

Python 実践データ分析 100 本ノック 【正誤表】

(第1版第8刷以降では以下の訂正は修正済)

●16 ページ 本文 下から6行目

【誤】 `transaction_dateil_1.csv`、

【正】 `transaction_detail_1.csv`、

●17 ページ 表内

【誤】 4-1: `transaction_deatil_1.csv`

4-2: `transaction_deatil_2.csv`

【正】 4-1: `transaction_detail_1.csv`

4-2: `transaction_detail_2.csv`

●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が、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が、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、6786、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 3

【誤】 `from sklearn.decomposition import PCA`

【正】 `from sklearn.decomposition import PCA`

●98 ページ コードの最終行

【誤】 `customer_join.head()`

【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、デー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回来ている顧客で、どちらも8ヶ月の在籍期間の顧客の翌月の来店回数を予測します。

●128 ページ 最下段のコード 2 行目

【誤】 `routing_flg = 1`

【正】 `routine_flg = 1`

●129 ページ 中段コードの 9~12 行目

【誤】 `elif campaign_name == "デイトタイム":`

`campaign_name_list = [0, 1]`

`elif campaign_name == "ナイト":`

`campaign_name_list = [0, 0]`

【正】 `elif class_name == "デイトタイム":`

`class_name_list = [0, 1]`

`elif class_name == "ナイト":`

`class_name_list = [0, 0]`

●129 ページ 中段コード 下から 4 行目

【誤】 routing_flg

【正】 routine_flg

●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: 2082.0万円
```

●155 ページ 6-16: 輸送コストの計算

【訂正】 計算結果にまちがいがありません(156 ページの訂正参照)ので、以下の図と差し替えます。

■図6-16: 輸送コストの計算

ノック58: 輸送コスト関数を作成しよう

```
In [145]: import pandas as pd

# データ読み込み
df_tr = pd.read_csv('trans_route.csv', index_col="工場")
df_tc = pd.read_csv('trans_cost.csv', index_col="工場")

# 輸送コスト関数
def trans_cost(df_tr, df_tc):
    cost = 0
    for i in range(len(df_tc.index)):
        for j in range(len(df_tr.columns)):
            cost += df_tr.iloc[i][j]*df_tc.iloc[i][j]
    return cost

print("総輸送コスト:"*str(trans_cost(df_tr, df_tc)))

総輸送コスト:1493
```

■5-18: 退会の予測結果

ノック50: 顧客の退会を予測しよう

```
In [21]: count_1 = 3
routine_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 == "デイトタイム":
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, routine_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]))

[1.]
[[0.00278552 0.99721448]]
```

●156 ページ 本文 1 行目

【誤】 総輸送コストは 1433(万円)

【正】 総輸送コストは 1493(万円)

●166 ページ コード 6 行目

【誤】 df_tr = pd.read_csv('trans_route.csv', index_col="工場")

【正】 df_tr = df_tr_sol.copy()

●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、右の図に差し替えます。

●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:

●187 ページ 最下段のコード 2 行目

【誤】 T_NUM = 100

【正】 T_NUM = 36

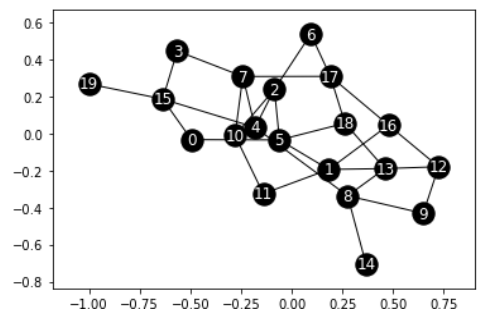
```
In [3]: import networkx as nx
import matplotlib.pyplot as plt

# 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 j 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()
```



●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、右の図に差し替えます。

●188 ページ 本文 下から 3 行目
 【誤】 100 ステップ繰り返す
 【正】 36 ステップ繰り返す

●188 ページ 本文 下から 2 行目
 【誤】 100 か月繰り返す
 【正】 36 か月繰り返す

●189 ページ 3 つ目のコード 1 行目
 【誤】 t = 10
 【正】 t = 11

●189 ページ 最下段のコード 1 行目
 【誤】 t = 99
 【正】 t = 35

●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、右の図に差し替えます。

●190 ページ 本文 下から 3 行目
 【誤】 t=0,10,99(0 か月後、10 か月後、99 か月後)
 【正】 t=0,11,35(0 か月後、11 か月後、35 か月後)

●200 ページ「拡散の確率推定」コード 11 行目
 【誤】 if (df_mem_info.iloc[df_link_temp.index[j]][t]==0):
 【正】 if (df_mem_info.iloc[df_link_temp.index[j]][str(t)]==0):

●200 ページ「拡散の確率推定」コード 14 行目
 【誤】 if (df_mem_info.iloc[df_link_temp.index[j]][t+1]==1):
 【正】 if (df_mem_info.iloc[df_link_temp.index[j]][str(t+1)]==1):

ノック72： 口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:
            return 1
        else:
            return 0

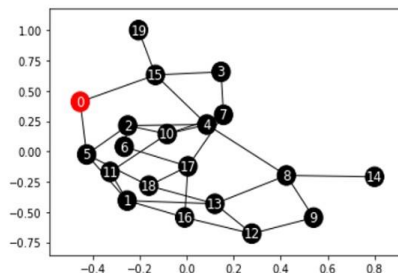
In [6]: def simulate_percolation(num, list_active, percent_percolation):
        for i in range(num):
            if list_active[i]==1:
                for j in range(num):
                    node_name = "Node" + str(j)
                    if df_links[node_name].iloc[i]==1:
                        if 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

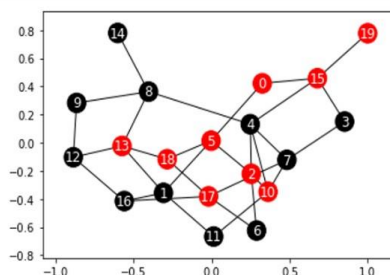
        list_timeSeries = []
        for t in range(T_NUM):
            list_active = simulate_percolation(NUM, list_active, percent_percolation)
            list_timeSeries.append(list_active.copy())
```

```
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")
                else:
                    list_color.append("k")
            #print(len(list_color))
            return list_color
```

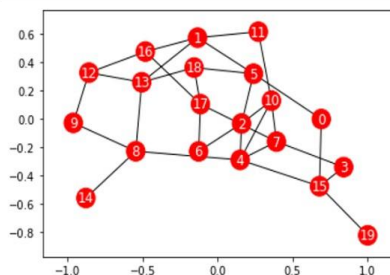
```
In [9]: # 描画
        t = 0
        nx.draw_networkx(G,font_color="w",node_color=active_node_coloring(list_timeSeries[t]))
        plt.show()
```



```
In [10]: # 描画
        t = 11
        nx.draw_networkx(G,font_color="w",node_color=active_node_coloring(list_timeSeries[t]))
        plt.show()
```



```
In [11]: # 描画
        t = 35
        nx.draw_networkx(G,font_color="w",node_color=active_node_coloring(list_timeSeries[t]))
        plt.show()
```



<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/>