Python 実践データ分析 100 本ノック 【正誤表】

●16～17 ページ 表 1-1：データー覧
【誤】 No.1：transaction_dateil_1.csv
No.4-1：transaction_deatil_1.csv
No.4-2：transaction_deatil_2.csv
【正】 No.1：transaction_detail_1.csv
No.4-1：transaction_detail_1.csv
No.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。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、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】cfrom 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。

●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
```

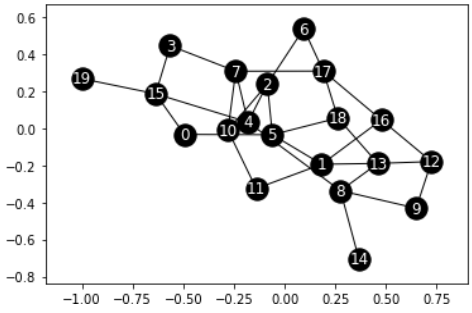
●155～156 ページ　ノック 58 の実行結果
【誤】 総輸送コスト：1433(万円)
【正】 総輸送コスト：1493(万円)

●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。

```
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()
```



●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、以下の図に差し替えます。

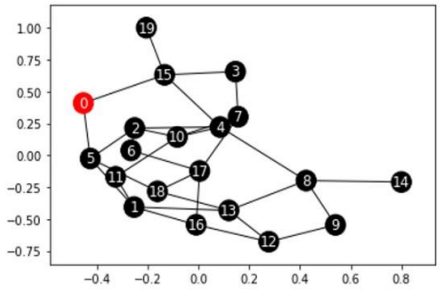
ノック72：口コミによる情報伝播の様子を可視化してみよう

```
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())
```

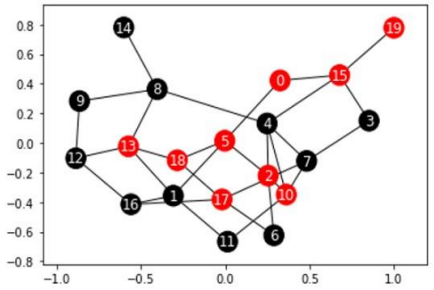
●190 ページの図 8-4：口コミ伝播の可視化
【訂正】 187 ページの修正に伴い、以下の図に差し替えます。

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

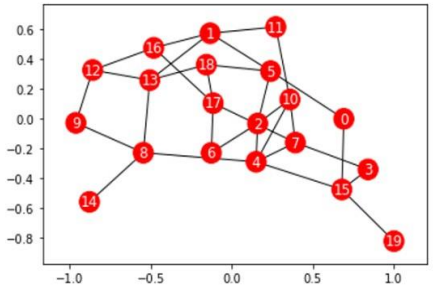
In [9]:
```
# 描画
t = 0
nx.draw_networkx(G,font_color="w",node_color=active_node_coloring(list_timeSeries[t]))
plt.show()
```



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()
```



●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):

＜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/