レポート提出票

科目名: 情報工学実験3

実験課題名: 課題2 パターン認識

実施日: 2021年 6月 3日

学籍番号: 4619055

氏名: 辰川力駆

共同実験者:

© 2020-21 東京理科大学 工学部 情報工学科

1　実験の要旨

データ分析にとって必要不可欠である前処理を演習や課題を通して理解し、Python を用いて実際に行うことでやり方を身に付ける。

2　実験の目的

演習や課題を通して、抽出や集約などの基本的なデータ前処理の仕方を理解する。

3　実験の原理

データ分析のためには主に、3 つの前処理がある。

1. 「指標、表、グラフ作成」のための前処理

   指標の計算、表やグラフに簡単に変換できるようなデータを準備することが目的である。必要な列 (Column) を全て持ち、扱いやすい単位で集約された行 (Row) が必要な範囲分存在するデータを作成する。

2. 「教師なし学習」の入力のための前処理

   教師なし機械学習モデルに利用する説明変数を持つデータを準備することが目的である。1 と同様なデータであるが、機械学習モデルの種類に応じて扱いやすいデータの型に変換したり、データのスケールを揃えたりする前処理が必要になる。

3. 「教師あり学習」の入力のための前処理

   教師あり機械学習モデルに利用する学習データ、テストデータ、そしてモデルに適用するデータの準備をすることが目的である。1 や 2 と同様の前処理だけでなく、例えば、複数の列を組み合わせて新たな特徴量に変換する前処理などが必要になることがある。

こういった前処理の目的や役割があるが、実際に前処理を進めていくにあたっての流れは、データ構造を対象とした前処理とデータ内容を対象とした前処理に分けている。

1. データ構造を対象とした前処理

   複数の行にまたがったデータ全体に及ぶ処理であり、前処理の中でも早い段階で実施されることが多い。例えば、特定のデータを抽出したり、データを結合したり、あるルールに従って複数の行を 1 行にしたりする。

2. データ内容を対象とした前処理

   行ごとのデータの値に応じた処理で、行ごとに独立で処理できる。比較的規模の小さい処理で、条件を変えて繰り返されることが多い。例えば、複数の列を組み合わせて新たな列を生成したり、日時のデータを月毎のデータに集約したりする。

4　実験方法

Python を Jupyter Notebook 上で実行することで、今回の実験を行う。

4.1　課題 1

reserve_tb から 50 % の確率で顧客をサンプリングする。サンプリングした各顧客ごとの予約件数を調査して平均を求める。顧客のサンプリングに関して以下の 2 つのやり方で行う。

reserve_tb の行を 50 % サンプリングした後に顧客を重複なく列にした場合と、reserve_tb から重複のない顧客の列を抽出し 50 % サンプリングをした場合で予約件数の平均がどの様に変わるかを確認し考察する。

4.2　課題 2

reserve_tb、customer_tb を使用する。2016 年 1 月 1 日から 2016 年 12 月 31 日までにされた男性客による予約 (reserve_datetime が 2016-01-01 から 2016-12-31) の total_price の最大値をホテルごとに算出する (指定された期間に男性客による予約のなかったホテルは除外する)。最も total_price の最大値が高かったホテルから上位 10 件をランキング形式で降順に表示する。同じ件数の場合は同じ順位で表示する。

4.3　課題 3

アヤメデータを実験で使ったのとは違う特徴量の組み合わせで基データ、標準化したデータ、正規化したデータのプロットを行う。プロットした図から読み取れることを考察する。

4.4　課題 4

reserve_tb を使用する。ある顧客の予約の行にその顧客の過去 90 日間の予約の total_price の平均を (total_price_90d) という名前の列で情報として付与する。過去 90 日間に予約がなかった場合は 0 とする。日にちは予約のされた日 (reserve_datetime) を基準にする。

4.5　結果・検討・考察

4.6　課題 1

作成したプログラムは付録 A のソースコード 1,2 に載せた。実行結果は次のようになった。

(reserve_tb の行を 50 %サンプリングした後に顧客を重複なく列にした場合) ＝ 2.61010
(reserve_tb から重複のない顧客の列を抽出し 50 %サンプリングをした場合) ＝ 4.69369

このような差が生まれた理由は、サンプルされる顧客の数が違うことに原因があると考える。後者の場合の方が顧客の数がすくなくなるので相対的に一人あたりの数が増えると考える。

4.7　課題 2

作成したプログラムは付録 A のソースコード 3 に載せた。実行結果は次の図 1 のようになった。price_max の列を消そうか考えたが、あるほうが自然なので残した。

	hotel_id	price_max	price_max_rank
38	h_134	811200	1.0
142	h_23	781200	2.0
65	h_159	776400	3.0
88	h_180	654000	4.0
161	h_247	640800	5.0
84	h_177	630900	6.0
278	h_83	559800	7.0
185	h_269	540800	8.0
239	h_48	540000	9.0
68	h_161	535200	10.0

図 1: 課題 2 の結果

4.8　課題 3

作成したプログラムは付録 A のソースコード 4 に載せた。実行結果は次の図 2 のようになった。

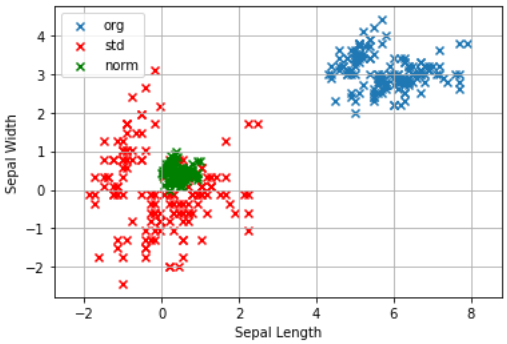

図 2: 課題 3 の結果

正規化したデータを見ると、当たり前であるが元データを正規化したことによりデータが 0 から 1 の範囲に収まっていることが分かる。

また、どのデータを見ても分かるが、相関はあまり強くない特徴量の組合せであることがわかる。

4.9　課題 4

作成したプログラムは付録 A のソースコード 5 に載せた。実行結果は次の図 3 のようになった。

	reserve_id	hotel_id	customer_id	reserve_datetime	checkin_date	checkin_time	checkout_date	people_num	total_price	total_price_90d
0	r1	h_75	c_1	2016-03-06 13:09:42	2016-03-26	10:00:00	2016-03-29	4	97200	0.0
1	r2	h_219	c_1	2016-07-16 23:39:55	2016-07-20	11:30:00	2016-07-21	2	20600	0.0
2	r3	h_179	c_1	2016-09-24 10:03:17	2016-10-19	09:00:00	2016-10-22	2	33600	20600.0
3	r4	h_214	c_1	2017-03-08 03:20:10	2017-03-29	11:00:00	2017-03-30	4	194400	0.0
4	r5	h_16	c_1	2017-09-05 19:50:37	2017-09-22	10:30:00	2017-09-23	3	68100	0.0
5	r6	h_241	c_1	2017-11-27 18:47:05	2017-12-04	12:00:00	2017-12-06	3	36000	68100.0
6	r7	h_256	c_1	2017-12-29 10:38:36	2018-01-25	10:30:00	2018-01-28	1	103500	36000.0
7	r8	h_241	c_1	2018-05-26 08:42:51	2018-06-08	10:00:00	2018-06-09	1	6000	0.0
8	r9	h_217	c_2	2016-03-05 13:31:06	2016-03-25	09:30:00	2016-03-27	3	68400	0.0
9	r10	h_240	c_2	2016-06-25 09:12:22	2016-07-14	11:00:00	2016-07-17	4	320400	0.0
10	r11	h_183	c_2	2016-11-19 12:49:10	2016-12-08	11:00:00	2016-12-11	1	29700	0.0
11	r12	h_268	c_2	2017-05-24 10:06:21	2017-06-20	09:00:00	2017-06-21	4	81600	0.0
12	r13	h_223	c_2	2017-10-19 03:03:30	2017-10-21	09:30:00	2017-10-23	1	137000	0.0
13	r14	h_133	c_2	2018-02-18 05:12:58	2018-03-12	10:00:00	2018-03-15	2	75600	0.0
14	r15	h_92	c_2	2018-04-19 11:25:00	2018-05-04	12:30:00	2018-05-05	2	68800	75600.0

図 3: 課題 4 の結果

5　まとめ

パターン認識とは、入力データをクラス分けする処理のことであることを学んだ。また、実際に Python でデータ前処理を行って、抽出や集約などの基本的なデータ前処理の仕方を理解することができた。

A 付録

ソースコード 1: 課題 1(1)

```
##課題 1(reserve_tb の行を 50 %サンプリングした後に顧客を重複なく列にした場合)
import numpy as np
import pandas as pd
reserve_tb = pd.read_csv('data/reserve.csv', encoding='UTF-8')

reserve_tb = reserve_tb.sample(frac=0.5)
target = reserve_tb['customer_id'].unique()

kadai1_rslt = reserve_tb[reserve_tb['customer_id'].isin(target)].groupby('customer_id'
    ).size().reset_index()

kadai1_rslt.columns = ['customer_id','rsv_cnt']

print(kadai1_rslt['rsv_cnt'].mean())
```

ソースコード 2: 課題 1(2)

```
##課題 1(reserve_tb から重複のない顧客の列を抽出し 50 %サンプリングをした場合)
import numpy as np
import pandas as pd
reserve_tb = pd.read_csv('data/reserve.csv', encoding='UTF-8')

target = pd.Series(reserve_tb['customer_id'].unique()).sample(frac=0.5)
kadai1_rslt = reserve_tb[reserve_tb['customer_id'].isin(target)].groupby('customer_id'
    ).size().reset_index()

kadai1_rslt.columns = ['customer_id','rsv_cnt']

print(kadai1_rslt['rsv_cnt'].mean())
```

ソースコード 3: 課題 2

```
##課題 2
import numpy as np
import pandas as pd
reserve_tb = pd.read_csv('data/reserve.csv', encoding='UTF-8')
customer_tb = pd.read_csv('data/customer.csv',encoding='UTF-8')

kadai2_rslt = pd.merge(reserve_tb.query('"2016-01-01" <= reserve_datetime <= "2016-12-31"'),customer_tb.query('sex == "man"'),
                       on='customer_id',how='inner')
kadai2_rslt = kadai2_rslt.groupby('hotel_id').agg({'total_price':['max']}).reset_index()
kadai2_rslt.columns = ['hotel_id','price_max']
kadai2_rslt['price_max_rank'] = kadai2_rslt['price_max'].rank(ascending=False,method='min')
kadai2_rslt = kadai2_rslt.sort_values('price_max_rank', ascending=True)
kadai2_rslt.head(10)
```

ソースコード 4: 課題 3

```
##課題 3
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
mmsc = MinMaxScaler()
from sklearn.preprocessing import StandardScaler
stdsc = StandardScaler()
from sklearn.datasets import load_iris
iris = load_iris()

x1 = iris.data[:,0].reshape(-1,1)
x2 = iris.data[:,1].reshape(-1,1)

x=np.concatenate((x1,x2), axis=1)
plt.axis('equal')
plt.scatter(x[:,0],x[:,1], marker="x", label="org")
x_norm = mmsc.fit_transform(x)
x_std = stdsc.fit_transform(x)
plt.scatter(x_std[:,0],x_std[:,1], c='red',marker="x", label="std")
plt.scatter(x_norm[:,0],x_norm[:,1], c='green',marker="x", label="norm")
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.legend()
plt.grid()
plt.show()
```

ソースコード 5: 課題 1(2)

```
##課題 4
import numpy as np
import pandas as pd
reserve_tb = pd.read_csv('data/reserve.csv', encoding='UTF-8')

reserve_tb['reserve_datetime'] = pd.to_datetime(reserve_tb['reserve_datetime'])

kadai4_rslt = pd.merge(reserve_tb.loc[:, ['reserve_id','customer_id','
    reserve_datetime']],\
                       reserve_tb.loc[:, ['customer_id','reserve_datetime', '
                           total_price']],on='customer_id',how='inner')
kadai4_rslt['diffday'] = (kadai4_rslt['reserve_datetime_x'] - kadai4_rslt['
    reserve_datetime_y']).dt.days
kadai4_rslt = kadai4_rslt.query('0 < diffday <= 90').groupby('reserve_id')['
    total_price'].mean().reset_index()
kadai4_rslt.columns = ['reserve_id', 'total_price_90d']
kadai4_rslt = pd.merge(reserve_tb,kadai4_rslt,on='reserve_id',how='left').fillna(0)
kadai4_rslt.head(15)
```