

Week 6 – HomeWork 3

Paritosh Rai



7335 Machine Learning 2



HW 3: Decision Making with Matrices

*This is a pretty simple assignment.  You will do something you do every day, but today it will be with matrix manipulations.*

*The problem is: you and your work friends are trying to decide where to go for lunch. You have to pick a restaurant that’s best for everyone.  Then you should decide if you should split into two groups, so everyone is happier.*

*Despite the simplicity of the process, you will need to make decisions regarding how to process the data.*

*This process was thoroughly investigated in the operation research community.  This approach can prove helpful on any number of decision-making problems that are currently not leveraging machine learning.*

**# You asked your 10 work friends to answer a survey. They gave you back the following dictionary object.**

**people = {'Jane': {'willingness to travel':**

**'desire for new experience':**

**'cost':**

**'indian food':**

**'mexican food':**

**'hipster points':**

**'vegetarian': }**

**}**

Created a dictionary of friends:

>>> print(friends)

{'John': {'willingness to travel': 3, 'desire for new experience': 3, 'cost': 2, 'vegetarian': 4}, 'Ani': {'willingness to travel': 2, 'desire for new experience': 4, 'cost': 3, 'vegetarian': 3}, 'Ben': {'willingness to travel': 3, 'desire for new experience': 1, 'cost': 1, 'vegetarian': 1}, 'Mary': {'willingness to travel': 5, 'desire for new experience': 4, 'cost': 5, 'vegetarian': 4}, 'Don': {'willingness to travel': 2, 'desire for new experience': 2, 'cost': 2, 'vegetarian': 5}, 'Ron': {'willingness to travel': 1, 'desire for new experience': 5, 'cost': 2, 'vegetarian': 4}, 'Sam': {'willingness to travel': 5, 'desire for new experience': 5, 'cost': 5, 'vegetarian': 4}, 'Kelly': {'willingness to travel': 2,

'desire for new experience': 2, 'cost': 3, 'vegetarian': 2}, 'Tim': {'willingness to travel': 1, 'desire for new experience': 1, 'cost': 1, 'vegetarian': 3}, 'Ken': {'willingness to travel': 4, 'desire for new experience': 4, 'cost': 3, 'vegetarian': 1}}

*# all values are between 1 to 5*

*# Willingness to travel (1 to 5): 1 is not willing to travel and 5 is highly willing to travel.*

*# Desire for new experience (1 to 5): 1 is no desire of new experience and 5 is high desire for new experience.*

*# Cost (1 to 5): 1 is high cost and 5 is low cost*

*#indian food (1 to 5): 1 low desire for Indian food and 5 is high desire for Indian food*

*# Mexican food (1 to 5): 1 low desire for Mexican food and 5 is high desire for Mexican food*

*#hipster points (1 to 5): 1 is low hipster points and 5 is hipster points*

*# vegetarian (1 to 5): 1 is for vegetarian not preferred and 5 is vegetarian preferred*

**# Transform the user data into a matrix(M\_people). Keep track of column and row ids.**

Created a 4X10 matrix with name friendsMatrix with following four fields:

* willingness to travel
* desire for new experience
* cost
* vegetarian

>>> print(friendsMatrix)

[[3 3 2 4]

[2 4 3 3]

[3 1 1 1]

[5 4 5 4]

[2 2 2 5]

[1 5 2 4]

[5 5 5 4]

[2 2 3 2]

[1 1 1 3]

[4 4 3 1]]

**# Next you collected data from an internet website. You got the following information.**

**restaurants  = {'flacos':{'distance' :**

**'novelty' :**

**'cost':**

**'average rating':**

**'cuisine':**

**'vegetarians':}}**

Created restaurant dictionary:

{'Rio': {'distance': 1, 'novelty': 1, 'cost': 1, 'vegetarians': 5}, 'Ind': {'distance': 3, 'novelty': 2, 'cost': 4, 'vegetarians': 1}, 'Bang': {stance': 3, 'novelty': 1, 'cost': 1, 'vegetarians': 4}, 'Desi': {'distance': 2, 'novelty': 1, 'cost': 3, 'vegetarians': 5}, 'Del': {'distance': 5, 'novelty': 2, 'cost': 3, 'vegetarians': 5}, 'Bell': {'distance': 4, 'novelty': 4, 'cost': 4, 'vegetarians': 4}, 'veg': {'distance': 5, 'novelty': 3, 'cost': 4, 'vegetarians': 5}, 'Motu': {'distance': 4, 'novelty': 4, 'cost': 4, 'vegetarians': 4}}

*For our analysis we will choses bigger the number better it is.*

*all valises are between 1 to 5*

*distance (1 to 5): 1 means longer distance and 5 means smaller distance*

*novelty (1 to 5): 1 is low novelty and 5 is high novelty*

*cost (1 to 5):1 is high cost and 5 is low cost*

*average rating (1 to 5): 1 is low rating and 5 is high rating*

*cuisine (1 to 2):1 is Indian and 2 is Mexican*

*vegetarians (1 to 5): 1 is for low in vegetarian and 5 is high on vegetarian*

# Transform the restaurant data into a matrix(M\_resturants) use the same column index.

Created a 4X10 matrix with name restaurantsMatrix with following four fields:

* distance
* novelty
* cost
* vegetarians

>>> print(restaurantsMatrix)

[[1 1 1 5]

[2 2 2 4]

[3 2 4 1]

[4 4 5 2]

[3 1 1 4]

[2 1 3 5]

[5 2 3 5]

[4 4 4 4]

[5 3 4 5]

[4 4 4 4]]

>>>

**# The most important idea in this project is the idea of a linear combination.**

**# Informally describe what a linear combination is and how it will relate to our restaurant matrix.**

Linear combination is constructed from a set of terms by multiplying each term by a constant and/or adding. Concept of linear combination is around linear algebra.

In term of relating to restaurant we can add or multiply constant to numbers in matrix or add and multiply the restaurant matrix with another matrix like friends matrix created above. It can add or multiply with a constant number.

**# Choose a person and compute (using a linear combination) the top restaurant for them.  What does each entry in the resulting vector represent?**

Created a function to pull any column (or friend) from the matrix.

Swap the matrix (transpose the matrix)

Pulling 2nd person in the matrix:

Score given by the 2nd person in the list is: [2, 4, 3, 3]

Score generated by 2nd person for respective restaurant are: [24 30 29 45 25 32 42 48 49 48]

Dictionary of restaurant and score by a friend :

{'Rio': 24, 'Ind': 30, 'Tin\_Drum': 29, 'Bang': 45, 'Ranch': 25, 'Desi': 32, 'Del': 42, 'Bell': 48, 'veg': 49, 'Motu': 48}

Best restaurant selected by person is: veg

Each value represents the score generated based on value assigned by individual to attributes and rating of restaurant.

**# Next, compute a new matrix (M\_usr\_x\_rest  i.e. an user by restaurant) from all people.  What does the a\_ij matrix represent?**

Matrix score of every restaurant based on friends preference is:

[[28 32 27 42 30 35 47 48 52 48]

[24 30 29 45 25 32 42 48 49 48]

[10 14 16 23 15 15 25 24 27 24]

[34 44 47 69 40 49 68 72 77 72]

[31 32 23 36 30 37 45 44 49 44]

[28 32 25 42 26 33 41 48 48 48]

[35 46 49 73 41 50 70 76 80 76]

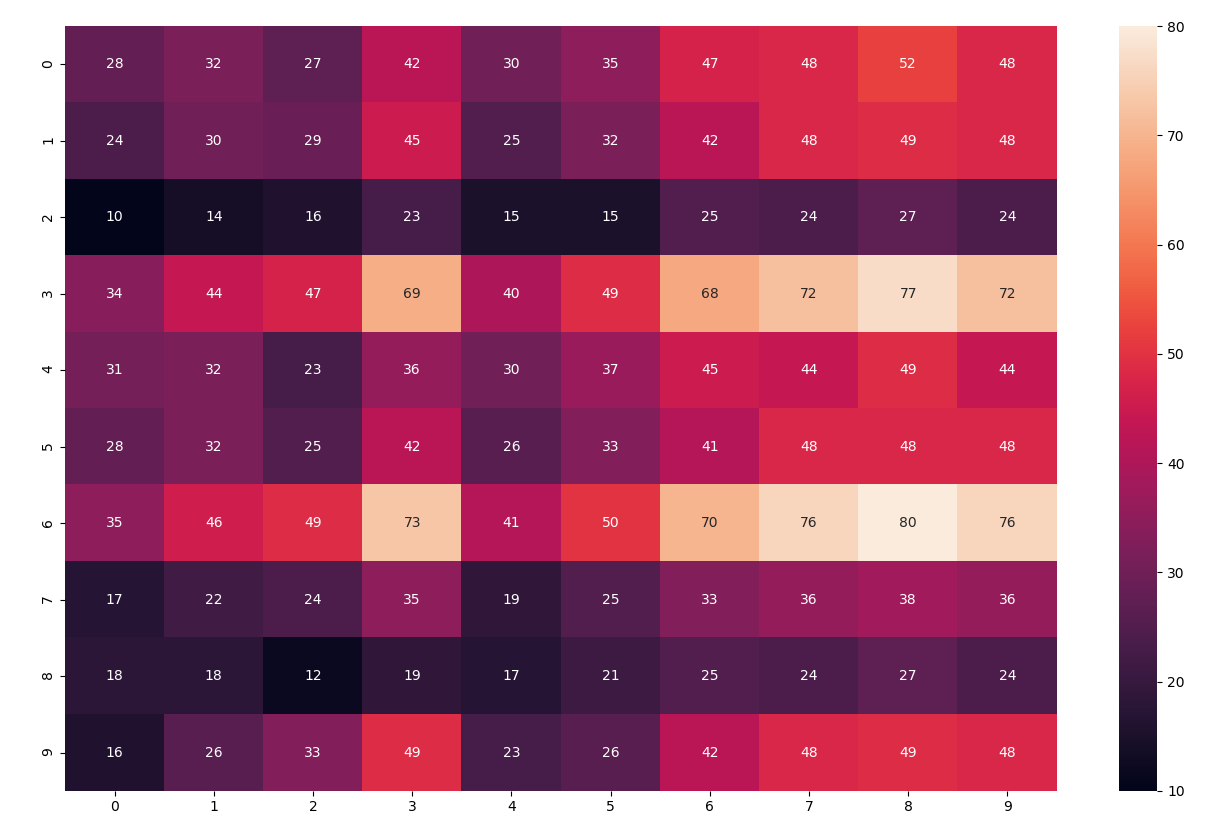
[17 22 24 35 19 25 33 36 38 36]

[18 18 12 19 17 21 25 24 27 24]

[16 26 33 49 23 26 42 48 49 48]]

Matrix generated above give us score for each restaurant based on preferences of friend and restaurant feature

Fig: Plot raw data in heatmap



**# Sum all columns in M\_usr\_x\_rest to get the optimal restaurant for all users.  What do the entries represent?**

Score of each restaurant:

{'Rio': 389, 'Ind': 372, 'Tin\_Drum': 193, 'Bang': 572, 'Ranch': 371, 'Desi': 371, 'Del': 596, 'Bell': 285, 'veg': 205, 'Motu': 360}

# Best restaurent selected is:

...

('Del', 596)

('Bang', 572)

('Rio', 389)

('Ind', 372)

('Ranch', 371)

('Desi', 371)

('Motu', 360)

('Bell', 285)

('veg', 205)

('Tin\_Drum', 193)

>>> best\_res = max(res\_score,key=res\_score.get)

>>> print('Best restaurent selected by person is:',best\_res)

Best restaurent selected by person is: Del

**# Now convert each row in the M\_usr\_x\_rest into a ranking for each user and call it M\_usr\_x\_rest\_rank.   Do the same as above to generate the optimal restaurant choice.**

>>> M\_usr\_x\_rest

array([[28, 32, 27, 42, 30, 35, 47, 48, 52, 48],

[24, 30, 29, 45, 25, 32, 42, 48, 49, 48],

[10, 14, 16, 23, 15, 15, 25, 24, 27, 24],

[34, 44, 47, 69, 40, 49, 68, 72, 77, 72],

[31, 32, 23, 36, 30, 37, 45, 44, 49, 44],

[28, 32, 25, 42, 26, 33, 41, 48, 48, 48],

[35, 46, 49, 73, 41, 50, 70, 76, 80, 76],

[17, 22, 24, 35, 19, 25, 33, 36, 38, 36],

[18, 18, 12, 19, 17, 21, 25, 24, 27, 24],

[16, 26, 33, 49, 23, 26, 42, 48, 49, 48]])

Convert to Rank

>>> M\_usr\_x\_rest\_rank

array([[ 1, 5, 2, 6, 3, 7, 8, 10, 4, 9],

[ 3, 5, 1, 2, 4, 6, 8, 10, 7, 9],

[ 1, 5, 2, 3, 6, 7, 4, 8, 10, 9],

[ 1, 5, 2, 3, 6, 7, 4, 8, 10, 9],

[ 3, 5, 1, 2, 6, 7, 4, 8, 9, 10],

[ 3, 5, 1, 2, 4, 6, 8, 10, 7, 9],

[ 1, 5, 2, 3, 6, 7, 4, 8, 10, 9],

[ 1, 2, 5, 6, 3, 4, 8, 10, 7, 9],

[ 1, 5, 3, 2, 6, 7, 4, 8, 10, 9],

[ 3, 1, 5, 2, 6, 4, 7, 8, 10, 9]], dtype=int64)

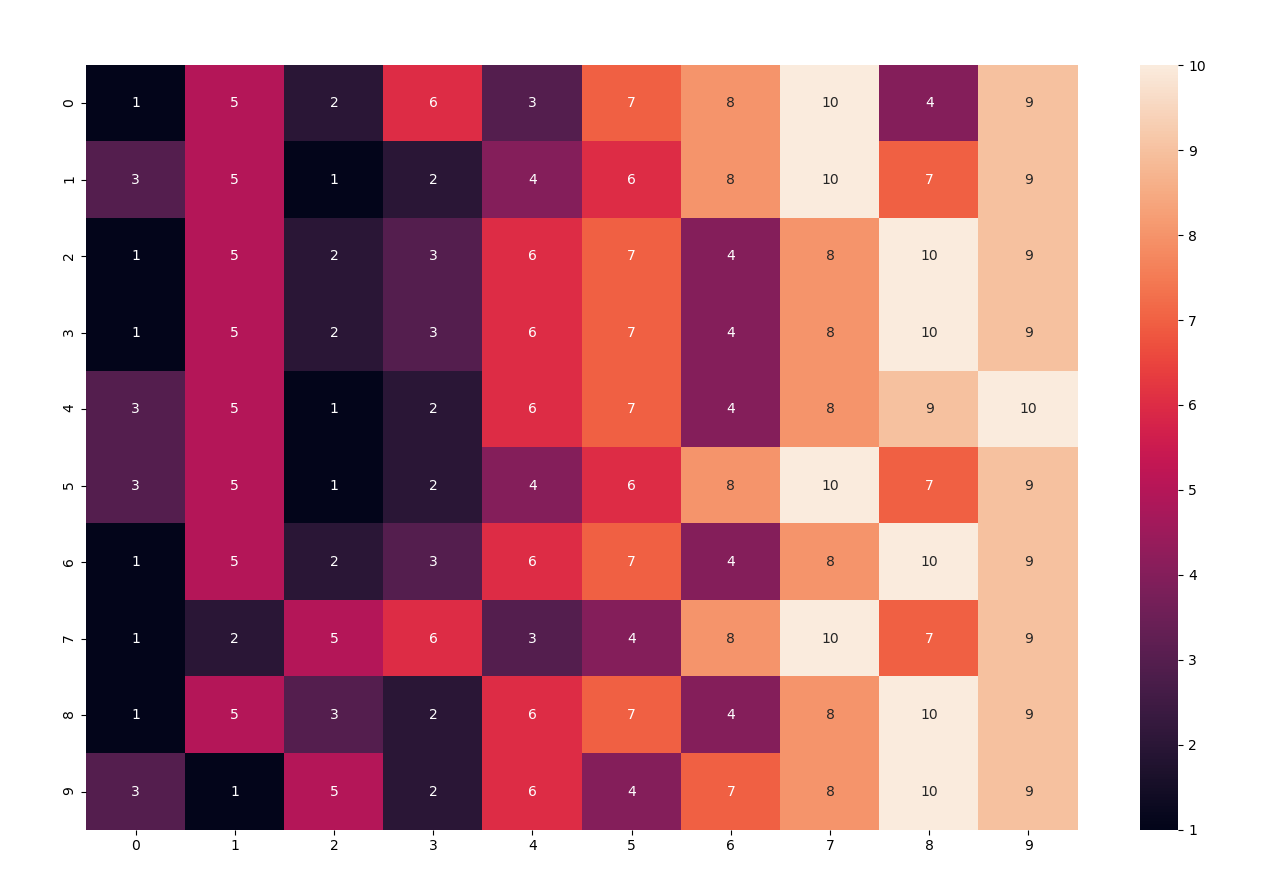
Rank Score of each restaurant :

{'Rio': 55, 'Ind': 55, 'Tin\_Drum': 55, 'Bang': 55, 'Ranch': 55, 'Desi': 55, 'Del': 55, 'Bell': 55, 'veg': 55, 'Motu': 55}>>>

>>> print('Best restaurant selected by person is:',best\_res\_r)

Best restaurant selected by person is: Rio

>>>Rank of restaurant



**# Why is there a difference between the two?  What problem arrives?  What does it represent in the real world?**

Ranking brings in the position of data and gap between them are constant. This breaks the relationship between numbers. However, when we deal with raw numbers the gaps are not constant and there is relationship among the data. Extreme and raw values skew the results. This will have a very different results compare to rank where they differences are constant. These can create very different results. Ranking get the outcome in relative scale losing inter relationship whereas raw data keep the inter dependency and variance into account. Also, the ranking is ‘purer’ than the top-down bottom-up judgments because they take no account of external statistical information, or of the mark totals on the scripts. They are based purely on a holistic judgment of relative quality of performance. Ranking are more quality measure say one is better than other. In case of raw data quantity and quality both plays the role.

In real world we just go by rating of restaurant which is kind of rank and do not look at individual preferences. When we look at rating of restaurant its linear combination of multiple factors and input from multiple users. So we have to be carful when we decide the restaurant based on just ranking.

**# How should you preprocess your data to remove this problem.**

We will scale the data and relook at outcome.

Following is the scaled Matrix:

>>> print(restaurantsMatrix\_scaled)

[[0. 0. 0. 1. ]

[0.25 0.33333333 0.25 0.75 ]

[0.5 0.33333333 0.75 0. ]

[0.75 1. 1. 0.25 ]

[0.5 0. 0. 0.75 ]

[0.25 0. 0.5 1. ]

[1. 0.33333333 0.5 1. ]

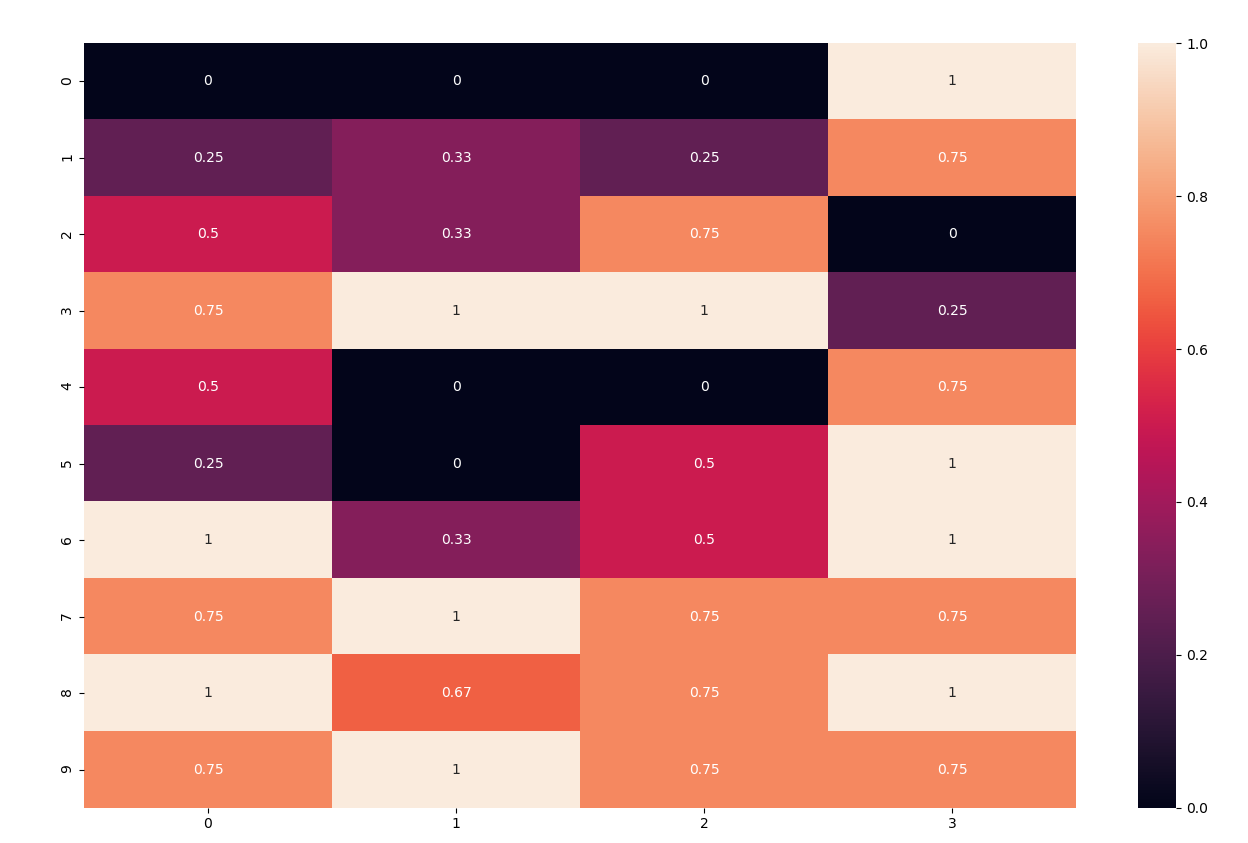
[0.75 1. 0.75 0.75 ]

[1. 0.66666667 0.75 1. ]

[0.75 1. 0.75 0.75 ]]

>>>

Heatmap of scaled restaurant Matrix



friends scaled Matrix

>>> print(friendsMatrix\_scaled)

[[0.5 0.5 0.25 0.75]

[0.25 0.75 0.5 0.5 ]

[0.5 0. 0. 0. ]

[1. 0.75 1. 0.75]

[0.25 0.25 0.25 1. ]

[0. 1. 0.25 0.75]

[1. 1. 1. 0.75]

[0.25 0.25 0.5 0.25]

[0. 0. 0. 0.5 ]

[0.75 0.75 0.5 0. ]]

>>>

Score of scaled friends and restaurant

>>> print(M\_usr\_scale\_x\_rest\_scale)

[[0.75 0.91666667 0.60416667 1.3125 0.8125 1. 1.54166667 1.625 1.77083333 1.625 ]

[0.5 0.8125 0.75 1.5625 0.5 0.8125 1.25 1.6875 1.625 1.6875 ]

[0. 0.125 0.25 0.375 0.25 0.125 0.5 0.375 0.5 0.375 ]

[0.75 1.3125 1.5 2.6875 1.0625 1.5 2.5 2.8125 3. 2.8125 ]

[1. 0.95833333 0.39583333 0.9375 0.875 1.1875 1.45833333 1.375 1.60416667 1.375 ]

[0.75 0.95833333 0.52083333 1.4375 0.5625 0.875 1.20833333 1.75 1.60416667 1.75 ]

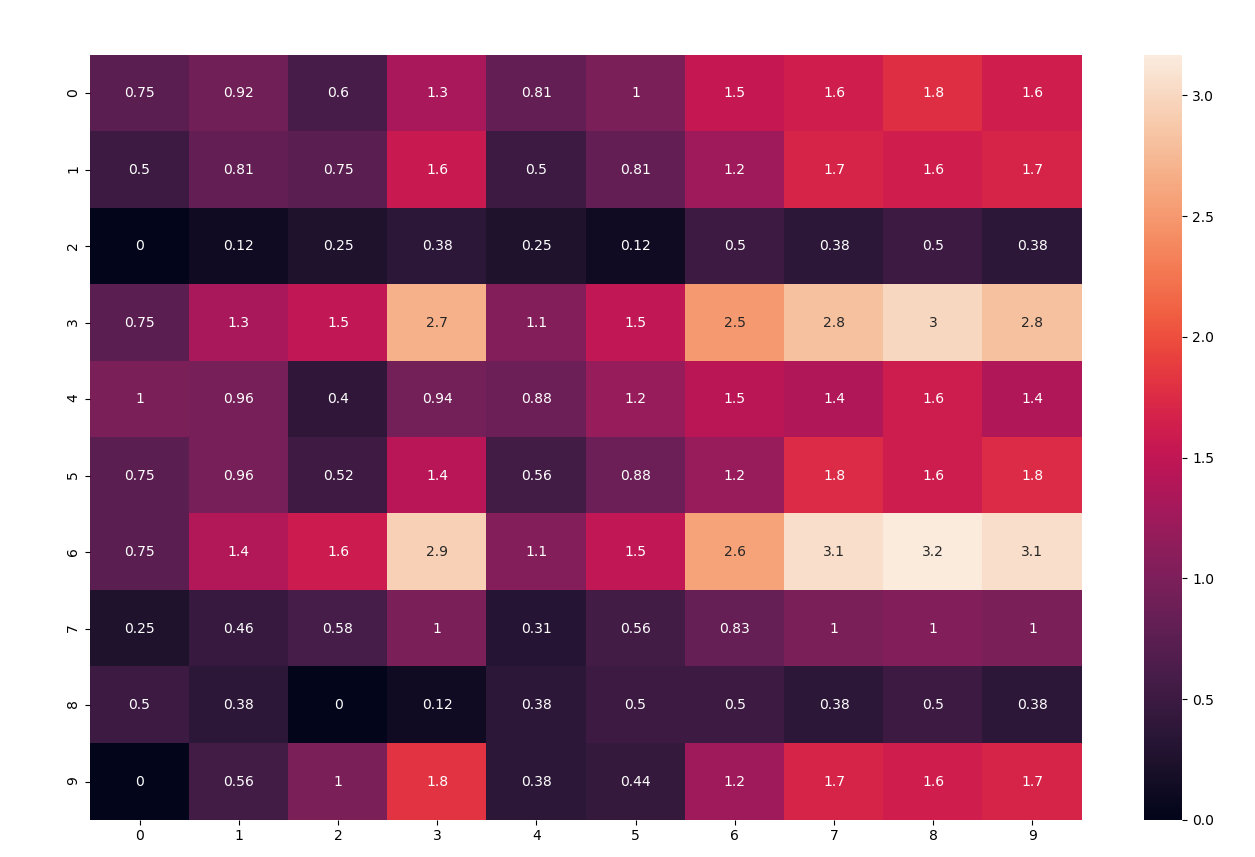
[0.75 1.39583333 1.58333333 2.9375 1.0625 1.5 2.58333333 3.0625 3.16666667 3.0625 ]

[0.25 0.45833333 0.58333333 1. 0.3125 0.5625 0.83333333 1. 1.04166667 1. ]

[0.5 0.375 0. 0.125 0.375 0.5 0.5 0.375 0.5 0.375 ]

[0. 0.5625 1. 1.8125 0.375 0.4375 1.25 1.6875 1.625 1.6875 ]]

Heatmap of score of restaurants with scaled friends and scaled restaurant



>>> best\_res\_s = max(res\_score\_s,key=res\_score\_s.get)

>>> print('Best restaurant selected by person is:',best\_res\_s)

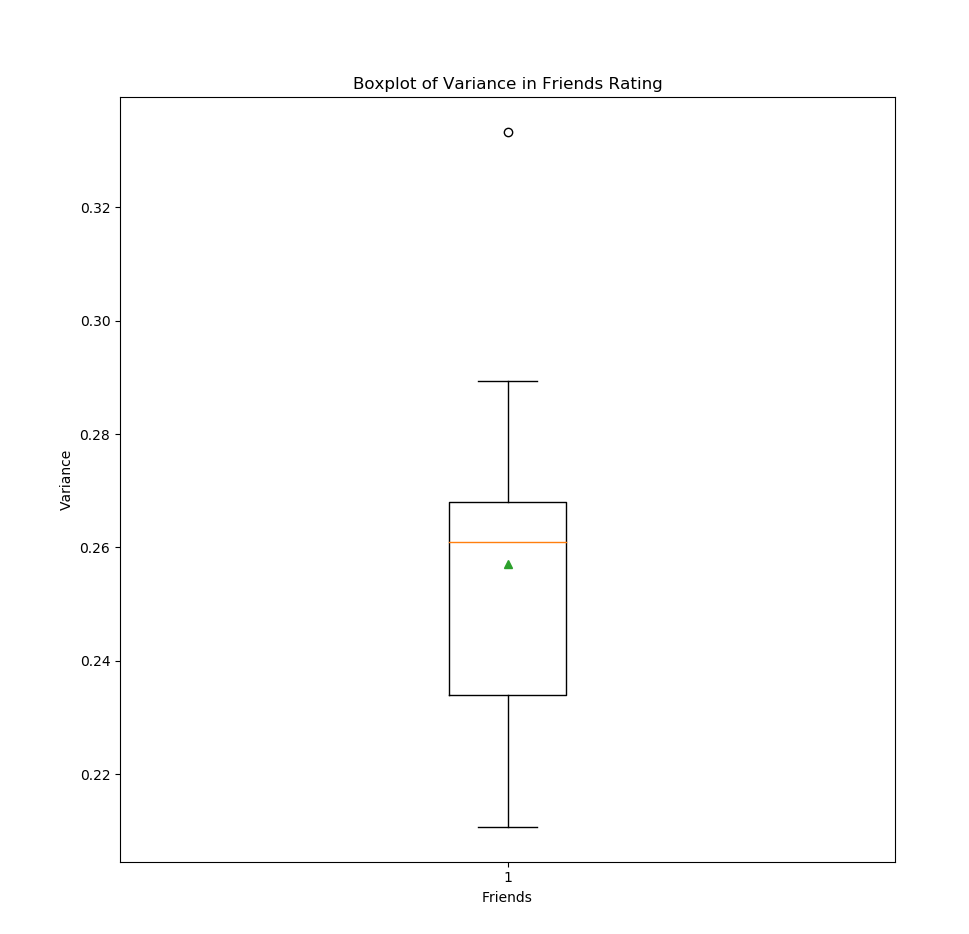
Best restaurant selected by person is: Del

>>>

After scaling friends and restaurant matrix we get same best matrix as from raw data.

**# Find  user profiles that are problematic, explain why?**

Create Box plot of variance to identify the problematic users



Following is variance

{'John': 0.23206121277295644, 'Ani': 0.25862394560872454, 'Ben': 0.2894606782064958, 'Mary': 0.26314528282834654, 'Don': 0.21101867952614375, 'Ron': 0.23971040085516493, 'Sam': 0.26962831445101565, 'Kelly': 0.2631578947368421, 'Tim': 0.21066168665062618, 'Ken': 0.3333333333333333}

Ken looks to be good in friends as it has variance, so he is differentiating good and bad restaurant whereas Don has lowest variance indicating he is not able helping create differentiation.

**# Think of two metrics to compute the dissatisfaction with the group.**

Method 1:

Del came up with best score from raw and scaled method. So, we assume friends will go Del. We can take friends del score as satisfaction rating and substrate from max score (i.e., 100)

**Del score of friends:**

{'John': 47, 'Ani': 42, 'Ben': 25, 'Mary': 68, 'Don': 45, 'Ron': 41, 'Sam': 70, 'Kelly': 33, 'Tim': 25, 'Ken': 42}

Calculate Dissatisfaction rating:

disat= del\_rating

disat = [100 - x for x in disat]

disat

frd\_disat = dict(zip(friendsName, disat))

frd\_disat

>>> frd\_disat

{'John': 53, 'Ani': 58, 'Ben': 75, 'Mary': 32, 'Don': 55, 'Ron': 59, 'Sam': 30, 'Kelly': 67, 'Tim': 75, 'Ken': 58}

>>>

Method 2:

The Dis-satisfaction score from method 1 and convert to rating 1 to 5.

disat2 = disat

disat2=[x\*5/100 for x in disat2]

disat2

frd\_disat2 = dict(zip(friendsName, disat2))

frd\_disat2

>>> frd\_disat2

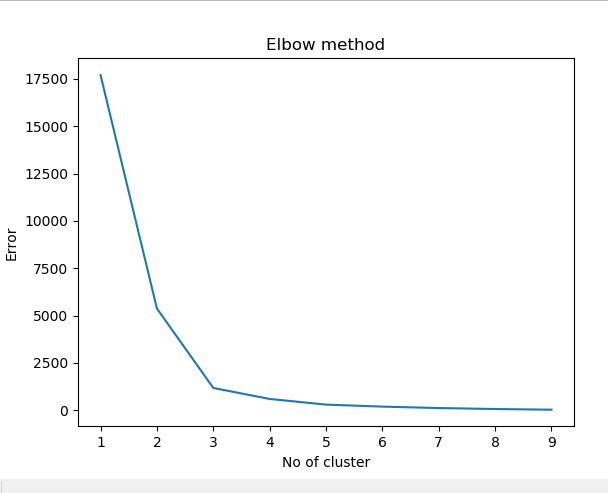
{'John': 2.65, 'Ani': 2.9, 'Ben': 3.75, 'Mary': 1.6, 'Don': 2.75, 'Ron': 2.95, 'Sam':

1.5, 'Kelly': 3.35, 'Tim': 3.75, 'Ken': 2.9}

>>>

**# Should you split in two groups today?**

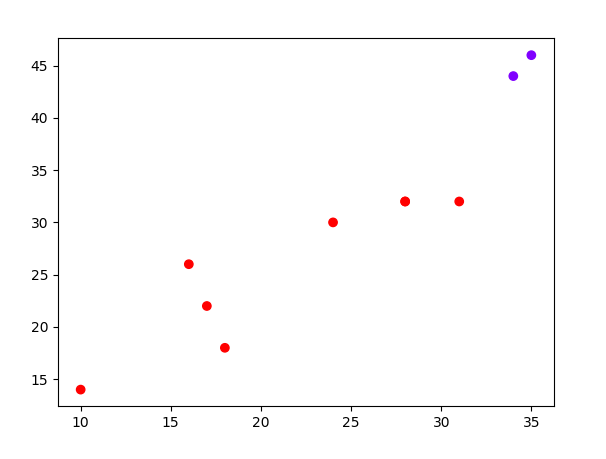
Determine number of clusters required using elbow method



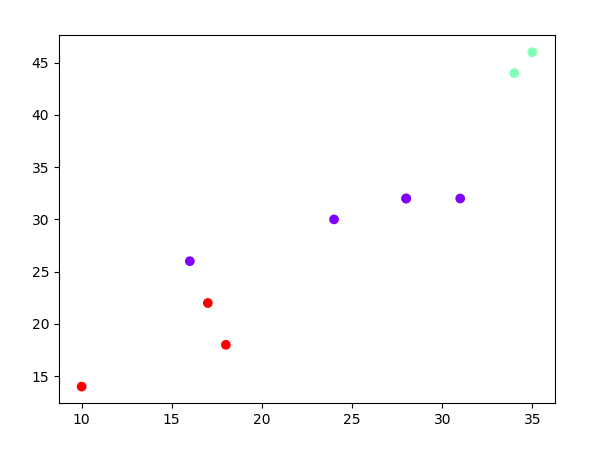
Based on elbow graph three (3) appears to make more sense however we will start with two groups to get initial view

# Ok. Now you just found out the boss is paying for the meal. How should you adjust? Now what is the best restaurant?

This how it looks with two group:



Going with Elbow recommendation and doing three cluster:



>> frd\_grp3

{'John': 0, 'Ani': 0, 'Ben': 2, 'Mary': 1, 'Don': 0, 'Ron': 0, 'Sam': 1, 'Kelly': 2, 'Tim': 2, 'Ken': 0}

Based on Elbow method and looking at two and three cluster graphs three groups make sense. So, two group will not make sense.

**Ok. Now you just found out the boss is paying for the meal. How should you adjust? Now what is the best restaurant?**

We will remove the price component from the restaurant and friends and calculate M\_usr\_x\_rest with cost/price.

Create a new friends matrix with cost value zero (0)

>>> print(mod\_friendsMatrix)

[[3 3 0 4]

[2 4 0 3]

[3 1 0 1]

[5 4 0 4]

[2 2 0 5]

[1 5 0 4]

[5 5 0 4]

[2 2 0 2]

[1 1 0 3]

[4 4 0 1]]

>>>

Similarly create a restaurant matrix

>>> print(mod\_restaurantsMatrix)

[[1 1 0 5]

[2 2 0 4]

[3 2 0 1]

[4 4 0 2]

[3 1 0 4]

[2 1 0 5]

[5 2 0 5]

[4 4 0 4]

[5 3 0 5]

[4 4 0 4]]

>>>

Score generated with make cost zero in friends and restaurant matrix

>>> print('Matrix score of every restaurant based on friends preference is:\n', mod\_M\_usr\_x\_rest)

Matrix score of every restaurant based on friends preference is:

[[26 28 19 32 28 29 41 40 44 40]

[21 24 17 30 22 23 33 36 37 36]

[ 9 12 12 18 14 12 22 20 23 20]

[29 34 27 44 35 34 53 52 57 52]

[29 28 15 26 28 31 39 36 41 36]

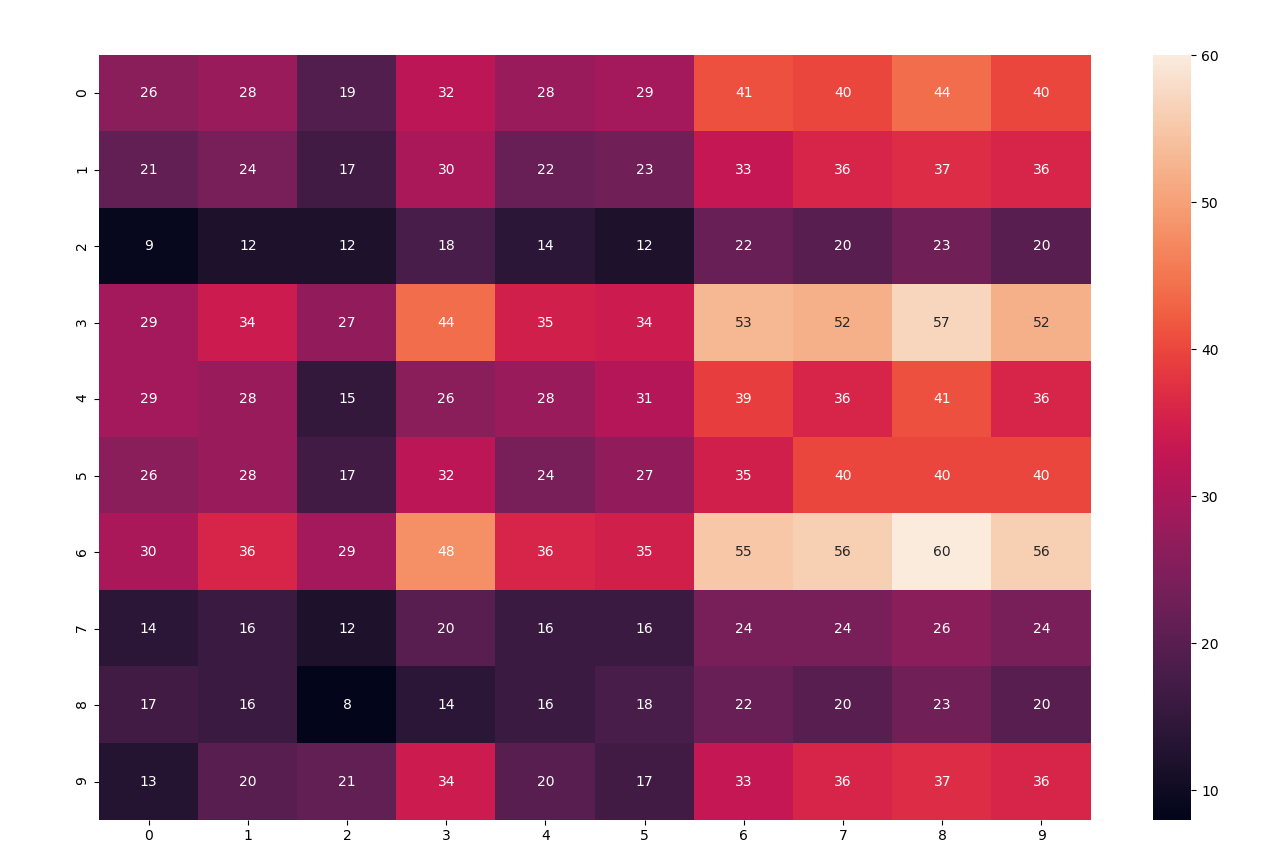
[26 28 17 32 24 27 35 40 40 40]

[30 36 29 48 36 35 55 56 60 56]

[14 16 12 20 16 16 24 24 26 24]

[17 16 8 14 16 18 22 20 23 20]

[13 20 21 34 20 17 33 36 37 36]]



New rating without accounting for cost is Del

>>> for k in sorted(mod\_res\_score, key=mod\_res\_score.get, reverse=True):

... k, mod\_res\_score[k]

... # Best restaurant selected is:

...

('Del', 441)

('Bang', 417)

('Rio', 327)

('Ranch', 309)

('Desi', 309)

('Ind', 279)

('Motu', 267)

('Bell', 192)

('veg', 174)

('Tin\_Drum', 162)

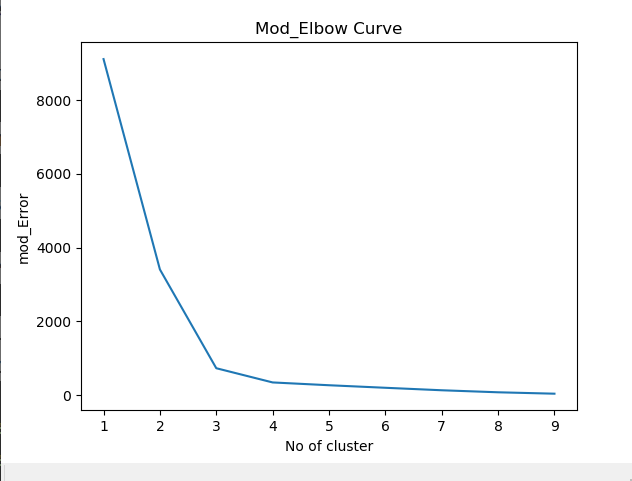
>>> mod\_best\_res = max(mod\_res\_score,key=mod\_res\_score.get)

>>> print('Best restaurant selected by person is:',mod\_best\_res)

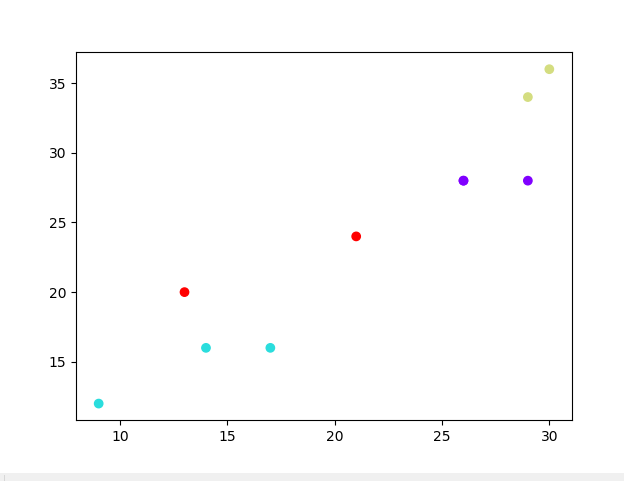
Best restaurant selected by person is: Del

Doing Kmean and seeing how they cluster

Elbow method show four clusters makes most sense



Cluster looks like



>>> mod\_frd\_grp4 = dict(zip(friendsName, mod\_labels))

>>> mod\_frd\_grp4

{'John': 0, 'Ani': 3, 'Ben': 1, 'Mary': 2, 'Don': 0, 'Ron': 0, 'Sam': 2, 'Kelly': 1, 'Tim': 1, 'Ken': 3}

>>>

**# Tomorrow you visit another team. You have the same restaurants and they told you their optimal ordering for restaurants.  Can you find their weight matrix?**

This may be challenging and will come up with limitation. As ordering show the sequency but ignores the relationship between the data points. As we have seen in above analysis ranking data give us different outcome than raw data or scaled data. As the scoring done by another team may not align with scoring done previous team of friends.