クレジット:
UTokyo Online Educationデータマイニング入門 2018 森　純一郎

東京大学
THE UNIVERSITY OF TOKYO

CC BY NC ND

課題3 データの前処理と可視化

In [ ]:
```
# 必要なモジュールの読み込み
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

Q1 欠損値の処理

欠損値の検出

pandasのシリーズやデータフレームではデータに欠損値（例えば数値データ内の空値）が含まれる場合、欠損値が `NaN` となります。以下では欠損値（空値）を含む得点データのcsvファイル, 'score_missing.csv', を読み込み、データフレーム `score` を作成します。

In [ ]:
```
score = pd.read_csv('score_missing.csv')
score
```

データフレームに欠損値が含まれるかどうかはpandasの `isnull()` メソッドを使うと調べることができます。 `isnull()` メソッドはデータフレームの各要素の値について、欠損値であれば `True` 、欠損値でなければ `False` を要素としたデータフレームを返します。

`isnull()` メソッドの詳細は以下を参照してください。

isnull() (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.isnull.html#pandas.isnull)

In [ ]:
```
score.isnull()
```

`isnull()` メソッドで返された上記のデータフレームの `any()` メソッド使うと、データフレームにおいて欠損値を含む行または列を調べることができます。以下では、列ごとに欠損値を含むかどうかを示すシリーズオブジェクトを返します。

In [ ]:
```
# 列ごとに欠損値があるか調べる
score.isnull().any()
```

上記で `any()`メソッド の引数 `axix` を1とすると、行ごとに欠損値を含むかどうかを示すシリーズオブジェクトを返します。

In [ ]:
```
# 行ごとに欠損値があるか調べる
score.isnull().any(axis=1)
```

上記のシリーズオブジェクトを以下のようデータフレーム抽出の条件として用いると、元のデータフレームから欠損値を含む行または列からなるデータフレームを抽出できます。

In [ ]:
```
# データフレームから欠損値を含む行を抽出
score[score.isnull().any(axis=1)]
```

In [ ]:
```
# データフレームから欠損値を含む列を抽出
score.loc[:, score.isnull().any()]
```

欠損値の削除

pandasの `dropna()` メソッドを使うと欠損値を含む行を削除することができます。 `dropna()` メソッドでは、この他にもさまざまな方法で欠損値を削除することができます。詳細は以下を参照してください。

dropna() (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.dropna.html)

In [ ]:
```
score.dropna()
```

欠損値の補完

pandasの `fillna()` メソッドを使うと欠損値を補完することができます。以下では、欠損値が含まれる列の欠損がない要素の値の平均値でその列にある欠損値を補完しています。 `fillna()` メソッドでは、この他にもさまざまな方法で欠損値を補完することができます。詳細は以下を参照してください。

fillna() (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.fillna.html)

In [ ]:
```
score = pd.read_csv('score_missing.csv')
score.fillna(score.mean())
```

Q1.1

欠損値（空値）を含む得点データのcsvファイル, 'score_missing.csv', を読み込み作成したデータフレームを受け取り、各列の欠損値をその列の欠損がない要素の値の中央値で補完したデータフレームを返す `fill_median()` 関数を完成させてください。

In [ ]:
```
score = pd.read_csv('score_missing.csv') # csvファイル読み込み
def fill_median(df):

```

`fill_median()` 関数が完成したら、以下のセルを実行してデータフレームのすべての要素の和を求めることで動作を確認してください。すべての要素の和は `2013` となります。

In [ ]:
```
fill_median(score).sum().sum()
```

Q2 外れ値の処理

NumPyでの記述統計

以下では、次のような形式の"exam_score.csv"ファイルを読み込み、NumPyの配列を作成し、NumPyの関数または `ndarray` オブジェクトのメソッドを用いて、配列の記述統計を求めています。

```
## exam_score.csvファイル
kokugo, shakai, sugaku, rika
30, 43, 51, 63
39, 21, 49, 56
...
```

In [ ]:
```
# csvファイルからNumPy配列の作成
score = np.loadtxt("exam_score.csv", delimiter=",", skiprows=1)
```

In [ ]:
```
# NumPyの関数で配列の各列の平均、分散、中央値、最大、最小を求める
print(np.mean(score, axis=0)) # 平均
print(np.var(score, axis=0)) # 分散
print(np.median(score, axis=0)) # 中央値
print(np.amax(score, axis=0)) # 最大値
print(np.amin(score, axis=0)) # 最小値
```

In [ ]:
```
# ndarrayのメソッドで配列の各列の平均、分散、最大、最小を求める
print(score.mean(0)) # 平均
print(score.var(0)) # 分散
print(score.max(0)) # 最大値
print(score.min(0)) # 最小値
```

Q2.1

整数を要素とする任意の長さの1次元配列を入力として受け取り、配列の要素の値の中で、以下の外れ値の基準にあてはまる要素からなる1次元配列を返す `find_outliers()` 関数を完成させてください。なお、IQRは四分位範囲であり、第3四分位から第1四分位を引いた値です。

- （第1四分位-1.5IQR）を値の下限としてそれより小さい値は外れ値とする
- （第3四分位+1.5IQR）を値の上限としてそれより大きい値は外れ値とする

配列の第1四分位と第3四分位は、NumPyの `percentile()` 関数を用いて以下のように求められる。

```
第1四分位, 第3四分位=np.percentile(配列, [25, 75])
```

In [ ]:
```
def find_outliers(input_array):

```

`find_outliers()` 関数が完成したら、以下のセルを実行して動作を確認してください。 `[-100, 200, 1000]` が外れ値の配列です。

In [ ]:
```
find_outliers(np.array([30,39,-100,29,95,70,67,200,29,1000,56,45,68]))
```

Q2.2

以下のような形式の"score_outlier.csv"ファイルを読み込み、データフレーム score を作成します。

```
## score_outlier.csvファイル
kokugo, shakai, sugaku, rika
30, 43, 51, 63
39, 21, 49, 56
...
```

score を受け取り、各列ごとに、Q2.1と同様に四分位範囲, 第3四分位, 第1四分位を用いて外れ値を検出し、外れ値を含む行を除いたデータフレームを返す関数 drop_outliers() を完成させてください。データフレームの各列の第1四分位と第3四分位は、pandas の describe() 関数を用いて以下のように求められる。

```
第1四分位 = データフレーム[列名].describe()['25%']
第3四分位 = データフレーム[列名].describe()['75%']
```

In [ ]:
```
score = pd.read_csv('score_outlier.csv') # csvファイル読み込み
def drop_outliers(df):

```

drop_outliers() 関数が完成したら、以下のセルを実行して外れ値を含む行を除いたデータフレームの行数を求めることで動作を確認してください。行数は 155 となります。

In [ ]:
```
len(drop_outliers(score).index)
```

Q2.3

matplotlib では、以下のようにして複数の1次配列を入力として、それぞれを箱ひげ図として同時に可視化することができます。

```
plt.boxplot([配列1, 配列2, 配列3, ....], showmeans=True);
```

"score_outlier.csv"ファイルを読み込み、NumPyの配列を作成し、各教科の点数の箱ひげ図を可視化してください。

In [ ]:
```
# csvファイルからNumPy配列の作成
score = np.loadtxt("score_outlier.csv", delimiter=",", skiprows=1)

```

Q3 標準化

以下のような形式の"exam_score.csv"ファイルを読み込み、作成したNumPyの配列を受け取り、各教科の点数をその教科の平均値と分散を用いて標準化した配列を返す `normalize_score()` 関数を完成させてください。平均値と分散の計算には、NumPyの関数またはndarrayオブジェクトのメソッド、どちらを使ってもよいです。また、分散は標本分散としてください。

```
## exam_score.csvファイル
kokugo, shakai, sugaku, rika
30, 43, 51, 63
39, 21, 49, 56
...
```

配列の列ごとに関数やメソッドを適用するには以下のように `axis` 引数を0とする。

```
np.mean(配列, axis=0)
配列.mean(axis=0)
```

また、配列の元の次元を保持したい場合は `keepdims` 引数をTrueとする。

```
np.mean(配列, axis=0, keepdims=True)
配列.mean(axis=0, keepdims=True)
```

In [ ]:
```
# csvファイルからNumPy配列の作成
score = np.loadtxt("exam_score.csv", delimiter=",", skiprows=1)
def normalize_score(a):

```

参考

pandasでは以下のようにしてデータフレームの各列をその列の平均値と分散を用いて標準化できます。

In [ ]:
```
score_df = pd.read_csv('exam_score.csv')
((score_df - score_df.mean()) / score_df.std(ddof=0)).head(5)
```

Q3 ヒストグラム

matplotlib では、以下のように1次配列を入力として、ヒストグラムを可視化することができます。

```
plt.hist(配列, bins=階級数);
```

"exam_score.csv"ファイルを読み込み、NumPyの配列を作成し、数学（3列目）の点数のヒストグラムを可視化してください。その際、階級数は任意に設定してください。

In [ ]:
```
# csvファイルからNumPy配列の作成
score = np.loadtxt("exam_score.csv", delimiter=",", skiprows=1)
```

参考

平均=muと標準偏差=stdの時の正規分布に従うランダムな値からなる配列を入力とするヒストグラム

In [ ]:
```
mu, std = 0, 1
nor = np.random.normal(mu, std,200)
plt.hist(nor, bins=20);
```

Q4 相関

Q4.1

整数を要素とする同じ長さの2つの1次元配列を入力として受け取り、それらの配列の要素の相関係数を返す correlation() 関数を完成させてください。2つの配列は先頭から順にそれぞれの要素が対応するものとします。それぞれの配列の対応する要素の値をx_i, y_iとすると、相関係数は以下のように求められます。

$\frac{1}{n}\Sigma_{i=1}^{n}(\frac{x_i-\bar{x}}{s_x})(\frac{y_i-\bar{y}}{s_y})$

$\bar{x}, \bar{y}$は平均、s_x, s_yは標準偏差です。

In [ ]:
```
def correlation(x, y):
```

correlation() 関数が完成したら、以下のセルを実行して動作を確認してください。それぞれ相関係数は 1 , -1 となります

In [ ]:
```
correlation(np.array([-1,1]),np.array([ -1, 1]))
```

In [ ]:
```
correlation(np.array([1,-1]),np.array([-1, 1]))
```

Q4.2

以下のような形式の"exam_score.csv"ファイルを読み込み、数学と理科の点数の関係を散布図として可視化してください。

```
## exam_score.csvファイル
kokugo, shakai, sugaku, rika
30, 43, 51, 63
39, 21, 49, 56
...
```

matplotlib では、以下のように2つの1次配列を入力として、散布図を可視化することができます。

```
plt.plot(配列1, 配列2, 'o');
```

In [ ]:
```
# csvファイルからNumPy配列の作成
score = np.loadtxt("exam_score.csv", delimiter=",", skiprows=1)
```

参考

NumPyの corrcoef() 関数を使うと2つの配列の共分散行列を以下のように求めることができます。

In [ ]:
```
score = np.loadtxt("exam_score.csv", delimiter=",", skiprows=1)
np.corrcoef(score[:,0], score[:,1])
```

pandasのcorrメソッドを使うとデータフレームの列間の相関係数を以下のように求めることができます。

In [ ]:
```
score = pd.read_csv("exam_score.csv", sep=",")
score.corr(method='pearson')
```

これらの相関係数の大小に基づいて以下のように列間の相関の大小をヒートマップとして可視化することができます。

In [ ]:
```
plt.colorbar(plt.imshow(score.corr(method='pearson')));
```