機械学習のための
PythonマスターコースDay1

https://to-kei.net

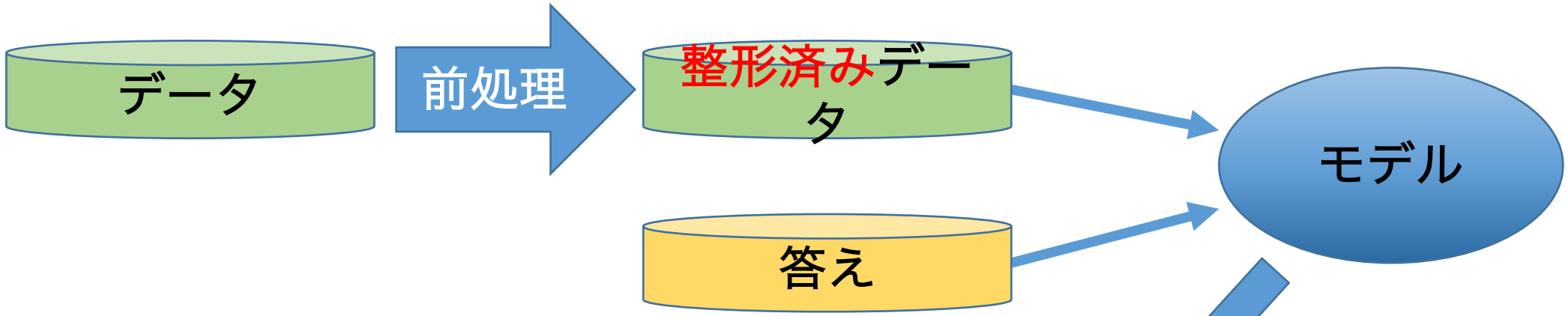
全人類がわかる統計学 Presents

本日の目標

ライブラリであるpandasの基本操作を習得し、
Python上で自在にデータの操作ができるようになる

データ分析、機械学習の基本

学習フェーズ



予測フェーズ

目次

- データの読み込み
- データの確認
- データから情報を取り出す
- 演習問題（データの取り出し）
- 新たな列の作成
- 並び替え
- 置換
- データの取り出し（高度）
- 演習問題（新列作成,書き出し,置換）
- データの結合
- 欠損値処理
- ダミー変数化
- 学習データと検証データへの分割

データの読み込み

csvファイルの読み込み

```
#必要ライブラリの読み込み
import pandas as pd
import sklearn

#データの読み込み
df = pd.read_csv("International_football_results.csv")
print(df)
```

```
           date        home_team        away_team  home_score  away_score  \
0    1872-11-30         Scotland          England           0           0
1    1873-03-08          England         Scotland           4           2
2    1874-03-07         Scotland          England           2           1
3    1875-03-06          England         Scotland           2           2
4    1876-03-04         Scotland          England           3           0
5    1876-03-25         Scotland            Wales           4           0
6    1877-03-03          England         Scotland           1           3
7    1877-03-05            Wales         Scotland           0           2
8    1878-03-02         Scotland          England           7           2
9    1878-03-23         Scotland            Wales           9           0
10   1879-01-18          England            Wales           2           1
```

現在のディレクトリにファイルがない場合は、絶対パス、相対パスで指定

データの確認

最初の5行だけ表示

```
#最初の5行だけ表示
print(df.head())
```

```
         date home_team away_team  home_score  away_score tournament     city  \
0  1872-11-30  Scotland   England           0           0   Friendly  Glasgow
1  1873-03-08   England  Scotland           4           2   Friendly   London
2  1874-03-07  Scotland   England           2           1   Friendly  Glasgow
3  1875-03-06   England  Scotland           2           2   Friendly   London
4  1876-03-04  Scotland   England           3           0   Friendly  Glasgow

    country  neutral
0  Scotland    False
1   England    False
2  Scotland    False
3   England    False
4  Scotland    False
```

df.head(10)などとすると最初の10行を表示

データの詳細情報のチェック

➢ プログラム

```
#データの情報を表示
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39008 entries, 0 to 39007
Data columns (total 9 columns):
date          39008 non-null object
home_team     39008 non-null object
away_team     39008 non-null object
home_score    39008 non-null int64
away_score    39008 non-null int64
tournament    39008 non-null object
city          39008 non-null object
country       39008 non-null object
neutral       39008 non-null bool
dtypes: bool(1), int64(2), object(6)
memory usage: 2.4+ MB
```

➢ポイント

- 行数、列数、要素数、型、メモリのチェックが可能

データの形をチェック

```
#データの形を確認
print(df.shape)

#列名の表示
print(df.columns)

#行名の表示
print(df.index)

#データの縦の長さをチェック（行数）
print(len(df))
```

```
(39008, 9)
Index(['date', 'home_team', 'away_team', 'home_score', 'away_score',
       'tournament', 'city', 'country', 'neutral'],
      dtype='object')
RangeIndex(start=0, stop=39008, step=1)
39008
```

データから情報を取り出す

列名で列を取り出す

➢プログラム

```
#1列の場合
print(df.home_team) #列名がhome_teamの
print(df["home_team"]) #列名がhome_tea
```

```
#2列以上の場合
print(df[["home_team","away_team"]])
```

➢ポイント

- 一列の場合は、
  df.列名かdf["列名"]で取り出す
- 複数列取り出す場合は、
  df[列名のリスト]
  を与える。

特定の列だけを抽出して別変数に代入することも可

```
#特定の列だけを抽出した新しいデータフレームを作成
df_teams = df[["date","home_team","away_team"]]
print(df_teams)
```

	date	home_team	away_team
0	1872-11-30	Scotland	England
1	1873-03-08	England	Scotland
2	1874-03-07	Scotland	England
3	1875-03-06	England	Scotland
4	1876-03-04	Scotland	England
5	1876-03-25	Scotland	Wales
6	1877-03-03	England	Scotland
7	1877-03-05	Wales	Scotland
8	1878-03-02	Scotland	England
9	1878-03-23	Scotland	Wales

行番号で行を取り出す

➢プログラム

```
#0行目以上、1000行目未満だけを抽出
print(df[0:1000])

#15000行目以降を抽出
print(df[15000:])

#0行目以上、100行目未満を2行ごとに抽出
print(df[0:100:2])

#全データを2行ごとに抽出
print(df[::2])
```

➢ポイント

- df[n1:n2:n3]で
  n1以上、n2未満、増加量n3
  が基本

- 省略可。省略した場合、それぞれ
  n1省略：0
  n2省略：最後の行
  n3省略：1
  となる。

.locで行番号と列名を同時に利用する

```
#10行目以上、20行目以下、のhome_teamとaway_team
print(df.loc[10:20,["home_team","away_team"]])

#10行目以上、20行目以下、1行置きの、home_teamとaway_team
print(df.loc[10:20:2,["home_team","away_team"]])
```

➢ポイント

- df.loc[行指定,列指定]でデータの一部を取り出す。

- 行指定をn:mとした場合n以上m以下。m未満ではないことに注意！

.ilocで行番号と列番号で取り出す

➢プログラム

```
#1行目、1列目
print(df.iloc[1,1])

#10行目以上、20行目未満。1列目以上、5列目未満。
print(df.iloc[10:20,1:5])

#全行。1列目以上、5列目未満。
print(df.iloc[:,1:5])
```

➢ポイント

- df.iloc[行,列]で指定。
- 書き方は、行番号で取り出す場合と同じルールが適用される。

列から統計的な値を計算する

➢プログラム

```
#最大値
print(df[["home_score","away_score"]].max())

#最小値
print(df[["home_score","away_score"]].min())

#平均値
print(df[["home_score","away_score"]].mean())

#標準偏差
print(df[["home_score","away_score"]].std())
```

```
home_score    31
away_score    21
dtype: int64
home_score    0
away_score    0
dtype: int64
home_score    1.740130
away_score    1.180091
dtype: float64
home_score    1.730864
away_score    1.378856
dtype: float64
```

➢ポイント

- .max()で列の最大値を計算する
- .min()で最小値
- .mean()で平均
- .std()で標準偏差

データから情報を取り出す（応用）

numpy配列におけるデータの取り出し①

➢プログラム

```
import numpy as np

array1 = np.array([16,10,3])

#全要素を取り出す
print(array1[[True,True,True]])

#1つ目と3つ目の要素を取り出す
print(array1[[True,False,True]])
```

```
[16 10  3]
[16  3]
```

➢ポイント

- numpy配列は真偽値ベクトルを与えることで、Trueに対応する部分だけを抽出することが可能
- pandasの基本動作はnumpy配列と同じ！！

numpy配列におけるデータの取り出し②

➢プログラム

```
#array1の各要素が5より大きいかの判定
print(array1 > 5)

#array1の各要素が10以下かの判定
print(array1 <= 10)

#array1から10以下の要素のみを抽出する
print(array1[array1 <= 10])
```

```
[ True  True False]
[False  True  True]
[10  3]
```

➢ポイント

- numpy配列は単体の値と比較することで、真偽ベクトルを出力する
- 条件を指定することによって、条件に合うデータのみを抽出可能
- この方法はpandasのデータにも応用可能！！

条件に一致する行を取り出す（pandas）

```
#home_teamがEnglandのデータ
print(df[df["home_team"] == "England"])

#home_teamがEngland　かつ　away_teamがScotlandのデータ
print(df[(df["home_team"] == "England") & (df["away_team"] == "Scotland")])

#home_teamがEngland　または　away_teamがScotlandのデータ
print(df[(df["home_team"] == "England") | (df["away_team"] == "Scotland")])

#dateに1872が含まれる行を抽出（1872年に行われた試合を抽出）
print(df[df.date.str.contains("1872")])

#1872~1874年の間に行われた試合を抽出
print(df[df.date.str.contains("1872") | df.date.str.contains("1873") | df.date.str.contains("1874")])

#Japanの試合のみを全て抽出
print(df[(df["home_team"] == "Japan") | (df["away_team"] == "Japan")])
print(df[(df[["home_team","away_team"]] == "Japan").any(axis = 1)])

#Japanの試合のみ抽出してdf_Japanに格納
df_Japan = df[(df[["home_team","away_team"]] == "Japan").any(axis = 1)]
```

&で積集合（かつ）、|で和集合（または）を抽出可能

演習問題（データの取り出し）

演習問題

1.列名が"tournament"の列を取り出せ。

2.行番号100行目以上6000行目未満を取り出せ。

3.列名が"date","city","coutry"となっている列を3列まとめて取り出せ。

4."Brazil"がhome_teamとなっている試合を取り出せ。

5."Brazil"が出場している試合を全て取り出せ。

6.2003年に行われた試合を全て取り出せ。

新たな列の作成

データフレームの列同士の演算

```
#点差を計算
point_diff = df["home_score"] - df["away_score"]
point_diff = abs(point_diff)

#点差（point_diff）という列を作成
df["point_diff"] = point_diff
print(df.head()) #新しい列が作成されていることの確認
```

➢ポイント

- pandasでは行列演算が可能
- abs() で絶対値をとる
- df["新しい列名"] = 配列　で新しい列を作成可能

内包表現を用いた文字列の部分切り出し

```
#年、月、日を格納する列を作成（難易度高め,初学者は書き方を知って置いて、出てきたときにビビらない程度になれば良い）
df['year'] = pd.to_numeric([date.split('-')[0] for date in df['date']]) #実施年だけを格納した列
df['month'] = pd.to_numeric([date.split('-')[1] for date in df['date']]) #実施月だけを格納した列
df['day'] = pd.to_numeric([date.split('-')[2] for date in df['date']]) #実施日だけを格納した列
```

➢ポイント

- pd.to_numeric で文字列型を数値型に変換
- 文字列.split(-) を使うと-で区切ったリストが生成する
- df["新しい列名"] = 配列　で新しい列を作成可能

並び替え

並び替え

```
#home_scoreによる並び替え
#ascending = Falseで降順指定。Trueだと昇順。デフォルトはTrue。
df.sort_values("home_score",ascending = False)

#away_scoreとhomescoreによる並び替え。away_score優先
df.sort_values(["away_score","home_score"],ascending = False)
```

➢ポイント

- ascending=False で降順。指定なしで昇順。

- 引数のリストを2つ以上に指定することも可能、左から優先される。

置換

置換

➢プログラム

```
#1を100に置き換える
print(df.replace({1:100}))

#1を100に2を200に置き換える
print(df.replace({1:100,2:200}))

#Trueを1、Falseを0に置き換える
print(df.replace({True:1,False:0}))
```

➢ポイント

- df.raplace({置換元:置換先})で、置換が可能。
- 置換したデータは元の変数に代入しない限りは表示されるだけ

データの取り出し（高度）

分析に使う国だけのデータを作成する

```
Japan_exists = (df[["home_team","away_team"]] == "Japan").any(axis = 1).replace({True:1,False:0})
Brazil_exists = (df[["home_team","away_team"]] == "Brazil").any(axis = 1).replace({True:1,False:0})
Germany_exists = (df[["home_team","away_team"]] == "Germany").any(axis = 1).replace({True:1,False:0})
Italy_exists = (df[["home_team","away_team"]] == "Italy").any(axis = 1).replace({True:1,False:0})
Argentina_exists = (df[["home_team","away_team"]] == "Argentina").any(axis = 1).replace({True:1,False:0})

df5 = df[Japan_exists+ Brazil_exists + Germany_exists + Italy_exists + Argentina_exists == 2]
print(df5)
```

➢ポイント

- Trueは1、Falseは0に置換
- 試合で5カ国中の2カ国が含まれていれば和は2
- df5に作成した新規データを追加

データの書き出し

csvファイルへのデータの書き出し

➢プログラム

```
#データの書き出し
df5.to_csv("5team.csv",index = False)
```

➢ポイント

- index = False で行名無し
- デフォルトは上書きモード
- 追記モードにするには、
  mode = "a"

演習問題（新列作成,書き出し,置換）

演習問題

1. 試合2チームの総合得点の列、point_addを追加せよ

2. イングランド(England)が出場した試合を全て抜き出し新たな変数に代入せよ

3. 2で作成したデータフレームをcsvファイルへ書き出せ

データの結合

図で見るデータの結合

pd.concat([df1, df2])

df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df2

	A	B	C	D
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7

Result

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7

縦方向の結合

pd.concat([df1, df4], axis=1)

df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df4

	B	D	F
2	B2	D2	F2
3	B3	D3	F3
6	B6	D6	F6
7	B7	D7	F7

Result

	A	B	C	D	B	D	F
0	A0	B0	C0	D0	nan	nan	nan
1	A1	B1	C1	D1	nan	nan	nan
2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3
6	nan	nan	nan	nan	B6	D6	F6
7	nan	nan	nan	nan	B7	D7	F7

横方向の結合

参考：http://sinhrks.hatenablog.com/entry/2015/01/28/073327

勝った国の列を作成

```
#勝った国の列を追加
homewin = (df5[df5.away_score < df5.home_score]).home_team #ホームチームが勝った場合
awaywin = (df5[df5.away_score > df5.home_score]).away_team #アウェイチームが勝った場合
winCountry = pd.concat([homewin,awaywin]) #
df5["win_country"] = winCountry #
print(df5.head())
```

➢ポイント

- homewinにホームで勝った国を代入
- awaywinにアウェイで勝った国を代入
- winCountryにhomewinとawaywinを結合したものを代入
- 引き分けは自動的にNaNとなる

home_teamが勝利したかの列を追加

```
import numpy as np

#勝利は1、引き分け0、敗北は-1
df5["home_win"] = np.nan
df5["home_win"] = df5["home_win"].mask(df5.home_score > df5.away_score , 1)
df5["home_win"] = df5["home_win"].mask(df5.home_score == df5.away_score , 0)
df5["home_win"] = df5["home_win"].mask(df5.home_score < df5.away_score , -1)
print(df5.head())
```

➢ポイント

- 新しい列”home_win”を作成
- home_teamが勝った場合home_winの要素を1に置き換える
- 引き分けは0に置き換える
- home_teamが負けた場合home_winを-1に置き換える

欠損値処理

欠損値処理

➢ プログラム

```
#欠損値のある行を除外する
print(df5.dropna())

#欠損値を穴埋めする
print(df5.fillna(0))

#欠損値を抽出する
print(df5.isnull())
print(df5[df5["win_country"].isnull()])

#欠損値を0で穴埋めしてもとデータに代入
df5 = df5.fillna(0)
print(df5)
```

➢ポイント

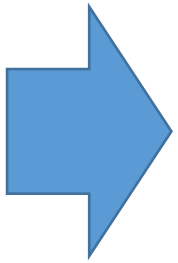
- .dropna()で欠損行除外
- .fillna(a)でaで欠損値を穴埋め
- .isnull()で欠損かどうかの真偽判定

- 穴埋めしたデータは代入で、元データフレームに反映させることが可能

ダミー変数化

ダミー変数とは？

国
日本
ブラジル
ドイツ
イタリア
アルゼンチン



日本	ブラジル	ドイツ	イタリア	アルゼンチン
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

カテゴリーデータ

数値データ

- ダミー変数化はカテゴリーデータを数値データに変換する一つの手段

ダミー変数とは？

```
#チームをダミー変数化したデータフレームを作成
team_dummy = pd.get_dummies(df5[["home_team","away_team"]])
print(team_dummy)
```

```
#元のデータフレームとチームをダミー変数化したデータフレームを結合
df5 = pd.concat([df5,team_dummy],axis=1)
```

```
#ダミー変数化したもの、tournament,city,country,neautralを消去
df5 = df5.drop(["home_team","away_team","tournament","city","country","neutral"],axis=1)
print(df5.head())
```

➢ポイント

- pd.get_dummiesでダミー変数化したものを取得
- ダミー変数にした変数は通常元のデータフレームからは削除する

学習データと検証データへ分割

ランダムにデータを抽出する

➢ プログラム

```
#ランダムに1行抽出
print(df5.sample())

#ランダムに3行抽出
print(df5.sample(3))

#ランダムに5行抽出
print(df5.sample(5))
```

➢ポイント

- .sample()でランダムに抽出
- ()内に何もいれなければ1行
  整数をいれると、入れた行数分
  抽出を行う

train_test_splitの利用

```
from sklearn.model_selection import train_test_split


#25%をテストデータとして使う
train_df5, test_df5 = train_test_split(df5)
print(len(train_df5),len(test_df5))

#50%をテストデータとして使う
train_df5, test_df5 = train_test_split(df5,test_size = 0.5)
print(len(train_df5),len(test_df5))
```

➢ポイント

- 返り値は二つなので、変数も二つ用意する

- test_sizeは特に指定しなければデフォルトの0.25

総合演習問題

総合演習問題

新規データsample-data.csvを読み込み以下の作業をせよ

1. データの基本情報を確認せよ。（行数、列数、初めの5行など）

2.データには欠損値がある。欠損値のある行を除け

3.性別（Gender）の列をダミー変数化してデータに付け加えよ

4.病気（Disease）が1の人のみを抜き出せ

5.データを学習用と検証用に8対2の割合で分割せよ。

アンケートのお願い

講座の改善のため、以下のURLからアンケートの協力をお願いしております

https://seminar.to-kei.net/qt/

仕事のご依頼・ご相談は
info@avilen.co.jp
までお問い合わせください