sklearnを使った回帰分析

単回帰モデル、重回帰モデル

importするライブラリ

- import numpy as np
- import pandas as pd
- import matplotlib.pyplot as plt
- import seaborn as sns
- from sklearn.linear_model import LinearRegression
- from sklearn.feature_selection import f_regression
- from sklearn.preprocessing import StandardScaler

データの読み込み

pandasを使ってcsvファイルの読み込みを行う

```
data = pd.read_csv('ファイル名')
```

読み込んだファイルの確認

```
data
```

読み込んだファイルの最初の方を確認

```
data.head()
```

従属変数、独立変数の設定

従属変数（y）ターゲット

```
y = data.['従属変数名']
```

独立変数（x）特徴量

単回帰分析

```
x = data.[['独立変数名']]
```

※独立変数は2次元配列で格納する為[ ]で入力する。

重回帰分析

```
x = data.[['独立変数名1','独立変数名2','独立変数名3',・・・・]]
```

単回帰オブジェクトの作成

sklearnのLinearRegressionクラスオブジェクトの実装

```
reg = LinearRegression()
```

ターゲットと特徴量の代入

```
reg.fit(x_matrix,y)
```

決定係数を求める

scoreメソッドを使って決定係数を求める

```
reg.score(x_matrix,y)
```

coef_で係数の確認　　※独立変数の個数の係数を配列で返す

```
reg.coef_
```

intercept_で切片の確認

```
reg.intercept_
```

自由度調整済み決定係数の求め方

sklearnには自由度調整済み決定係数を求めるメソッドがないため、
下記の関数を利用する。

```
def adj_coef(x,y):
    r2 = reg.score(x,y)
    # n = データの数
    n = x.shape[0]
    # p = 変数の数
    p = x.shape[1]
    # 公式
    adjusted_r2 = 1-(1-r2)*(n-1)/(n-p-1)
    return adjusted_r2
```

作成した回帰モデルを使って予測してみる

predictメソッドを使用する。

reg.predict([[任意の特徴量]])

※特徴量は複数入力することができる。
predictの引数は2次元配列で渡す。

p値の計算

from sklearn.feature_selection import f_regression を使ってp値を求める

f_regressionメソッドからp値を取り出す
p_values = f_regression(x,y)[1]

p値を丸める（小数点第3位表示にする）
p_values = p_values.round(3)

※このp値(p_values)は各独立変数毎に表示されている。
独立変数が3つあればp_valuesは3つ返される。

独立変数名・係数・p値のテーブルの作成

下記の関数を呼び出せば作成できる。

```
def reg_table(x,reg,p_values):

    reg_summary = pd.DataFrame(data = x.columns.values,columns=['Features'])
    reg_summary ['Coefficients'] = reg.coef_
    reg_summary ['p-values'] = p_values.round(3)
    return reg_summary
```

フューチャースケーリング（標準化）

from sklearn.preprocessing import StandardScalerを使って標準化する。

StandardScalerクラスを実装する。
　　scaler = StandardScaler()

fitメソッドを使って標準化の準備をする。
　　scaler.fit(x)

transformメソッドを使って標準化を実行する
　　x_scaled = scaler.transform(x)

※標準化の準備と標準化を一気に行いたい場合は
　　x_scaled = scaler.fit_transform(x)
で実行できる。