Homework 3

Problem 1:

a) Given these ROC test results, what is your estimate of the total readmissions and CareTracker costs for AMI patients for the past year if Tahoe had used the Xaltra system? Explain your estimate.

Ans) CareTracker, the new program the clinical staff piloted with AMI patients proved effective at reducing readmissions through a combination of patient education and post-discharge monitoring. Based on the data given to us, if someone would have been re-admitted and we give them CareTracker, the probability of them being re-admitted drops to 0.6, making the expected penalty: $8,000 × 0.6 = $4,800. But, we also need to pay $1,200 to give them CareTracker, which results in a total cost: of $4,800 + $1,200 = $6,000**.**

Using this data, we can create a Cost Matrix. Now, we can load the Tahoe data on past patients that did not receive the CareTracker intervention into Python as a Pandas data frame.

The column: *readmit30* is equal to 1 if the patient was re-admitted within 30 days of discharge, and 0 otherwise.

```
In [1]: import pandas as pd
        import numpy as np

In [54]: penalty_per_readmit = 8000
         caretracker_cost = 1200
         readmit_reduction = 0.6

In [55]: cost_matrix = pd.DataFrame([[0,1200],[8000,6000]])

In [56]: formatted_currency = lambda x : '${:,.2f}'.format(x)

In [57]: df = pd.read_excel('Tahoe_Data.xlsx')
         df
```

Out[57]:

	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
...	...	...	...	...	...	...	...
4377	88	1	0	1	34	94	0
4378	98	0	0	1	51	136	0
4379	84	1	0	1	10	32	0
4380	67	1	1	1	10	73	0
4381	79	1	0	1	20	92	0

4382 rows × 7 columns

We can analyze whether or not Tahoe should implement the Xaltra system for predicting readmission. The main factor that we will consider is the amount of money the system will save Tahoe. The best case would be if we can perfectly predict and apply CareTracker only to patients we knew would be readmitted, which comes up to $5,988,000, leading to a potential saving of $1,996,000.

```
In [58]: best_case = pd.DataFrame([[(1-df.readmit30).sum(), 0], [0, df.readmit30.sum()]])
         formatted_currency((cost_matrix*best_case).sum().sum())

Out[58]: '$5,988,000.00'
```

We need to calculate the best cost, we write a function that calculates cost given a true positive rate, a false positive rate, and a cost matrix. We first build a 'confusion matrix. It summarizes four kinds of outcomes from a classification based on the true positive and false positive rates, as well as the number of actual positives (true_positive) and actual negatives (true_negative). We create a function that returns the 'best_cost' by multiplying and adding the matrices element-wise. Next, we can calculate costs for each threshold based on the true positive and false positive rates.

```
In [59]: true_positive= df['readmit30'].sum()
         true_negative=len(df)-true_positive

         def best_cost(TPR, FPR, costs):
             confusion_matrix = pd.DataFrame([[(1-FPR)*(true_negative), FPR*(true_negative)],[(1-TPR)*true_positive, TPR*true_pc
             return (confusion_matrix*costs).sum().sum()
```

```
In [67]: df= pd.read_excel('homework3.xlsx', sheet_name = 'Pb1_ROC')

         df= df[['False Positive Rate','Logistic True Positive Rate',
                           'Xaltra True Positive Rate']]

         df

Out[67]:
```

	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
...	...	...	...
95	0.000	0.000	0.00000
96	0.000	0.000	0.00000
97	0.000	0.000	0.00000
98	0.000	0.000	0.00000
99	0.000	0.000	0.00000

100 rows × 3 columns

```
In [68]: df['Logistic Cost'] = best_cost( df['Logistic True Positive Rate'],
                                          df['False Positive Rate'], cost_matrix )
         df['Xaltra Cost'] = best_cost( df['Xaltra True Positive Rate'],
                                        df['False Positive Rate'], cost_matrix )
         df
```

Out[68]:

	False Positive Rate	Logistic True Positive Rate	Xaltra True Positive Rate	Logistic Cost	Xaltra Cost
0	1.000	1.000	1.00000	10048800.0	10048800.0
1	1.000	1.000	1.00000	10048800.0	10048800.0
2	0.997	1.000	1.00000	10036617.6	10036617.6
3	0.975	1.000	1.00000	9947280.0	9947280.0
4	0.934	0.997	0.99805	9786775.2	9784679.4
...	...	...	...	...	...
95	0.000	0.000	0.00000	7984000.0	7984000.0
96	0.000	0.000	0.00000	7984000.0	7984000.0
97	0.000	0.000	0.00000	7984000.0	7984000.0
98	0.000	0.000	0.00000	7984000.0	7984000.0
99	0.000	0.000	0.00000	7984000.0	7984000.0

100 rows × 5 columns

```
In [87]: logisict_cost_minimum=min(df['Logistic Cost'])
         xaltra_cost_minimum=min(df['Xaltra Cost'])
         print('Minimum Xaltra Cost: '+formatted_currency(xaltra_cost_minimum))
         print('Total Logistic Cost: ' + formatted_currency(logisict_cost_minimum))
         print('Total Xaltra Cost: ' + formatted_currency(xaltra_cost_minimum))
         print('Potential Cost Savings: ' + formatted_currency(-(xaltra_cost_minimum - logisict_cost_minimum)))

Minimum Xaltra Cost: $7,129,282.40
Total Logistic Cost: $7,489,608.80
Total Xaltra Cost: $7,129,282.40
Potential Cost Savings: $360,326.40
```

To determine the cost of readmissions to Tahoe if they had used Xaltra we find the minimum of the Xaltra cost, which comes up to **$7,129,282.40.**

b) What would have been the reduction in cost relative to Tahoe's current system, over the last year? Do the savings justify the fees Xaltra is charging? Why or why not.

Ans) To determine the cost of readmissions to Tahoe if they had used Xaltra we can use the best_cost function we created to find the best_cost for Xaltra and the Logistic Cost, and then find the minimum cost for both. Then, we can find the difference between the two to get the reduction in cost relative to the current system, which is given as the Potential Cost Savings.

In [68]:
```
df['Logistic Cost'] = best_cost( df['Logistic True Positive Rate'],
                                 df['False Positive Rate'], cost_matrix )
df['Xaltra Cost'] = best_cost( df['Xaltra True Positive Rate'],
                               df['False Positive Rate'], cost_matrix )
df
```

Out[68]:

	False Positive Rate	Logistic True Positive Rate	Xaltra True Positive Rate	Logistic Cost	Xaltra Cost
0	1.000	1.000	1.00000	10048800.0	10048800.0
1	1.000	1.000	1.00000	10048800.0	10048800.0
2	0.997	1.000	1.00000	10036617.6	10036617.6
3	0.975	1.000	1.00000	9947280.0	9947280.0
4	0.934	0.997	0.99805	9786775.2	9784679.4
...	...	...	...	...	...
95	0.000	0.000	0.00000	7984000.0	7984000.0
96	0.000	0.000	0.00000	7984000.0	7984000.0
97	0.000	0.000	0.00000	7984000.0	7984000.0
98	0.000	0.000	0.00000	7984000.0	7984000.0
99	0.000	0.000	0.00000	7984000.0	7984000.0

100 rows × 5 columns

In [65]:
```
logisict_cost_minimum=min(df['Logistic Cost'])
xaltra_cost_minimum=min(df['Xaltra Cost'])
print('Total Logistic Cost: ' + formatted_currency(logisict_cost_minimum))
print('Total Xaltra Cost: ' + formatted_currency(xaltra_cost_minimum))
print('Potential Cost Savings: ' + formatted_currency(-(xaltra_cost_minimum - logisict_cost_minimum)))
```

```
Total Logistic Cost: $7,489,608.80
Total Xaltra Cost: $7,129,282.40
Potential Cost Savings: $360,326.40
```

Thus, we can expect a potential cost savings of **$360,326.40**.
We can analyze the savings from the entire lifetime of the system if we compare it with the yearly Xaltra Cost for the last year or ten years with the potential cost savings(and multiply by the number of years, i.e. 10). We notice that the potential cost savings have **increased** as compared to before, I think the savings **justify** the fees Xaltra is charging.

Problem 2:

a) Food rating data from your cluster is available in the homework3.xlsx file. Using this data, find the person in your learning team whose food preference most closely matches that of the overall cluster. Provide the name of your learning team reviewer and explain how you selected them. (Only consider students who are present in the data.) Hint: Compute the “food rating distance” of the candidate reviewer to every other student in your cluster by accounting for the food types they jointly rated. The learning team reviewer you will select will be the one with the smallest average food rating distance to students in your cluster.
Ans)

```
In [69]: df_ratings=pd.read_excel('homework3.xlsx', sheet_name = 'Pb2_Cluster_X')
         df_ratings
```

Out[69]:

	name	Mexican	Chinese	Greek	Indian	Thai	Italian	Ethiopian	French	Sushi	Steakhouse	Vegan	Spanish	Caribbean	Seafood	Bar food
0	Ruishi Tao	3.0	5.0	3.0	4.0	3.0	1.0	1.0	2.0	5.0	5.0	5.0	4.0	5.0	4.0	4.0
1	Blanche Loviton	4.0	5.0	5.0	5.0	5.0	5.0	4.0	5.0	5.0	3.0	2.0	3.0	4.0	5.0	3.0
2	Jules Deschamps	4.0	4.0	5.0	4.0	4.0	5.0	3.0	4.0	4.0	4.0	3.0	3.0	3.0	4.0	4.0
3	Carlie Iskandar	3.0	1.0	5.0	4.0	2.0	5.0	1.0	5.0	5.0	5.0	1.0	3.0	1.0	4.0	NaN
4	Victor Perroux	4.0	5.0	3.0	3.0	4.0	5.0	3.0	5.0	5.0	NaN	4.0	3.0	3.0	3.0	2.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
239	Yisen Wang	4.0	5.0	3.0	3.0	3.0	4.0	NaN	4.0	4.0	4.0	2.0	4.0	3.0	4.0	4.0
240	Qianqian Ye	3.0	5.0	3.0	2.0	4.0	3.0	4.0	3.0	3.0	4.0	3.0	3.0	NaN	3.0	3.0
241	Xuanru Deng	2.0	5.0	3.0	2.0	4.0	3.0	NaN	3.0	4.0	3.0	NaN	4.0	NaN	3.0	3.0
242	Sinjini Shah	5.0	5.0	3.0	5.0	5.0	5.0	3.0	3.0	NaN	NaN	5.0	3.0	NaN	NaN	4.0
243	Chenkai Li	3.0	5.0	4.0	4.0	4.0	4.0	3.0	5.0	5.0	5.0	3.0	4.0	NaN	4.0	4.0

244 rows × 16 columns

We begin by importing the ratings for the various cuisines as a Pandas data frame into Python. We then create a list of team members that we will be using to compare with the other reviewers.

```
In [70]: team_members = ['Vriddhi Misra', 'Srishti Priya', 'Radha Marathe', 'Archit Aggarwal', 'Parth Batra']
```

```
In [84]: def distances(df, target):
             interuser_distances= []
             for user in df.index:
                 u = df.loc[user,:]
                 t = target.values.tolist()[0]
                 distances = [(i-j)**2 for i, j in zip(u,t)] #calculate the distance between chosen and target user
                 cuisines = sum([pd.notnull(i) for i in distances]) #sum of distances
                 sum_of_distances = sum([i for i in distances if pd.notnull(i)])
                 if cuisines == 0:
                     user_dists.append(float('inf'))
                 else:
                     interuser_distances.append(np.sqrt(sum_of_distances/cuisines))
             return interuser_distances
```

We write a function to return the distances between each member of the team and each other reviewer from the dataset. This will help us compare and find the closest and farthest neighbors for each team member. We begin by parsing through each member's ratings and based on the sum of squared distances, if there are no overlapping ratings, we return the square root of the standardized distance.

```
In [85]: df_distances = df_lt = df_ratings[['name']].copy()
         for mem in team_members:
             df_distances[mem] = distances(df_ratings.loc[:, df_ratings.columns != 'name'],
                                           df_ratings.loc[df_ratings['name'] == mem, df_ratings.columns != 'name'])
         df_distances
```

Out[85]:

	name	Vriddhi Misra	Srishti Priya	Radha Marathe	Archit Aggarwal	Parth Batra
0	Ruishi Tao	1.653280	2.236068	2.033060	2.345208	1.511858
1	Blanche Loviton	1.591645	1.927248	1.807392	2.390457	1.069045
2	Jules Deschamps	1.341641	1.812654	1.366260	1.908627	0.845154
3	Carlie Iskandar	2.138090	2.415229	2.052873	2.056883	1.664101
4	Victor Perroux	1.581139	1.851640	1.603567	2.417882	0.919866
...	...	...	...	...	...	...
239	Yisen Wang	1.164965	1.812654	1.414214	1.980676	0.845154
240	Qianqian Ye	1.195229	1.647509	1.439246	1.961161	1.109400
241	Xuanru Deng	1.354006	1.889822	1.632993	2.195036	1.154701
242	Sinjini Shah	1.445998	1.732051	1.279204	2.408319	1.048809
243	Chenkai Li	1.535299	1.963961	1.732051	2.385856	0.960769

244 rows × 6 columns

```
In [97]: df_team = pd.DataFrame(team_members, columns=['name'])
         df_team ['Average']=np.mean(df_distances)[df_team['name']].values
         df_team ['Maximum']=np.max(df_distances)[df_team['name']].values
         df_team ['Least Similar Taste']=[df_distances[df_distances[name] == dist]['name'].values[0] for name, dist in
                                          zip(df_team['name'], df_team['Maximum'])]
         df_team
```

Out[97]:

	name	Average	Maximum	Least Similar Taste
0	Vriddhi Misra	1.591019	2.335497	Alex Lan
1	Srishti Priya	1.866117	2.878492	Natasha Ratanapan
2	Radha Marathe	1.735810	2.42384	Yixuan Zhu
3	Archit Aggarwal	2.227396	4.0	Yiwen Zhang
4	Parth Batra	1.241788	2.088932	Alex Lan

We make a new data frame with rows and columns as the names of the reviewers, and the data as the distances between them. We select our reviewers based on the least and most distance from the rest of the dataset. First, the reviewer in our group will be the one with the lowest average distance from the rest of the class (comparatively most similar to the others in the dataset on average). Thus, the first reviewer will be *Parth Batra*.

b) Next, we want to find a partner for the reviewer you selected in part (a). For the person selected to be your learning team's reviewer, find a fellow student in your cluster whose taste in food is least similar to your reviewer. Provide their name and explain how you selected them.
Ans) Partner to reviewer one will be *Alex Lan* as he has the Maximum distance from Parth Batra, which means that his taste is least similar to our chosen reviewer as compared to the others.