社内研修・データ分析

Version: 1.0

コンテンツ

Part 2

データ操作の基礎（参照・抽出・削除）

Version: 1.0

本研修の目的

データ分析の現場で即戦力となる人材の育成

パート2の目的:

DataFrameから必要なデータを条件に応じて正確に参照、抽出し、不要なデータを削除する操作（スライシング、フィルタリング）を習得します。

目次

ETLのT:データの変換

抽出 (Extraction):

- 基幹システム
- データベース
- 外部ファイルなど

様々なデータソースから必要なデータを抜き出す工程

変換 (Transformation):

- 抽出したデータを分析しやすい形式に整える工程です。
- データクレンジング、結合、新しい特徴量の作成など

格納 (Loading):

- 変換したデータを、分析用のデータベース（DWH）やデータマートに保存する工程です。

準備: 必要なライブラリとデータの読み込み

ライブラリの読み込み

本パートで使用するライブラリ:

```
from sqlalchemy import create_engine
import pandas as pd
import os
```

ワーキングディレクトリの変更とデータの読み込み

```
# データディレクトリへの移動（環境に合わせて調整してください）
if os.path.exists('data'):
    os.chdir('data')

# 1. 顧客データを読み込む（元のCSVは日本語ヘッダー）
df_customers = pd.read_csv('customers.csv')
```

読み込みます。

コラム名を物理名への変更

```python
# 2. 扱いやすいように英語の物理名に変換するマッピングを
column_mapping = {
    '顧客識別番号': 'customer_id',
    '氏名': 'customer_name',
    '氏名カナ': 'customer_name_kana',
    '生年月日': 'birth_date',
    '性別': 'gender',
    '住所': 'address',
    '電話番号': 'phone_number',
    'メールアドレス': 'email_address',
    '職業': 'occupation',
    '会社名': 'employer_name',
    '勤続年数': 'employment_years',
    '年収（万円）': 'annual_income_10k_yen',
    '申請目的': 'latest_application_purpose',
    '最初契約日': 'registration_date',
    '支店番号': 'main_branch_code',
    '主口座番号': 'main_account_number'
}
```

```python
# 3. renameメソッドで列名を変更
df_customers = df_customers.rename(columns=column_mapping)

# データの先頭を確認
df_customers[['customer_name','gender','address','occupation',
              'annual_income_10k_yen']].head()
```

	customer_name	gender	address	occupation	annual_income_10k_yen
0	田中 太郎	男性	東京都世田谷区成城6-1-1	会社員	650
1	佐藤 花子	女性	東京都港区六本木6-10-1	専門職	480
2	山本 健太	男性	神奈川県横浜市西区みなとみらい2-2-1	自営業	550
3	小林 美咲	女性	埼玉県さいたま市大宮区桜木町1-7-5	学生	150
4	加藤 誠	男性	東京都千代田区番町1-1	経営者	50000

目次

データの参照

主要の参照方法

特定の場所のデータをピンポイントで参照するための2つの主要な方法

- `.loc`: ラベルベースの参照。行ラベル（インデックス名） と 列ラベル（列名） を文字で指定します。

- `.iloc`: 位置（整数）ベースの参照。行番号 と 列番号（0から始まる）を数値で指定します。

参照方法

.loc

.iloc

ラベルベースの参照（.loc）

.loc

基本構文

```
df.loc[行ラベル, 列ラベル]
df.loc[開始行:終了行, [列A, 列B]]
```

> **重要なポイント:** `.loc` でスライス（範囲指定、例: `3:5`）を使う場合、**終点（この場合は `5`）も範囲に含まれます**。これはPythonの標準リストとは異なる挙動なので注意が必要です。

ラベルベースの参照（.loc）

使用例

```
df_customers.loc[3:5, ['customer_name', 'annual_income_10k_yen']]
```

✓ 0.0s

	customer_name	annual_income_10k_yen
3	小林 美咲	150
4	加藤 誠	50000
5	吉田 恵美	580

位置ベースの参照（.iloc）

.iloc

基本構文

```
df.iloc[行番号, 列番号]
df.iloc[開始行:終了行, 開始列:終了列]
```

> **重要なポイント:** `.iloc` でスライス（範囲指定、例: `3:6`）を使う場合、Pythonの標準リストと同様に**終点（この場合は`6`）は範囲に含まれません。**つまり、`3`, `4`, `5` 番目の行が対象となります。

位置ベースの参照(.iloc)

使用例

```python
# 行番号48のデータのうち、2列目(列番号1)、6列目(列番号5)、8列目(列番号7)を参照
display(df_customers.iloc[48, [1, 5, 7]])
# 4行目(行番号3)から6行目(行番号5)まで、2列目(列番号1)から4列目(列番号3)までのデータを参照
# 終点を含まないため、行は `3:6`、列は `1:4` と指定します
display(df_customers.iloc[3:6, 1:4])
```

✓ 0.0s Python

```
customer_name                  大塚 剛志
address                山梨県甲府市飯田2-2-1
email_address    takeshi.o@example.com
Name: 48, dtype: object
```

	customer_name	customer_name_kana	birth_date
3	小林 美咲	コバヤシ ミサキ	1999-02-28
4	加藤 誠	カトウ マコト	1965-10-15
5	吉田 恵美	ヨシダ エミ	1988-04-03

目次

条件によるデータの抽出（フィルタリング）

フィルタリング

データ分析の最も基本的な操作の一つが、特定の条件を満たす行だけを抽出するフィルタリングです。Excelのオートフィルタ やSQLのWHERE句にあたります。

主要方法

- 比較演算子でビットマスク
- 論理演算子で複数条件
- `.query()`でSQLぽいクエリ

単一条件でのフィルタリング

比較演算子ビットマスク

dataframe_o

df['列名'] 条

スク）」を使っ

ト（ブールマ

基本構文

```
df[df['列名'] == 値]
df[df['列名'] != 値]
df[df['列名'] >= 値]
df[df['列名'] <= 値]
df[df['列名'] > 値]
df[df['列名'] < 値]
```

単一条件でのフィルタリング

使用例

```
df_customers[mask].loc[:,['customer_
                          'annual_in
                          'email_add
```

	customer_name	customer_name_kana	annual_income_1	ress
4	加藤 誠	カトウ マコト		com
8	渡辺 大輔	ワタナベ ダイスケ		com
9	伊藤 綾香	イトウ アヤカ		com
28	藤井 健二	フジイ ケンジ		com
38	内田 啓介	ウチダ ケイスケ	1400	030-9000-1122 k.uchida@example.com

df_customers[mask]

上で作成したマスクを使って、年収が1,000万円以上の顧客を抽出

顧客の名前、年収、電話番号、メールアドレスを表示

複数条件でのフィルタリング

論理演算子複数条件

基本構文

```
df[(条件式1) & (条件式2)]  # AND条件
df[(条件式1) | (条件式2)]  # OR条件
```

> **非常に重要な注意点:**
>
> 1. 各条件式は必ず `()` **で囲んでください。**（例: `(df['col1'] > 10)`）。これはPythonの演算子の優先順位の問題です。
> 2. Pythonの `and` や `or` は使えません。Pandas（NumPy）専用の `&`, `|` を使用してください。

複数条件でのフィルタリング

使用例

mask1

職業が'経営者'

mask2

勤続年数が20年未満

```
# 職業が'経営者'で、かつ勤続年数が20年未満の顧客を抽出
df_customers[(df_customers['occupation'] == '経営者') &
             (df_customers['employment_years'] < 20)
             ].loc[:, ['customer_name', 'annual_income_10k_yen', 'employment_years']]
```

✓ 0.0s

	customer_name	annual_income_10k_yen	employment_years
36	田村 剛	900	19

SQLクエリぽい.query()による抽出

.query()メソッド

基本構文 (例)

基本的にはSQL文のWHERE以降の文を入力できます。
※pythonで有効な比較演算子とオブジェクトのみ利用可能
※列名を直接利用できます。

```
df.query('列名 == 値 and 列名2 < 値')
df.query('列名 in [値1, 値2]')
```

SQLクエリぽい.query()による抽出

使用例

```
df_customers.query(
    'occupation in ["会社員", "エンジニア"] and address.str.startswith("東京都")'
                ).loc[:, ['customer_name', 'address', 'occupation']].head()
```

✓ 0.0s

	customer_name	address	occupation
0	田中 太郎	東京都世田谷区成城6-1-1	会社員
16	清水 拓也	東京都江東区豊洲2-2-1	エンジニア
27	大野 真由美	東京都豊島区池袋1-1	会社員
33	前田 裕子	東京都品川区大井1-1	会社員
35	河野 明日香	東京都目黒区自由が丘1-1	エンジニア

目次

データの削除 (drop)

基本構文

```
# 列の削除
df.drop(columns=['列名A', '列名B'])
# 行の削除
df.drop(index=[行番号1, 行番号2])
df.drop(index=[開始行:終了行])
```

> **重要な概念: `inplace`引数** Pandasの`drop`は、デフォルトでは元のデータを変更せず、**削除後の新しいDataFrameを返します**（非破壊的処理）。 元の変数 `df_customers` を更新したい場合は、以下のいずれかを行います。
>
> 1. **再代入**: `df = df.drop(...)`
> 2. **引数指定**: `df.drop(..., inplace=True)`

列の削除

```
# 個人情報を削除した新しいDFを作成
df_dropped_cols = df_customers.drop(columns=['phone_number', 'email_address','address','birth_date'])
df_dropped_cols.head()
```

	customer_id	customer_name	customer_name_kana	gender	occupation	employer_name	employment_years
0	1	田中 太郎	タナカ タロウ	男性	会社員	株式会社東京商事	8
1	2	佐藤 花子	サトウ ハナコ	女性	専門職	医療法人梅田クリニック	3
2	3	山本 健太	ヤマモト ケンタ	男性	自営業	山本自動車整備工場	15
3	4	小林 美咲	コバヤシ ミサキ	女性	学生	アルバイト(カフェ)	1
4	5	加藤 誠	カトウ マコト	男性	経営者	加藤ホールディングス	30

annual_income_10k_yen	latest_application_purpose	registration_date	main_branch_code	main_account_number
650	住宅購入	2023-01-15	1	1000001
480	教育資金	2023-02-20	1	1000002
550	運転資金	2022-11-05	2	1000003
150	趣味・レジャー	2023-03-10	3	1000004

行の削除

使用例

```
# テスト用データである可能性のある、行ラベル0, 2, 4の顧客を削除
# 注意: ここでは元の df_customers から削除を行っています
df_dropped_rows = df_customers.drop(index=[0, 2, 4])
df_dropped_rows.loc[:, ['customer_name','employer_name','annual_income_10k_yen']].head()
```

✓ 0.0s

	customer_name	employer_name	annual_income_10k_yen
1	佐藤 花子	医療法人梅田クリニック	480
3	小林 美咲	アルバイト(カフェ)	150
5	吉田 恵美	千葉県庁	580
6	山田 雅也	株式会社先端技術開発	850
7	斎藤 結衣	コンビニエンスストア	250

総合ハンズオン: フィルタリング

データフィルタリング

データベース操作

タスク

依頼1（情報管理部より）： 「顧客データ管理の一元化企画に伴い、bankdataのデータベースの顧客テーブルcustomersにB支店_customers.csvのテーブルを追加してください。
※以降はbankdataのcustomersテーブルのデータを利用して依頼に対応します。

依頼1（マーケティング部より）：「富裕層向けの新商品案内のため、**年収が1500万円以上**で、かつ**勤続年数が10年以上**の顧客リストを作成してください。リストには、customer_id, customer_name, occupation, annual_income_10k_yen, email_address, phone_number のみを含めてください。」

依頼2（コンプライアンス部より）：「与信審査モデルの検証のため、**職業が'学生'または'無職'**の顧客リストが必要です。ただし、個人情報保護の観点から、address, phone_number, email_address の列は除外して提出してください。」