データ解析例（教師なし学習）

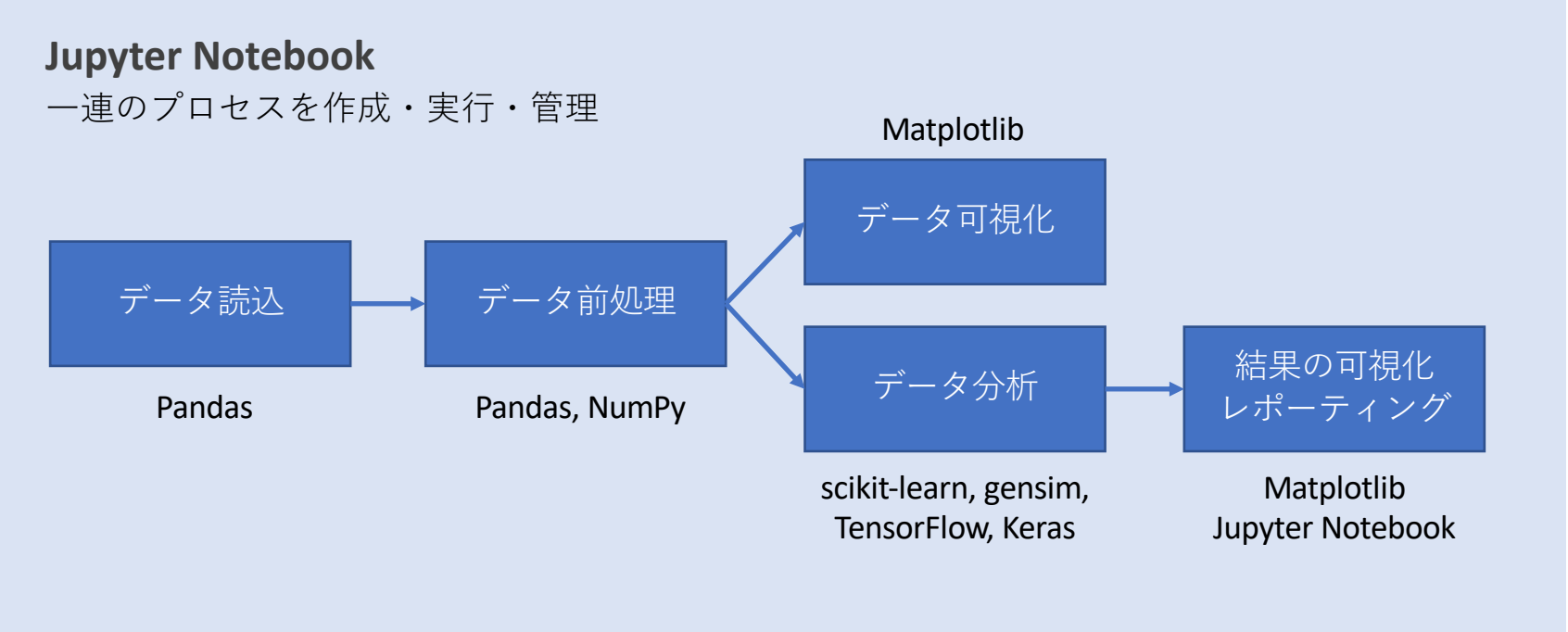
※バージョン情報
Python: 3.7.3, pandas: 0.24.2, scikit-learn: 0.20.3

Pythonとは

可読性の高いプログラミング言語でデータ分析で有用なライブラリや開発環境が提供されています.

- **豊富なライブラリ群**
  数値計算：NumPy, SciPy
  グラフ描画：Matplotlib, seaborn, Bokeh
  データ処理：Pandas
  機械学習：scikit-learn, gensim
  深層学習：TensorFlow, PyTorch
- **Jupyter Notebook**
  Pythonのインタラクティブな開発環境.
  一連のデータ分析過程を簡単に編集・保存出来，再利用性も高く
  データ分析結果の共有に有用です.

データ分析プロセス例



卸売業者の顧客データ分析

データの取得先：
https://archive.ics.uci.edu/ml/datasets/Wholesale+customers

データには以下の情報が含まれます.
この分析ではデータの性質をクラスタリングで調べます.

- Fresh : 生鮮品の年間支出額
- Milk : 乳製品の年間支出額
- Grocery : 食料雑貨の年間支出額
- Frozen : 冷凍食品の年間支出額
- Detergents_Paper : 衛生用品と紙類の年間支出額
- Delicassen : 惣菜の年間支出額
- Channel : 販売チャネル(1=Horeca (ホテル等), 2=個人向け小売)
- Region : 各顧客の地域 (1=リスボン市, 2=ポルト市, 3=その他)

Python, Jupyter Notebookを用いたデータ分析を体験してみましょう.

データの確認

Python + Jupyter notebookでデータの読み込みと中身の確認をします.

```
import pandas as pd
data_df = pd.read_csv('data/Wholesale customers data.csv')
data_df = data_df.sample(frac=1).reset_index(drop=True)
data_df.head(5)
```

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	1	2	4414	1610	1431	3498	387	834
1	2	3	4591	15729	16709	33	6956	433
2	1	3	76237	3473	7102	16538	778	918
3	1	1	6202	7775	10817	1183	3143	1970
4	1	3	13265	1196	4221	6404	507	1788

データの前処理

データの型と欠損を確認します.

```
data_df.isnull().sum()
```

```
Channel           0
Region            0
Fresh             0
Milk              0
Grocery           0
Frozen            0
Detergents_Paper    0
Delicassen        0
dtype: int64
```

```
data_df.dtypes
```

```
Channel           int64
Region            int64
Fresh             int64
Milk              int64
Grocery           int64
Frozen            int64
Detergents_Paper    int64
Delicassen        int64
dtype: object
```

欠損値は無いようです.

カテゴリ変数の列は今回の分析には用いず消去します.

```
cat_col = ['Channel', 'Region']
data_df = data_df.drop(cat_col, axis=1)
```

データの概要の確認

基本的な統計量等を確認します.

```
data_df.describe()
```

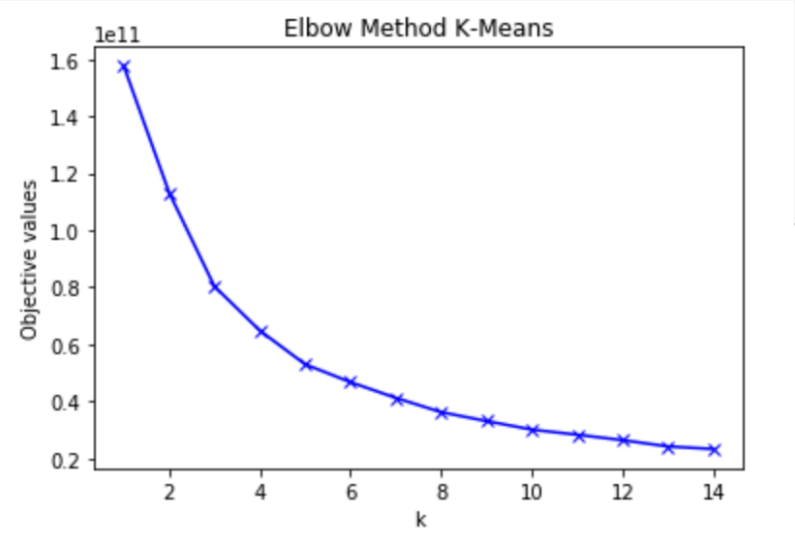
	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
count	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000
mean	12000.297727	5796.265909	7951.277273	3071.931818	2881.493182	1524.870455
std	12647.328865	7380.377175	9503.162829	4854.673333	4767.854448	2820.105937
min	3.000000	55.000000	3.000000	25.000000	3.000000	3.000000
25%	3127.750000	1533.000000	2153.000000	742.250000	256.750000	408.250000
50%	8504.000000	3627.000000	4755.500000	1526.000000	816.500000	965.500000
75%	16933.750000	7190.250000	10655.750000	3554.250000	3922.000000	1820.250000
max	112151.000000	73498.000000	92780.000000	60869.000000	40827.000000	47943.000000

K-平均法によるクラスタリング

エルボー法で適切なKを決定します.

```
X = data_df.values
from sklearn.cluster import KMeans
objective_vals = []
K = range(1,15)
for k in K:
    km = KMeans(n_clusters=k, init='k-means++', n_init=10, max_iter=300, tol=0.0001, random_state=0)
    km = km.fit(X)
    objective_vals.append(km.inertia_)
```

```
import matplotlib.pyplot as plt
plt.plot(K, objective_vals, 'bx-')
plt.xlabel('k')
plt.ylabel('Objective values')
plt.title('Elbow Method K-Means')
plt.show()
```



K=5のあたりが適当そうです.

K-平均法によるクラスタリング

K=5で再度K-平均法を学習します.

学習結果を見るためにデータセットにクラスタ番号列を追加します.

```
pred = KMeans(n_clusters=5, init='k-means++', n_init=10, max_iter=300, tol=0.0001, random_state=0).fit_predict(X)
data_df_ = pd.read_csv('data/Wholesale customers data.csv').drop(cat_col,axis=1)
data_df_['cluster']=pred
data_df_.head(5)
```

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen	cluster
0	12669	9656	7561	214	2674	1338	2
1	7057	9810	9568	1762	3293	1776	3
2	6353	8808	7684	2405	3516	7844	0
3	13265	1196	4221	6404	507	1788	1
4	22615	5410	7198	3915	1777	5185	0

学習結果の確認

各クラスタに割り当てられたデータ数と特徴の平均値を見てみます.

```
data_df_['cluster'].value_counts()
```

```
0   224
1   104
2    81
3    24
4     7
```

```
data_df_.groupby('cluster').mean()
```

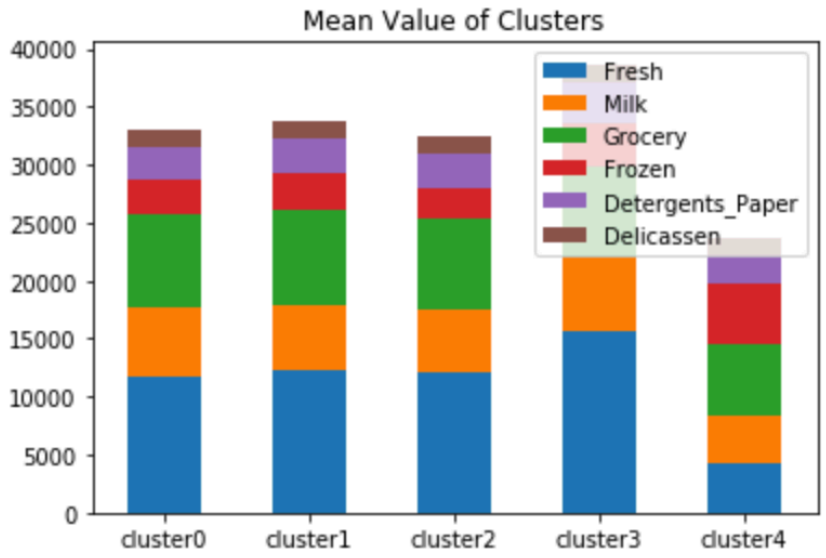
cluster	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	11697.116071	5988.598214	7973.589286	3042.040179	2778.968750	1493.129464
1	12206.365385	5735.971154	8171.942308	3190.086538	2879.298077	1631.019231
2	12164.098765	5313.185185	7799.925926	2607.938272	3014.740741	1478.839506
3	15641.000000	6377.875000	7819.000000	3765.625000	3558.833333	1508.791667
4	4262.714286	4133.285714	6163.714286	5263.714286	2330.714286	1551.285714

学習結果の確認

更に各クラスタの特徴の平均値を棒グラフで可視化します.

```
clusterinfo = pd.DataFrame()
for i in range(5):
    clusterinfo['cluster' + str(i)] = data_df_[data_df_['cluster'] == i].mean()
clusterinfo = clusterinfo.drop('cluster')

my_plot = clusterinfo.T.plot(kind='bar', stacked=True, title="Mean Value of Clusters")
my_plot.set_xticklabels(my_plot.xaxis.get_majorticklabels(), rotation=0)
```



分析結果はJupyter Notebook内表示され,
グラフィカルなレポートを作成出来ます.
Notebookでデータ分析結果の共有し課題
解決に向けた提案の材料にしましょう.