1	35.0	53.1000	0
4	0	3	35.0	8.0500	1

193

3.クラスター分析

k-meansの前処理：データの標準化

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
df_sc = sc.fit_transform(df)
df_sc = pd.DataFrame(df_sc, columns=df.columns)
df_sc.head()
```

StandardScalerを使用

年齢は22・26…，チケットクラスは1・3…，スケールの違いを合わせる：標準化

	survived	pclass	age	fare	sex_male
0	-0.789272	0.827377	-0.592481	-0.502445	0.737695
1	1.266990	-1.566107	0.638789	0.786845	-1.355574
2	1.266990	0.827377	-0.284663	-0.488854	-1.355574
3	1.266990	-1.566107	0.407926	0.420730	-1.355574
4	-0.789272	0.827377	0.407926	-0.486337	0.737695

194

3.クラスター分析

k-meansのクラスタリングを実行

```
from sklearn.cluster import KMeans
model = KMeans(n_clusters=4, random_state=1)
model.fit(df_sc)
```

n_clustersは分割するクラスタ数

sklearnからKMeansをインポートし，モデルを作成

```
cluster = model.labels_
cluster
```

labels_ でクラスタ番号を取得

```
array([3, 1, 2, 1, 3, 3, 0, 3, 2, 2, 2, 1, 3, 3, 2, 2, 3, 0, 2, 2, 0, 0,
       2, 0, 2, 2, 3, 1, 2, 3, 0, 1, 2, 0, 0, 0, 3, 3, 2, 2, 2, 2, 3, 2,
       2, 3, 3, 2, 3, 2, 3, 3, 1, 2, 0, 0, 2, 3, 2, 3, 3, 1, 0, 3, 0, 3,
       2, 3, 2, 3, 3, 2, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 2, 0, 2, 2, 3, 3,
       1, 3, 3, 3, 0, 3, 0, 3, 0, 0, 2, 3, 2, 3, 0, 3, 3, 3, 2, 3, 3, 2,
       0, 2, 3, 2, 2, 3, 0, 3, 1, 2, 3, 3, 3, 2, 0, 3, 3, 3, 2, 3, 3, 3,
```

195

3.クラスター分析

k-meansのクラスタリングを実行

```
df['cluster'] = cluster
df
```

DataFrame の新しいカラムに
クラスタ番号を追加

	survived	pclass	age	fare	sex_male	cluster
0	0	3	22.000000	7.2500	1	3
1	1	1	38.000000	71.2833	0	1
2	1	3	26.000000	7.9250	0	2
3	1	1	35.000000	53.1000	0	1
4	0	3	35.000000	8.0500	1	3
...	...	...	...	...	...	...
886	0	2	27.000000	13.0000	1	3
887	1	1	19.000000	30.0000	0	2

196

3.クラスター分析

クラスタリングの結果を確認

```
# style.bar で DataFrame にカラーバーを追加
df.groupby('cluster').mean().style.bar(axis=0)
```

読み込ませたデータの平均値をクラスタごとに確認

cluster	survived	pclass	age	fare	sex_male
0	0.280000	1.320000	42.981210	37.686277	0.986667
1	0.925926	1.009259	33.376021	123.153357	0.148148
2	0.650000	2.645455	25.380733	18.052823	0.000000
3	0.138015	2.828087	26.213949	13.968097	1.000000

197

3.クラスター分析

クラスタリングの結果を確認

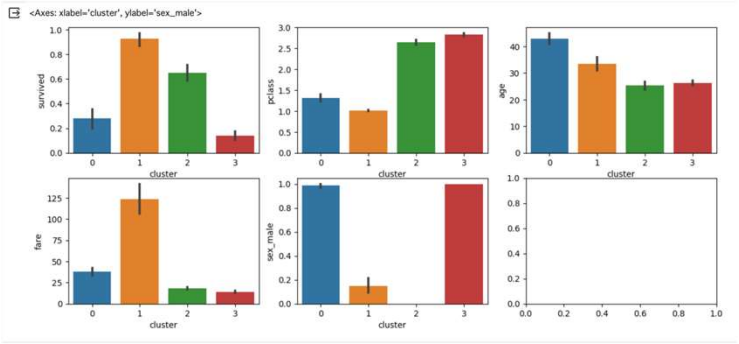
```
import matplotlib.pyplot as plt
fig, axes = plt.subplots(2,3, figsize=(14, 6))
sns.barplot(ax=axes[0,0], data=df, x='cluster', y='survived')
sns.barplot(ax=axes[0,1], data=df, x='cluster', y='pclass')
sns.barplot(ax=axes[0,2], data=df, x='cluster', y='age')
sns.barplot(ax=axes[1,0], data=df, x='cluster', y='fare')
sns.barplot(ax=axes[1,1], data=df, x='cluster', y='sex_male')
```

198

3.クラスター分析

クラスタリングの結果を確認

クラスタ0：男性で高齢，良い客室に泊まっており，生存率が低い
クラスタ1：女性で良い客室に泊まり運賃が最も高く，生存率も一番高い
クラスタ2：女性で安い客室に泊まっている
クラスタ3：男性で若年，安い客室に泊まっており，一番生存率が低い



199

3.クラスター分析

クラスタリングの結果を確認：主成分分析でグラフ化

```
df_sc['cluster'] = cluster
```

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2, random_state=1)
pca.fit(df_sc)
feature = pca.transform(df_sc)
feature
```

PCAをインポートし学習させる

```
array([[-1.79546026,  0.2666309 ],
       [ 2.61164075, -0.78970203],
       [ 0.02143814, -1.73958101],
       ...,
       [-0.43069953, -0.47767583],
       [ 2.14671007,  0.27131272],
       [-1.59197757,  0.65939251]])
```

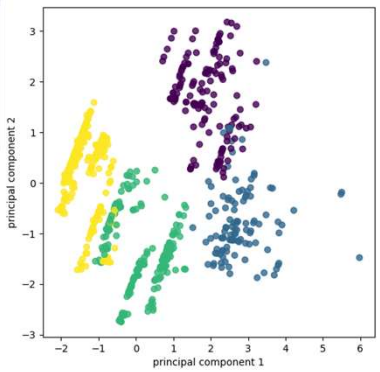
200

3.クラスター分析

クラスタリングの結果を確認：主成分分析でグラフ化

```
import matplotlib.pyplot as plt
plt.figure(figsize=(6, 6))
plt.scatter(feature[:, 0], feature[:, 1], alpha=0.8,
c=cluster)
plt.xlabel('principal component 1')
plt.ylabel('principal component 2')
plt.show()
```

散布図で可視化



201

3.クラスター分析（その2）

階層的クラスタリング（Hierarchical Clustering）

- 最も類似する（または最も類似しない）サンプルデータの組み合わせを見つけ出し，順番にグループ分けしていく手法
- 樹形図（デンドログラム）をプロットできる
- 二分木で階層的クラスタリングを可視化したもの

202

3.クラスター分析（その2）

計算の流れ

- 各サンプルを単一のクラスターとみなし，全てのクラスター間のユークリッド距離を計算
- クラスター間の距離に基づき，クラスターを連結
- クラスター情報を更新し，ユークリッド距離を再計算
- クラスターが最終的に１つになるまで繰り返す

203

3.クラスター分析（その2）

流れ

- 事前準備
- 最適なクラスターの数を見つけるための樹形図の作成
- クラスター数の決定・モデル学習
- クラスター可視化

204

3.クラスター分析（その2）

事前準備（SSDSE：教育用標準データセット）

```
from google.colab import files
uploaded = files.upload()
```

```
import pandas as pd
df = pd.read_csv('sample_7.csv')
df.head()
```

サンプルデータをインポート

統計センターのSSDSE（教育用標準データセット）を利用
https://www.nstac.go.jp/use/literacy/ssdse/

205

3.クラスター分析（その2）

事前準備（SSDSE：教育用標準データセット）

```
from google.colab import files
uploaded = files.upload()
```

```
import pandas as pd
df = pd.read_csv('sample_7.csv')
df.head()
```

サンプルデータをインポート

統計センターのSSDSE（教育用標準データセット）を利用
https://www.nstac.go.jp/use/literacy/ssdse/

	Unnamed: 0	総人口	幼稚園の数(10万人あたり)	残業時間の平均(時間/月)	科学技術振興費の研究費(万円/10万人あたり)	情報通信業の企業数
0	北海道	5250000	7.695238	12.0	95.238095	143
1	青森県	1246000	7.062600	10.5	0.000000	30
2	岩手県	1227000	7.497963	11.5	2086.389568	36
3	宮城県	2306000	10.320902	12.5	1036.426713	51
4	秋田県	966000	4.037267	9.0	5621.118012	16

206

3.クラスター分析（その2）

事前準備（SSDSE：教育用標準データセット）

```
import numpy as np
df.describe()
```

基本統計量を確認

	総人口	幼稚園の数（10万人あたり）	残業時間の平均（時間/月）	科学技術振興費の研究費（万円/10万人あたり）	情報通信業の企業数
count	4.700000e+01	47.000000	47.000000	47.000000	47.000000
mean	2.684404e+06	8.555025	11.553191	1584.176816	117.425532
std	2.779720e+06	3.126166	1.372217	1573.513577	419.861610
min	5.560000e+05	3.597122	8.500000	0.000000	9.000000
25%	1.075500e+06	6.671337	10.500000	807.490314	26.500000
50%	1.602000e+06	7.796773	11.500000	1142.857143	34.000000
75%	2.693500e+06	10.313550	12.500000	1660.102424	57.500000
max	1.392100e+07	16.758242	15.000000	7591.145833	2899.000000

207

3.クラスター分析（その2）

事前準備（SSDSE：教育用標準データセット）

```
import scipy.cluster.hierarchy as sch
# import matplotlib.pyplot as plt
```

クラスタリング軸設定　（人口vs残業時間の平均）

```
X = df.iloc[:, [1, 3]].values

dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward'))
```

樹形図作成用インスタンス

scipy.cluster.hierarchyクラスからインスタンスを生成することで、樹形図を作成
インスタンス内のlinkage()メソッドはデータ間の距離を計算しつながりを作る。
連結方法としてウォード連結法で距離を計算。

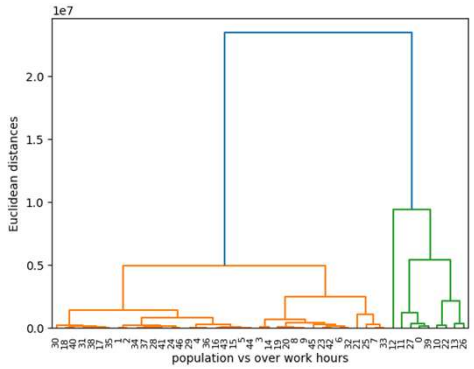
208

3.クラスター分析（その2）

最適なクラスターの数を見つけるための樹形図の作成

```
plt.xlabel('population vs over work hours')
plt.ylabel('Euclidean distances')
plt.show()
```

ラベルを貼りながら出力



この結果から２つのクラスターで
分割するのが良いかもしれない。

209

3.クラスター分析（その2）

モデル学習

```
from sklearn.cluster import AgglomerativeClustering
```

モデルの訓練

```
hir_clus = AgglomerativeClustering(n_clusters = 3, affinity = 'euclidean',
linkage = 'ward')
y_hir_clus = hir_clus.fit_predict(X)
```

第一引数：クラスター数（n_clusters）デフォルト値は2
第二引数：距離のパラメータ（affinity）
第三引数：クラスター連結法（linkage）

210

3.クラスター分析（その2）

モデル学習によって予想されたクラスターの結果

```
print(y_hir_clus)
```

```
[0 1 1 1 1 1 1 1 1 1 0 0 2 0 1 1 1 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 1 1
 1 1 0 1 1 1 1 1 1 1]
```

211

3.クラスター分析（その2）

クラスターの可視化

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm

cluster_labels = np.unique(y_hir_clus)
n_clusters = cluster_labels.shape[0]
```

クラスターの配列情報
一意なクラスター要素
配列の長さ

```
for i in range(len(cluster_labels)):
    color = cm.jet(float(i) / n_clusters)
    plt.scatter(X[y_hir_clus == i, 0], X[y_hir_clus == i, 1], s = 50, c = color, label =
'Cluster'+str(i))
```

可視化

```
plt.title('Clusters of population')
plt.xlabel('population')
plt.ylabel('over work hours')
plt.legend(loc="best")
plt.show()
```

グラフに関する情報

212

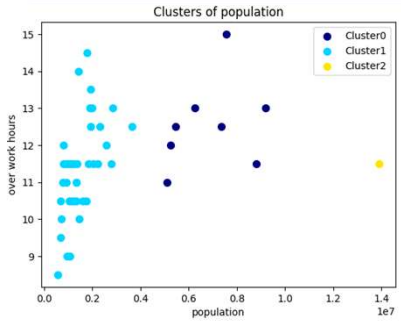
3.クラスター分析（その2）

クラスターの可視化

（以下再掲）

```
plt.title('Clusters of population')
plt.xlabel('population')
plt.ylabel('over work hours')
plt.legend(loc="best")
plt.show()
```

グラフに関する情報



213

参考文献（継続して学習したい方向け）

- 谷合廣紀（2018）
  『Pythonで理解する統計解析の基礎』技術評論社
- 塚本邦尊 (著), 山田典一 (著), 大澤文孝 (著), 中山浩太郎（監修）(その他), 松尾豊（監修）（2019）
  『東京大学のデータサイエンティスト育成講座 Pythonで手を動かして学ぶデータ解析』マイナビ出版
- 馬場真哉 (著)（2022）
  『Pythonで学ぶあたらしい統計学の教科書 第2版』翔泳社

214

データサイエンス・オンライン講座のご紹介

「誰でも使える統計オープンデータ」受講者募集中！

申込はこちらから！

講座の目的：e-Stat、jSTAT MAP、API機能等を使い、
統計オープンデータを活用したデータ分析の
基本的な知識を習得する
開 講 期 間：令和６年１月16日（火）～３月19日（火）
学 習 時 間：１回10分程度×５～７回程度（１週間）×４週
課　　題：各週の確認テストと最終課題の実施
講　　師：西内啓氏（統計家）ほか

週※5	各週のテーマ	内　容
1	e-Statを使ったデータ分析	e-Statの統計データを活用したデータ分析の事例、基本的な活用方法を学ぶ （e-Statの機能紹介、活用事例紹介等）
2	公的統計データの使い方	公的統計データの基本事項及び読み方を学ぶ （公的統計の種類と体系、労働力調査・家計調査の基礎知識及び利用の際のポイント等）
3	地図で見る統計（jSTAT MAP）の活用	統計データと地図を組み合わせた活用方法を学ぶ （地図で見る統計（jSTAT MAP）の機能紹介、簡単にできるレポート作成、活用事例紹介等）
4	統計オープンデータの高度利用	統計API機能の仕組みや具体的な活用事例等の統計オープンデータの高度な活用方法を学ぶ （統計APIの仕組み、統計オープンデータの活用事例、講座のまとめ等）

215