Lorenzo Rega

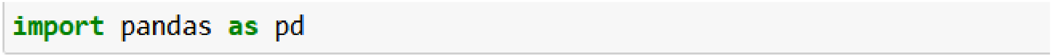
Lr3103

Homework 3

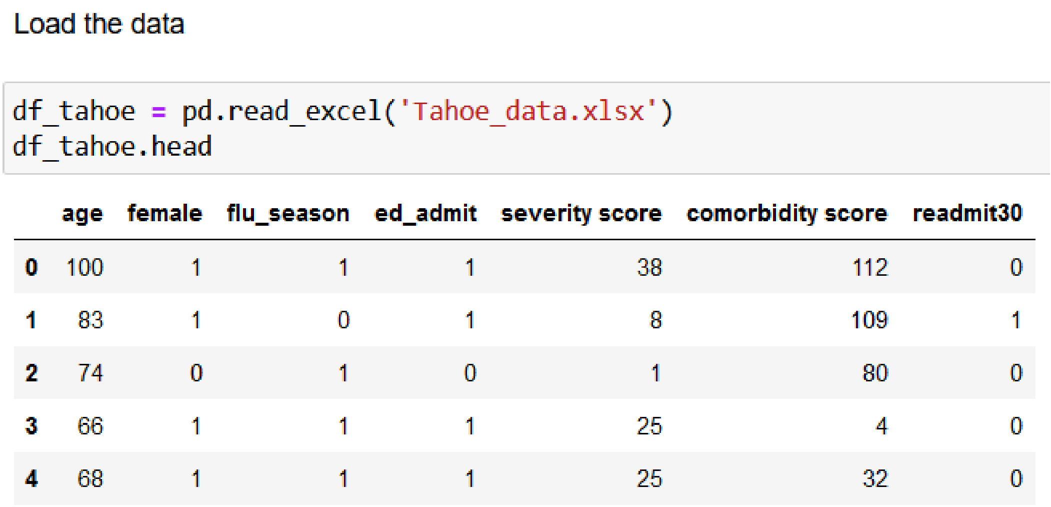
Exercise 1

a)

We first load the data using pandas:



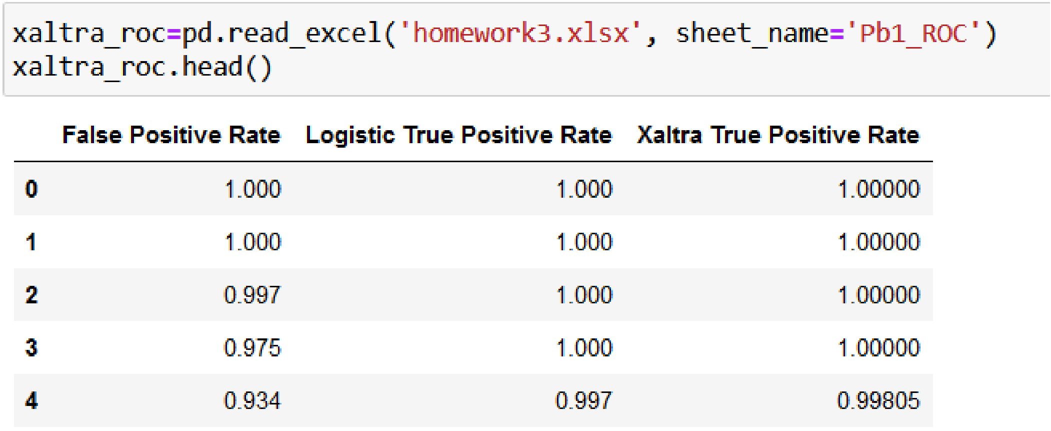
```
import pandas as pd
```



Load the data

```
df_tahoe = pd.read_excel('Tahoe_data.xlsx')
df_tahoe.head
```

	age	female	flu_season	ed_admit	severity score	comorbidity score	readmit30
0	100	1	1	1	38	112	0
1	83	1	0	1	8	109	1
2	74	0	1	0	1	80	0
3	66	1	1	1	25	4	0
4	68	1	1	1	25	32	0

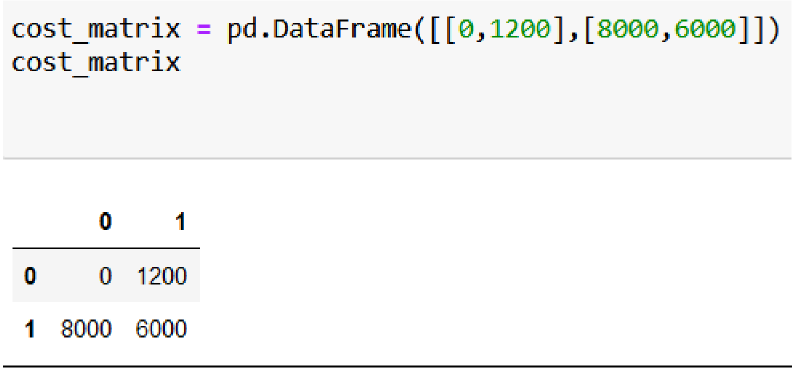


```
xaltra_roc=pd.read_excel('homework3.xlsx', sheet_name='Pb1_ROC')
xaltra_roc.head()
```

	False Positive Rate	Logistic True Positive Rate	Xaltra True Positive Rate
0	1.000	1.000	1.00000
1	1.000	1.000	1.00000
2	0.997	1.000	1.00000
3	0.975	1.000	1.00000
4	0.934	0.997	0.99805

To determine the estimated total cost, we need both the confusion matrix and the cost matrix.

The cost matrix was determined in class and is:

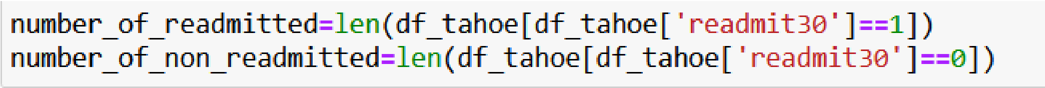


```
cost_matrix = pd.DataFrame([[0,1200],[8000,6000]])
cost_matrix
```

	0	1
0	0	1200
1	8000	6000

We must now compute the confusion matrix.

First, let's calculate the total number of readmitted and non-readmitted patients:



```
number_of_readmitted=len(df_tahoe[df_tahoe['readmit30']==1])
number_of_non_readmitted=len(df_tahoe[df_tahoe['readmit30']==0])
```

From the ROC, we have the False Positive Rate (FPR) and the True Positive Rate (TPR) for each threshold.

The FPR is defined as # false positives/ number of non-readmitted patients, and the TPR is defined as #true positives/ number of readmitted patients.

The total number of False Positives is: FPR* number of non-readmitted patients,

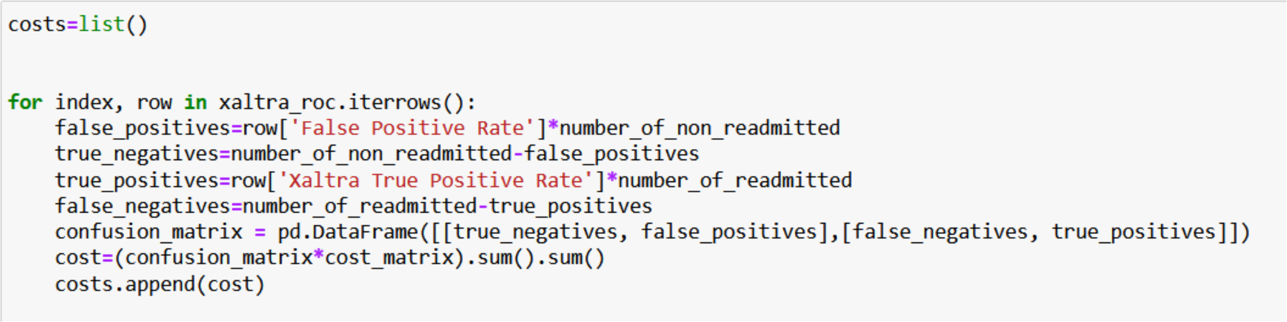
The total number of True Negatives is: number of non-readmitted patients - false positives,

The total number of True Positives is : TPR* number of readmitted patients,

The total number of False Negatives is: number of readmitted patients - True Positives

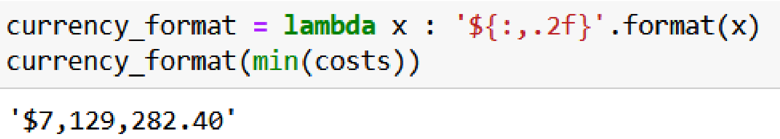
To determine the cost, we multiply the confusion matrix with the cost matrix.

We are going to do this for each pair or (FPR, TPR), getting as a result, a list of costs.



```
costs=list()


for index, row in xaltra_roc.iterrows():
    false_positives=row['False Positive Rate']*number_of_non_readmitted
    true_negatives=number_of_non_readmitted-false_positives
    true_positives=row['Xaltra True Positive Rate']*number_of_readmitted
    false_negatives=number_of_readmitted-true_positives
    confusion_matrix = pd.DataFrame([[true_negatives, false_positives],[false_negatives, true_positives]])
    cost=(confusion_matrix*cost_matrix).sum().sum()
    costs.append(cost)
```

Tahoe would have for sure used the best tradeoff between FPR and TPR by picking the minimum cost of this list, which is:

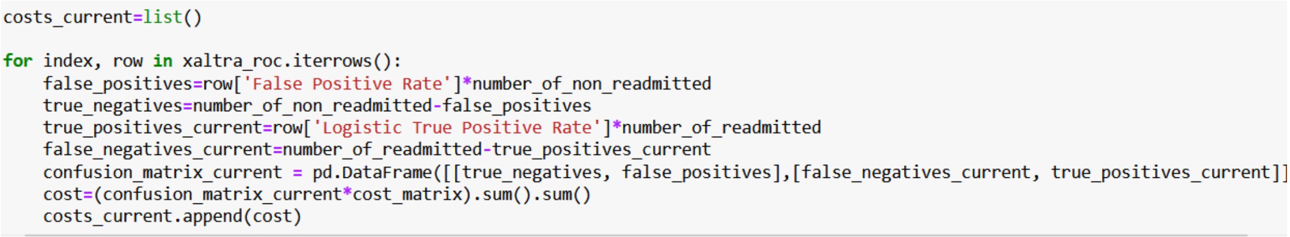


```
currency_format = lambda x : '${:,.2f}'.format(x)
currency_format(min(costs))
```

```
'$7,129,282.40'
```

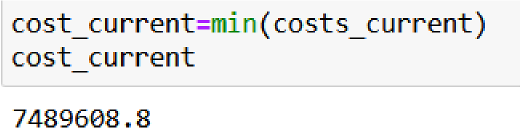
b)

To understand the reduction in cost, we must do the same thing we did before, using the Logistic TPR this time:



```
costs_current=list()

for index, row in xaltra_roc.iterrows():
    false_positives=row['False Positive Rate']*number_of_non_readmitted
    true_negatives=number_of_non_readmitted-false_positives
    true_positives_current=row['Logistic True Positive Rate']*number_of_readmitted
    false_negatives_current=number_of_readmitted-true_positives_current
    confusion_matrix_current = pd.DataFrame([[true_negatives, false_positives],[false_negatives_current, true_positives_current]]
    cost=(confusion_matrix_current*cost_matrix).sum().sum()
    costs_current.append(cost)
```

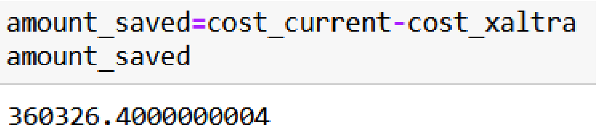
We obtain the cost:



```
cost_current=min(costs_current)
cost_current
```

```
7489608.8
```

If Tahoe had used the Xaltra system for prediction, they would have saved around $360, 326.



```
amount_saved=cost_current-cost_xaltra
amount_saved
```

```
360326.4000000004
```

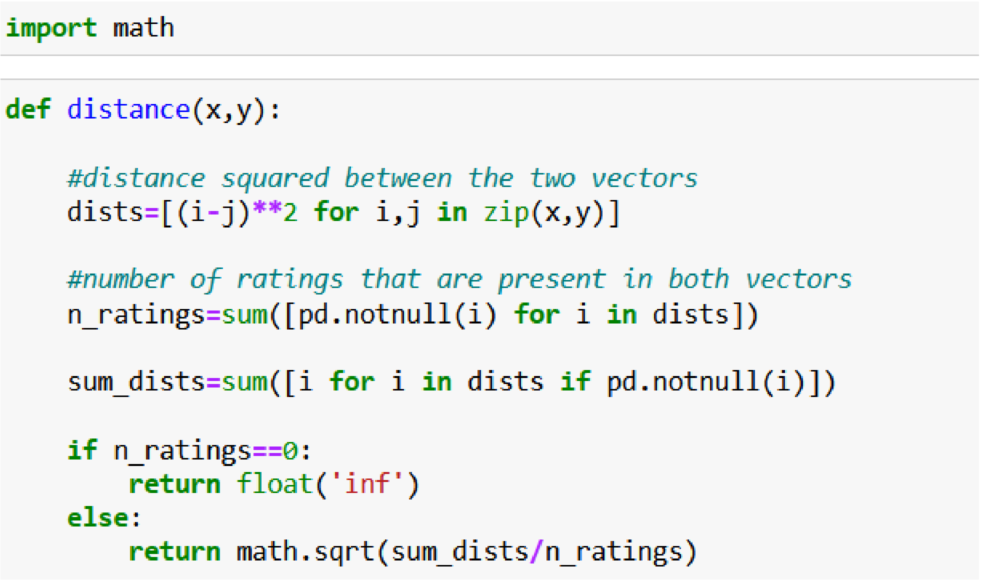
The Xaltra system has a lifetime of 10 years. The cost of the system every year is then $135,000 for maintenance and support + ($250,000)/10 = $25,000 for depreciation.

Total cost per year of the system is then $160,000, which is way less than the cost Tahoe is saving, so the applied charges by Xaltra are totally justified.

Exercise 2:

a)

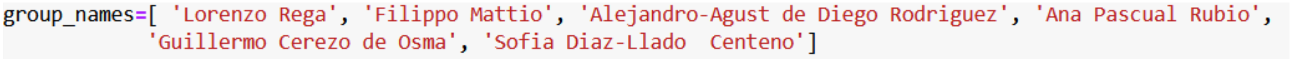
We then create a function distance which let's us compute the distance between two vectors (that represent the preferences of two people):



```
import math
```

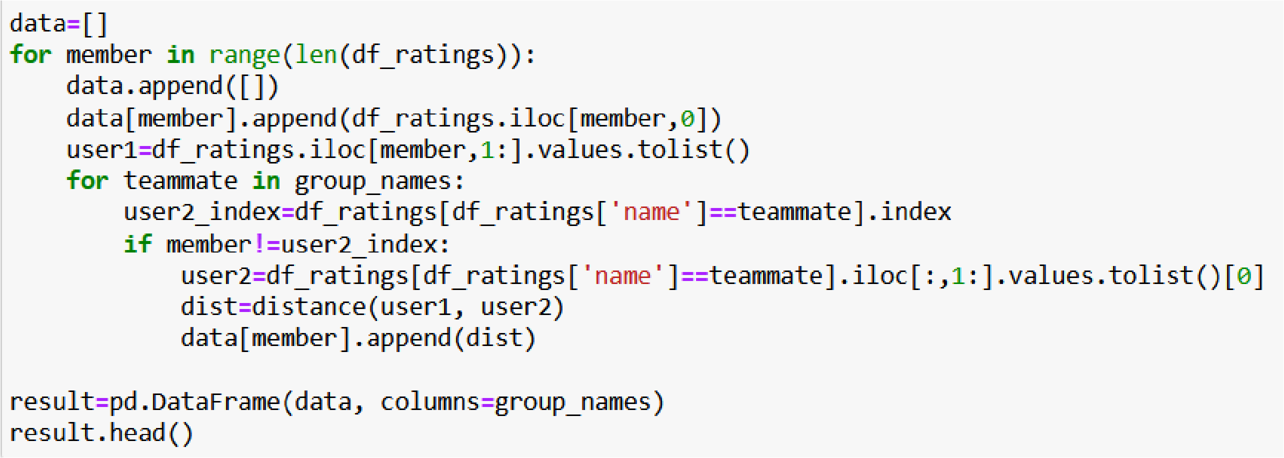
```
def distance(x,y):

    #distance squared between the two vectors
    dists=[(i-j)**2 for i,j in zip(x,y)]

    #number of ratings that are present in both vectors
    n_ratings=sum([pd.notnull(i) for i in dists])

    sum_dists=sum([i for i in dists if pd.notnull(i)])

    if n_ratings==0:
        return float('inf')
    else:
        return math.sqrt(sum_dists/n_ratings)
```

We select our group:

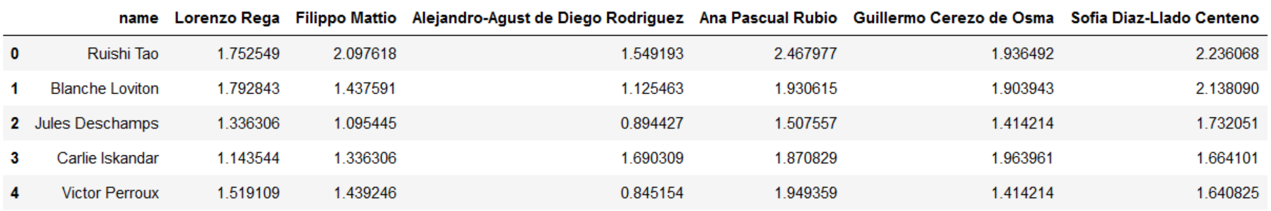


```
group_names=[ 'Lorenzo Rega', 'Filippo Mattio', 'Alejandro-Agust de Diego Rodriguez', 'Ana Pascual Rubio',
             'Guillermo Cerezo de Osma', 'Sofia Diaz-Llado  Centeno']
```

We go through the whole dataset and for each person, we find the distance with every member of the group (which represents a column):

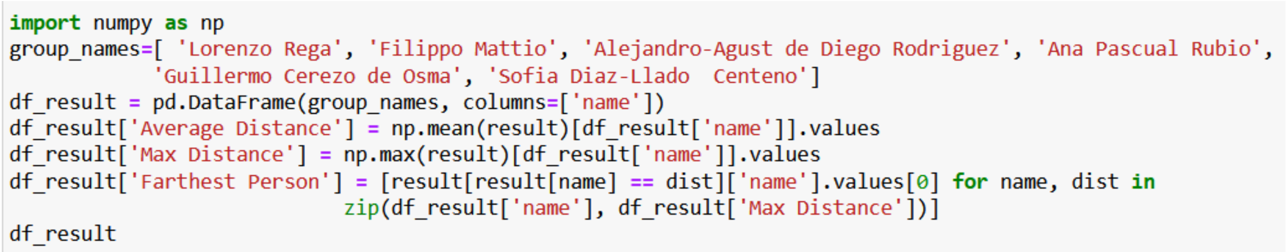


```
data=[]
for member in range(len(df_ratings)):
    data.append([])
    data[member].append(df_ratings.iloc[member,0])
    user1=df_ratings.iloc[member,1:].values.tolist()
    for teammate in group_names:
        user2_index=df_ratings[df_ratings['name']==teammate].index
        if member!=user2_index:
            user2=df_ratings[df_ratings['name']==teammate].iloc[:,1:].values.tolist()[0]
            dist=distance(user1, user2)
            data[member].append(dist)

result=pd.DataFrame(data, columns=group_names)
result.head()
```

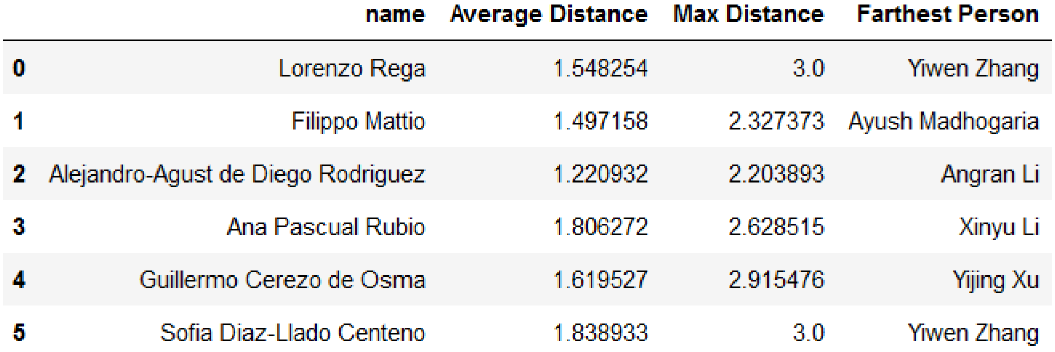


	name	Lorenzo Rega	Filippo Mattio	Alejandro-Agust de Diego Rodriguez	Ana Pascual Rubio	Guillermo Cerezo de Osma	Sofia Diaz-Llado Centeno
0	Ruishi Tao	1.752549	2.097618	1.549193	2.467977	1.936492	2.236068
1	Blanche Loviton	1.792843	1.437591	1.125463	1.930615	1.903943	2.138090
2	Jules Deschamps	1.336306	1.095445	0.894427	1.507557	1.414214	1.732051
3	Carlie Iskandar	1.143544	1.336306	1.690309	1.870829	1.963961	1.664101
4	Victor Perroux	1.519109	1.439246	0.845154	1.949359	1.414214	1.640825

In the end, we just get the average distance that every member of the group has with the entire class and see who the farthest person from them is:



```
import numpy as np
group_names=[ 'Lorenzo Rega', 'Filippo Mattio', 'Alejandro-Agust de Diego Rodriguez', 'Ana Pascual Rubio',
             'Guillermo Cerezo de Osma', 'Sofia Diaz-Llado  Centeno']
df_result = pd.DataFrame(group_names, columns=['name'])
df_result['Average Distance'] = np.mean(result)[df_result['name']].values
df_result['Max Distance'] = np.max(result)[df_result['name']].values
df_result['Farthest Person'] = [result[result[name] == dist]['name'].values[0] for name, dist in
                            zip(df_result['name'], df_result['Max Distance'])]
df_result
```



	name	Average Distance	Max Distance	Farthest Person
0	Lorenzo Rega	1.548254	3.0	Yiwen Zhang
1	Filippo Mattio	1.497158	2.327373	Ayush Madhogaria
2	Alejandro-Agust de Diego Rodriguez	1.220932	2.203893	Angran Li
3	Ana Pascual Rubio	1.806272	2.628515	Xinyu Li
4	Guillermo Cerezo de Osma	1.619527	2.915476	Yijing Xu
5	Sofia Diaz-Llado Centeno	1.838933	3.0	Yiwen Zhang

We can see that in our group, our reviewer is going to be Alejandro, whereas the second reviewer is going to be Angran Li, who is the farthest person from him in terms of preferences: