データ解析例（教師あり学習）

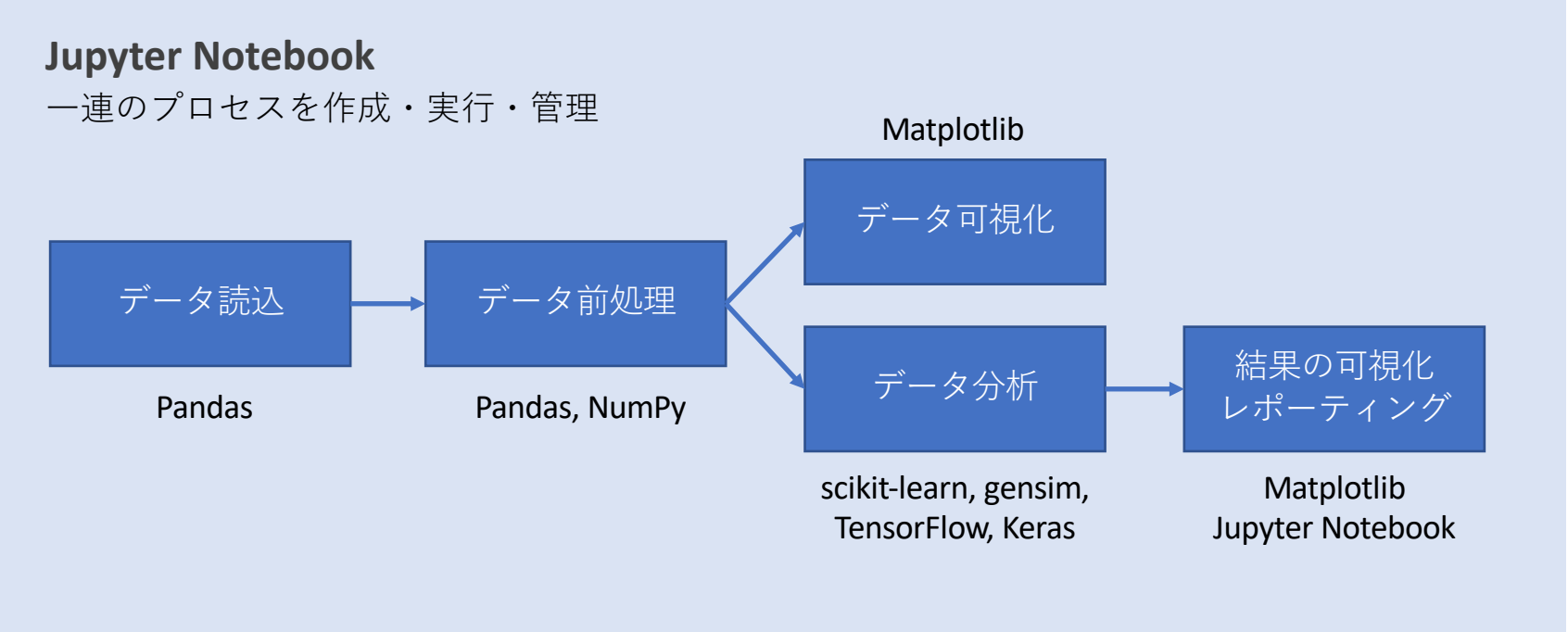
※バージョン情報
Python: 3.7.3, pandas: 0.24.2, scikit-learn: 0.20.3

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



タイタニックの生存分析

データの取得先：https://www.kaggle.com/c/titanic/data

本講義ではtrain.csvをdata.csvにリネームし訓練データとテストデータに分割します.

タイタニックデータには以下の情報が含まれます.
生存情報Survivedの分析を行います.

- PassangerID: 乗船者ID
- Survived: 生存情報 (0=死亡, 1 = 生存)
- Pclass: チケットのクラス (1=1st, 2=2nd , 3=3rd)
- Name / Sex / Age: 名前 / 性別 / 年齢
- SibSp / Parch: 乗船している兄弟・配偶者 / 親子の人数
- Ticket / Fare / Cabin : チケット番号 / 運賃 /キャビン番号
- Embarked: 乗船場所 (C = Cherbourg, Q = Queenstown, S = Southampton)

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/titanic.csv")
data_df
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S

欠損値（NaN）があることも分かります.

データの概要を確認

データの基本的な統計量等を確認します.

```
data_df.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

count: データ件数， mean: 平均値， std: 標準偏差,
min: 最小値， max: 最大値,
25%, 50%, 75%: 1/4分位数， 中央値， 3/4分位数.

データの前処理

各特徴のデータ型と欠損値の数を確認します.

```
data_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
```

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

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

※ isnull().sum()でも欠損値の確認が出来ます.

Age, Cabin, Embarkedが行数と整合せず欠損しています. 今回の分析ではCabinは欠損件数が多いので列を削除し, Cabin, Embarkedについては欠損している行を削除します.

```
data_df = data_df.drop(["Cabin"],axis=1)
data_df = data_df.dropna(subset=["Age", "Embarked"])
```

データの前処理

- 予測に無関係な列を消去します.

```
data_df = data_df.drop(["PassengerId", "Name","Ticket"],axis=1)
data_df.dtypes
```

IDや名前を用いた予測は汎化しない事に注意しましょう.

列削除後の特徴

```
Survived     int64
Pclass       int64
Sex         object
Age        float64
SibSp        int64
Parch        int64
Fare       float64
Embarked    object
dtype: object
```

- カテゴリ変数をダミー変数に変換します.

```
cat_col = ['Sex', 'Pclass', 'Embarked']
data_df = pd.get_dummies(data_df, columns=cat_col)
data_df.dtypes
```

ダミー変数に変換後の特徴

```
Survived       int64
Age          float64
SibSp          int64
Parch          int64
Fare         float64
Sex_female     uint8
Sex_male       uint8
Pclass_1       uint8
Pclass_2       uint8
Pclass_3       uint8
Embarked_C     uint8
Embarked_Q     uint8
Embarked_S     uint8
dtype: object
```

訓練データとテストデータの分割

dataを8:2に分割しそれぞれ訓練データ，テストデータとします．

訓練データで学習をしテストデータで分析手法の評価をします．

```
from sklearn.model_selection import train_test_split
train_df, test_df = train_test_split( data_df, test_size = 0.2, random_state = 0)
print('訓練データ数 : {}, テストデータ数 : {}'.format(len(train_df), len(test_df)))
```

```
訓練データ数 : 569, テストデータ数 : 143
```

※ Kaggleで配布されているテスト用ファイルはその予測値をKaggleに送信して評価するためのものであり正解ラベルを含みません．そのため，今回の講義では配布された訓練用データを実際の訓練用とテスト用に分割します．

データの標準化

データを標準化（各列の平均を0，分散を1）します．

標準化のための変換方法は訓練データでの平均・分散から定めます．

テストデータについては一般に訓練データの平均・分散で代用した標準化を行います（訓練データと同等の変形を担保するため）．

```
from sklearn.preprocessing import StandardScaler

# numpyの配列として値を取り出す
X_train = train_df.iloc[:, 1:].values
Y_train = train_df.iloc[:,0].values
X_test = test_df.iloc[:, 1:].values
Y_test = test_df.iloc[:,0].values

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

fit_transformは与えられたデータの平均・分散の計算と標準化をします．

transformではfit_transformで計算された平均・分散を用いた標準化を与えられたデータに対し実行します．

ロジスティック回帰による分類

正則化係数は5-交差検証で決定します.

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
import numpy as np

params = {"C": np.logspace(0, 4, 5)}
logreg_cv = GridSearchCV(LogisticRegression(), cv=5, param_grid=params)
logreg_cv.fit(X_train, Y_train)
print('訓練データでの分類精度 : {0:.2%}'.format(logreg_cv.score(X_train, Y_train)))
print('テストデータでの分類精度 : {0:.2%}'.format( logreg_cv.score(X_test, Y_test)))
```

```
訓練データでの分類精度 : 80.14%
テストデータでの分類精度 : 81.82%
```

GridSearchCVではfitで交差検証が行われ選ばれたモデルがデフォルトに設定されています.

混同行列の描画

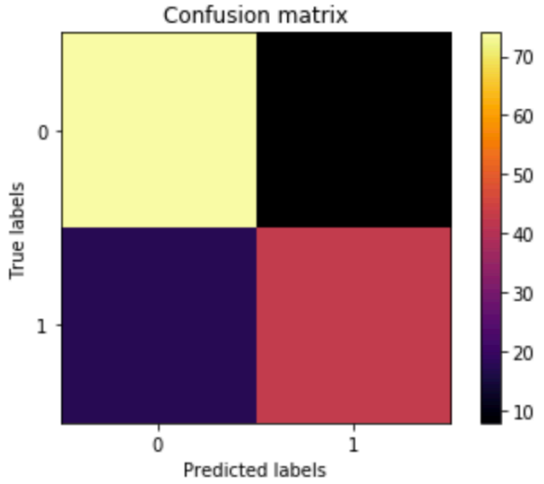
Matplotlibライブラリは豊富なグラフ描画機能を提供し
分類結果の混同行列も次のように描画出来ます.

```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt

cm = confusion_matrix(Y_test,logreg_cv.predict(X_test),labels=[0,1])

plt.xticks( [0,1,2], [0,1] )
plt.yticks( [0,1,2], [0,1] )
plt.title('Confusion matrix')
plt.xlabel('Predicted labels')
plt.ylabel('True labels')

plt.imshow(cm,  cmap=plt.cm.inferno)
plt.colorbar()
```



Jupyter Notebook内に図が表示され，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.