Python 実践データ分析 100 本ノック

【正誤表】

●21ページ 4~5 行目

【誤】 実行すると、5000、1737、6737 と出力されるはずです。5000 と 1737 を足して、6737 となりますので 【正】 実行すると、5000、1786、6786 と出力されるはずです。5000 と 1786 を足して、6786 となりますので

●22ページ 上段のコード 3 行目

【誤】print(len(transaction))

【正】(この行、不要につき削除)

●22ページ 下から3行目

【誤】実行すると、7144、6737、7144と出力されます。

【正】実行すると、7144、67<mark>86</mark>、7144と出力されます。

●33ページ 1 行目

【コード挿入】

本書は、「自分でやってみましょう!」の精神に基づいて、全ての実習コードを掲載してはいません。また、コードは図 1-13 に表示されていますが、本文に掲載されていないのはわかりにくいという声もありますので、本文に次のコードを挿入します (第4刷で追加)。

graph data = pd.pivot table(join data, index='payment month', columns='item name', values='price', aggfunc='sum') graph_data.head()

●95ページ 8 行目

【誤】 ccustomer_clustering.groupby("cluster").mean() 【正】 customer_clustering.groupby("cluster").mean()

●96ページ コードの1行目

【誤】 cfrom sklearn.decomposition import PCA 【正】 from sklearn.decomposition import PCA

●98 ページ コードの最終行

【誤】 <mark>customer_join.head()</mark> 【正】 (この行、不要につき削除)

●102ページ 1~2 行目

【誤】これは、4ヶ月で退会してしまったためです。

【正】これは、まだ入会してからの期間が短く、デ -タが存在し ないためです。

●107ページ 2~4 行目

【誤】1人目は、6ヶ月前から1ヶ月毎に8回、7回、8回、6回、 4回、4回、3回来ている顧客で、2人目は、8回、6回、4 回、3回、3回、2回、2回来ている顧客の翌月の来店回 数を予測します。 【正】1人目は、6ヶ月前から1ヶ月毎に7回、8回、6回、4回、

4回、3回来ている顧客で、2人目は、6回、4回、3回、3 回、2回、2回来でいる<mark>顧客で、どちらも8ヶ月の在籍期間の</mark>顧客の翌月の来店回数を予測します。

●129 ページ 中段コードの 9~12 行目 【誤】 elif campaign_name == "デイタイム": campaign_name_list = [0, 1] elif campaign_name == "ナイト": campaign name list = [0, 0]

【正】 elif class name == "デイタイム": class_name_list = [0, 1] elif class_name == "ナイト": class_name_list = [0, 0]

●130ページ 5-18:退会の予測結果

【訂正】129ページの修正に伴い、右上の図と差し替えます。

●142 ページ 図 6-7:輸送実績の総コスト集計結果 【訂正】 提供データ「tbl_transaction.csv」に誤りがありました。このデータを修正後の集計結果は下の図のようになります。

```
In [10]: # 支社のコスト合計を算出
          print("関東支社の総コスト: " + str(kanto["Cost"].sum()) + "万円")
print("東北支社の総コスト: " + str(tohoku["Cost"].sum()) + "万円")
          関東支社の総コスト: 2189.3万円
          東北支社の総コスト: 2062.0万円
```

```
ノック50:顧客の退会を予測しよう
In [21]: count_1 = 3
        routing_flg = 1
        period = 10
        campaign_name = "入会費無料"
        class_name = "オールタイム"
        gender = "M"
In [22]: if campaign_name == "入会費半額":
          campaign_name_list = [1, 0]
        elif campaign_name == "入会費無料":
         campaign_name_list = [0, 1]
        elif campaign_name == "通常":
         campaign_name_list = [0, 0]
        if class_name == "オールタイム":
          class_name_list = [1, 0]
        elif class_name == "デイタイム":
         class_name_list = [0, 1]
        elif class_name == "ナイト":
         class_name_list = [0, 0]
        if gender == "F":
          gender_list = [1]
        elif gender == "M":
          gender_list = [0]
        input_data = [count_1, routing_flg, period]
        input_data.extend(campaign_name_list)
        input_data.extend(class_name_list)
        input_data.extend(gender_list)
In [23]: print(model.predict([input_data]))
        print(model.predict_proba([input_data]))
        [[0.01219512 0.98780488]]
```

●185 ページ 下のコードの最終行 【誤】 if df_links.iloc[i][j]==1: 【正】 node_name = "Node" + str(j) if df_links[node_name].iloc[i]==1:

●186ページ 図8-2:ネットワークの可視化

【訂正】185ページの修正に伴い、以下の図に差し替えます。

```
In [3]: import networkx as nx
          import matplotlib.pyplot as plt
          # グラフオブジェクトの作成
          G = nx.Graph()
          # 頂点の設定
          NUM = len(df_links.index)
          for i in range(1,NUM+1):
              node_no = df_links.columns[i].strip("Node")
G.add_node(str(node_no))
          # 辺の設定
          for i in range(NUM):
for i in range(NUM):
                   node_name = "Node" + str(j)
if df_links[node_name].iloc[i]==1:
    G.add_edge(str(i),str(j))
          # 描画
         nx.draw_networkx(G,node_color="k", edge_color="k", font_color="w")
         plt.show()
            0.6
            0.4
            0.2
            0.0
           -0.2
           -0.4
           -0.6
           -0.8
                  -1.00 -0.75 -0.50 -0.25 0.00 0.25 0.50 0.75
```

●187 ページ 三段目のコードの 5 行目

【誤】 if df_links.iloc[i][j]==1:

(正) node_name = "Node" + str(j) if df_links[node_name].iloc[i]==1:

●188ページの図 8-3:口コミ伝播の計算

【訂正】187ページの修正に伴い、差し替えます。

```
ノック72: ロコミによる情報伝播の様子を可視化してみよう
In [4]: import numpy as np
In [5]: def determine_link(percent):
                rand_val = np.random.rand()
                if rand_val<=percent:</pre>
                    return 1
                else:
                    return 0
In [6]: def simulate_percolation(num, list_active, percent_percolation):
               for i in range(num):
    if list_active[i]==1:
                         for j in range(num):
   node_name = "Node" + str(j)
                              if df_links[node_name].iloc[i]==1:
                                   if determine_link(percent_percolation)==1:
                                       list_active[j] = 1
                return list_active
In [7]: percent_percolation = 0.1 
T_NUM = 36 
NUM = len(df_links.index) 
list_active = np.zeros(NUM) 
list_active[0] = 1
           list_timeSeries = []
           for t in range(T_NUM):
               list_active = simulate_percolation(NUM, list_active, percent_percolation)
list_timeSeries.append(list_active.copy())
```

●190ページの図 8-4:口コミ伝播の可視化

【訂正】187ページの修正に伴い、以下の図に差し替えます。

```
In [8]: # アクティブノード可視化 # def active_node_coloring(list_active):
                 #print(list_timeSeries[t])
list_color = []
                 for i in range(len(list_timeSeries[t])):
    if list_timeSeries[t][i]==1:
        list_color.append("r")
                           list_color.append("k")
                 #print(len(list_color))
return list_color
 In [9]: # 描画
            t = 0
            nx.draw_networkx(G,font_color="w",node_color=active_node_coloring(list_timeSeries[t]))
               1.00
               0.75
               0.50
               0.25
               0.00
              -0.25
              -0.50
              -0.75
                                 -0.2
                                         0.0
                                                 0.2
                                                         0.4
                                                                        0.8
                                                                 0.6
In [10]: # 描画
            nx.draw_networkx(G,font_color="w",node_color=active_node_coloring(list_timeSeries[t]))
            plt.show()
               0.8
               0.6
               0.4
               0.2
               0.0
              -0.2
              -0.4
             -0.6
              -0.8
                   -1.0
                                              0.0
                                                           0.5
                                                                        1.0
In [11]: # 描画
t = 35
            nx.draw_networkx(G,font_color="w",node_color=active_node_coloring(list_timeSeries[t]))
            plt.show()
               0.6
               0.4
               0.2
               0.0
              -0.2
              -0.4
              -0.6
              -0.8
                                                                       1.0
                     -1.0
                                  -0.5
                                                           0.5
                                              0.0
```

<本書紹介サイト>

https://www.shuwasystem.co.jp/book/9784798058757.html

<秀和システム>

http://www.shuwasystem.co.jp/