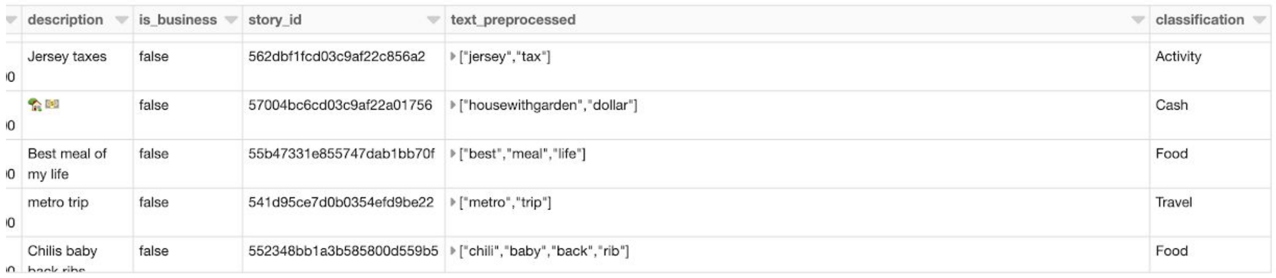
Team Five Guys
Submitted by: Ruby He, Vanessa Liang, Ruizhi Ma, Haonan Wang, Minke Wang

Q0 [5 pts]: Your first task is to open your Venmo app, find 10 words that are not already in the dictionary and add them to it. Make sure you don't add to the dictionary a duplicate word by hitting Control+F before adding your word.

The 10 new words we added are: tomato, popeye, carrot, hulu, hbo, mushroom, ups, fedex, wholefoods, italian.

Q1 [2 pts]: Use the text dictionary and the emoji dictionary to classify Venmo's transactions in your sample dataset.

We first utilized the emojis package to convert emojis to text. Then we preprocessed the transaction description data by first removing the punctuations and stopwords, then tokenizing and lemmatizing the text data. After preprocessing the text data, we used the text dictionary provided to classify the transactions.



description	is_business	story_id	text_preprocessed	classification
Jersey taxes	false	562dbf1fcd03c9af22c856a2	["jersey","tax"]	Activity
🏡💵	false	57004bc6cd03c9af22a01756	["housewithgarden","dollar"]	Cash
Best meal of my life	false	55b47331e855747dab1bb70f	["best","meal","life"]	Food
metro trip	false	541d95ce7d0b0354efd9be22	["metro","trip"]	Travel
Chilis baby	false	552348bb1a3b585800d559b5	["chili","baby","back","rib"]	Food

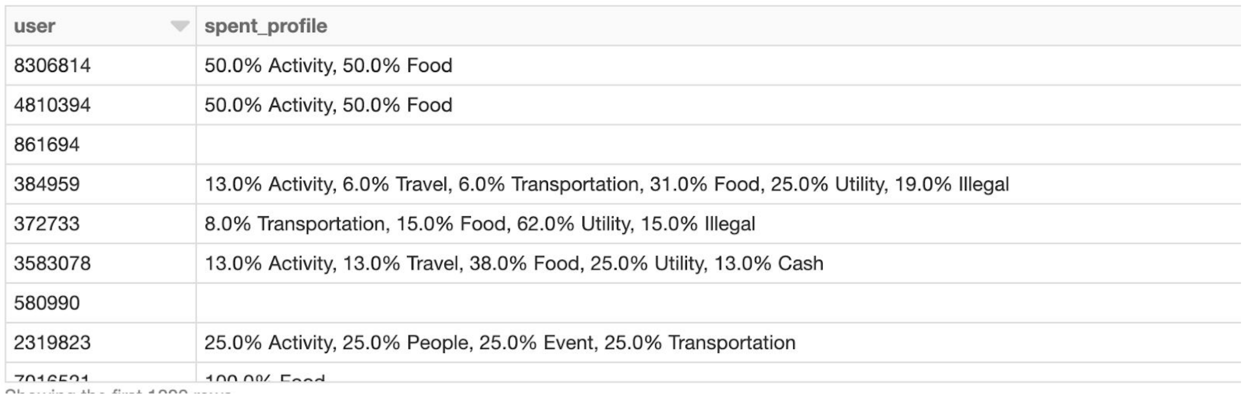
Figure 1: Classify transactions

Q2 [3 pts]: What is the percent of emoji only transactions? Which are the top 5 most popular emoji? Which are the top three most popular emoji categories?

1. 21% of transactions are emoji only
2. the top 5 most popular emoji: '💸', '🍕', '🍻', '🎉', '🍷'
3. the top three most popular emoji categories are Food, People, Activity

Q3 [2 pts]: For each user, create a variable to indicate their spending behavior profile. For example, if a user has made 10 transactions, where 5 of them are food and the other 5 are activity, then the user's spending profile will be 50% food and 50% activity.

We first reshaped the data by combining the transactions of user1 and user2. As we assumed that the transactions that can't be classified by the dictionary would not be considered in the spending behavior, we then calculated the static spending behavior profile for each user.

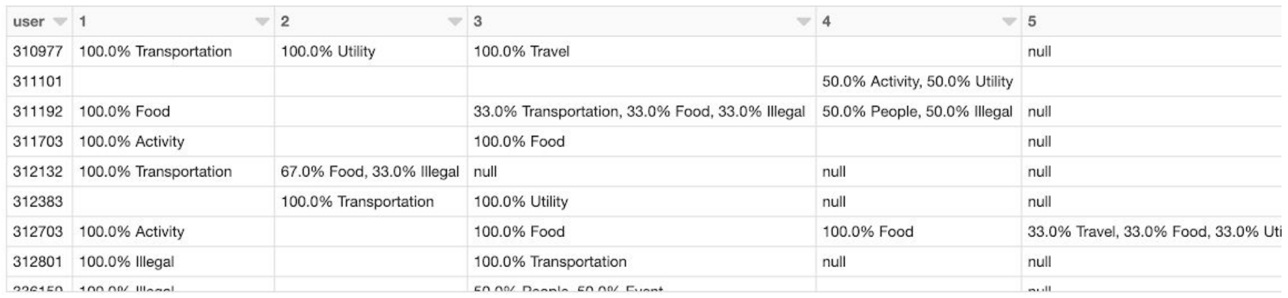


user	spent_profile
8306814	50.0% Activity, 50.0% Food
4810394	50.0% Activity, 50.0% Food
861694	
384959	13.0% Activity, 6.0% Travel, 6.0% Transportation, 31.0% Food, 25.0% Utility, 19.0% Illegal
372733	8.0% Transportation, 15.0% Food, 62.0% Utility, 15.0% Illegal
3583078	13.0% Activity, 13.0% Travel, 38.0% Food, 25.0% Utility, 13.0% Cash
580990	
2319823	25.0% Activity, 25.0% People, 25.0% Event, 25.0% Transportation
7016521	100.0% Food

Figure 2: Static Spending Behavior Profile

Q4 [3 pts]: In the previous question, you got a static spending profile. However, life and social networks are evolving over time. Therefore, let's explore how a user's spending profile is evolving over her lifetime in Venmo. First of all, you need to analyze a user's transactions in monthly intervals, starting from 0 (indicating their first transaction only) up to 12. For each time point, you need to compute the average and standard deviation of each spending category across all users.

We first filtered the transactions that exceed the 12th time points for each user. Then we calculated the dynamic spending behavior profile for each user at each time point.



user	1	2	3	4	5
310977	100.0% Transportation	100.0% Utility	100.0% Travel		null
311101				50.0% Activity, 50.0% Utility	
311192	100.0% Food		33.0% Transportation, 33.0% Food, 33.0% Illegal	50.0% People, 50.0% Illegal	null
311703	100.0% Activity		100.0% Food		null
312132	100.0% Transportation	67.0% Food, 33.0% Illegal	null	null	null
312383		100.0% Transportation	100.0% Utility	null	null
312703	100.0% Activity		100.0% Food	100.0% Food	33.0% Travel, 33.0% Food, 33.0% Uti
312801	100.0% Illegal		100.0% Transportation	null	null
326150	100.0% Illegal		50.0% People, 50.0% Event		null

Figure 3: Dynamic Spending Behavior Profile

We then computed the average and standard deviation of each spending category across all users, at each time point, and plotted the average and $average \pm 2 * standard\ deviation$ area. From the plot, we can see that most of the average spendings stabilized after customers' first life point.

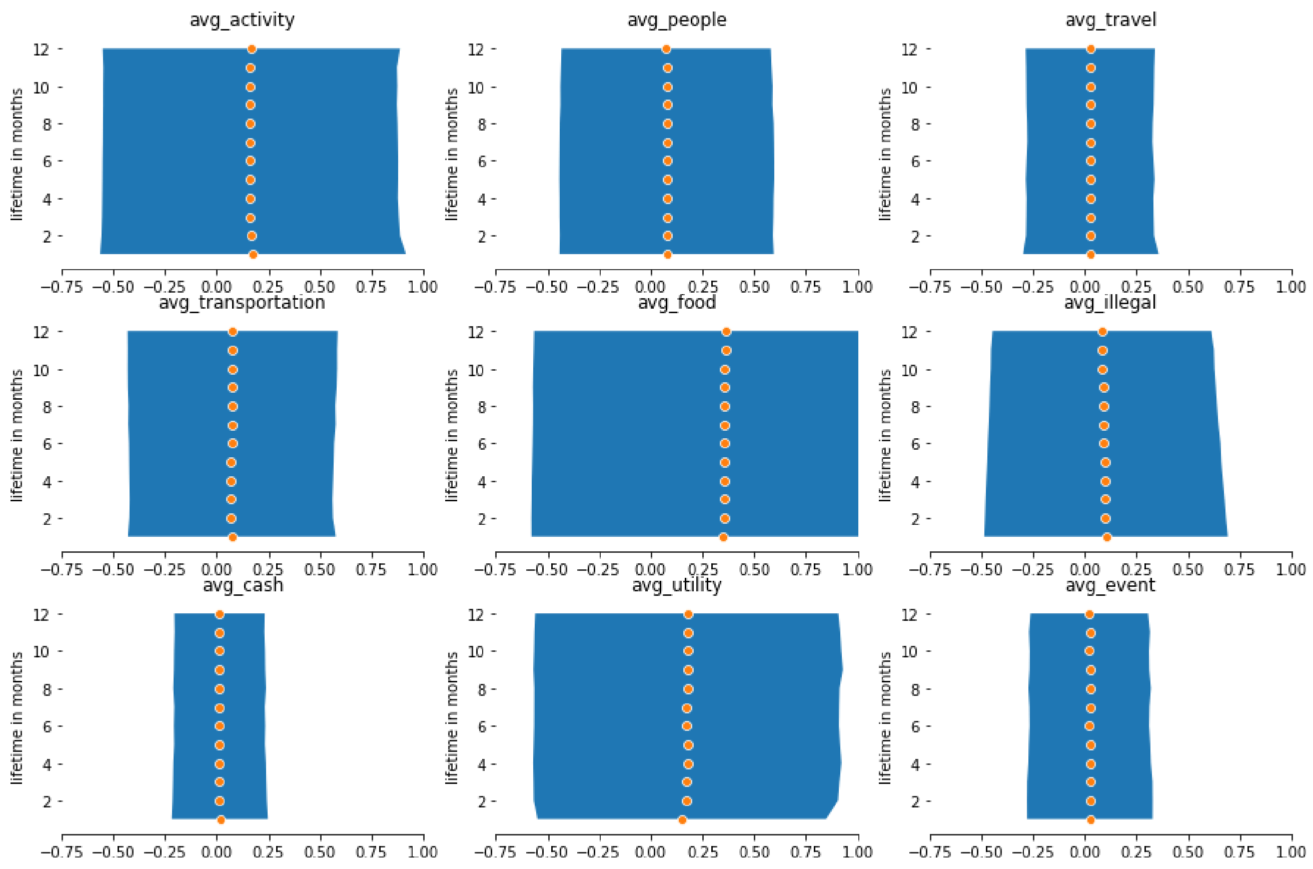
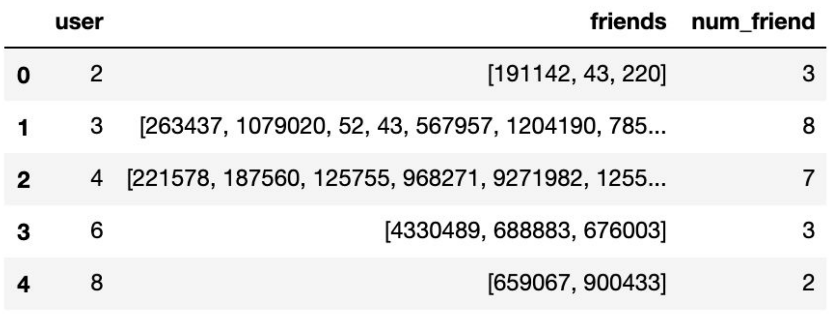


Figure 4: Average and Standard Deviation of Each Spending Category

Q5 [5 pts]: Write a script to find a user's friends and friends of friends (Friend definition: A user's friend is someone who has transacted with the user, either sending money to the user or receiving money from the user). Describe your algorithm and calculate its computational complexity. Can you do it better?

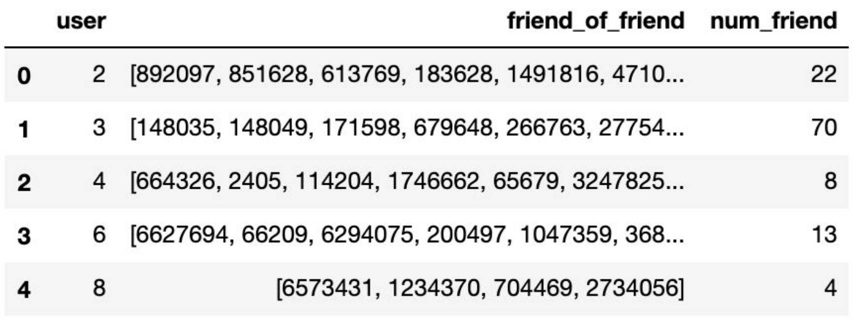
We used the 'networkx' package to acquire friends and friends of friends of all users. After creating a network graph for all users, degree information of all nodes can be obtained easily, which refers to the number of friends of each user in this question. And by using the graph[user] algorithm, friends of each user is shown and can be saved in a list. By running a for loop for all users in this dataset, friends information is acquired and saved in corresponding lists. Then a function named 'single_source_shortest_path

_length' is utilized to get friends of friends information. In this function, the 'cutoff' attribute is set as 2, which means that the result dict will only contain nodes with path less than or equal to 2. Then we defined a function called 'get_key' to get the nodes which have a path length equal to 2. In this way, we also saved information about friends of friends in corresponding lists.



	user	friends	num_friend
0	2	[191142, 43, 220]	3
1	3	[263437, 1079020, 52, 43, 567957, 1204190, 785...	8
2	4	[221578, 187560, 125755, 968271, 9271982, 1255...	7
3	6	[4330489, 688883, 676003]	3
4	8	[659067, 900433]	2

Figure 5. Users' Friends list



	user	friend_of_friend	num_friend
0	2	[892097, 851628, 613769, 183628, 1491816, 4710...	22
1	3	[148035, 148049, 171598, 679648, 266763, 27754...	70
2	4	[664326, 2405, 114204, 1746662, 65679, 3247825...	8
3	6	[6627694, 66209, 6294075, 200497, 1047359, 368...	13
4	8	[6573431, 1234370, 704469, 2734056]	4

Figure 6. Users' Friends of Friends list

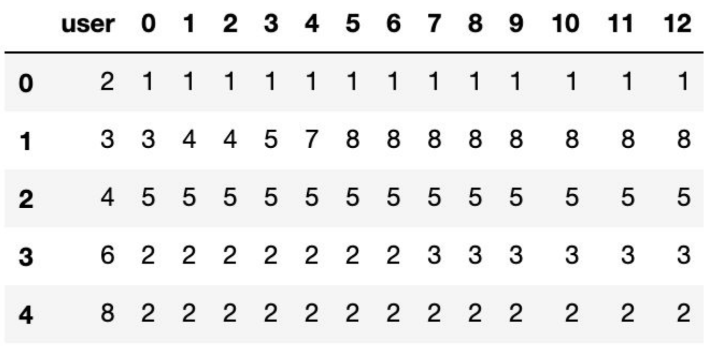
For the computational complexity of our algorithm, we used the most common metric for calculating time complexity, which is Big O notation. This removes all constant factors so that the running time can be estimated in relation to N as N approaches infinity. For a user's friend, our algorithm is linear in that the running time of the loop is directly proportional to N. When N doubles, so does the running time. Here N = all_nodes_unique, which is the number of all distinct users and the complexity of the operation is O(N). When calculating a user's friends of friends, our algorithm included one more loop, which is quadratic. The running time of the two loops is proportional to the square of N. When N doubles, the running time increases by N * N. Here the complexity of the operation is O(N*k), where N = all_nodes_unique and k = friends of friends number for each user. In summary, we think our algorithm is the one of the most efficient solutions for the problem. Compared with the Spark Sql, our algorithm has less computational complexity and takes less time.

Q6 [10 pts]: Now, that you have the list of each user's friends and friends of friends, you are in position to calculate many social network variables. Use the

dynamic analysis from before, and calculate the following social network metrics across a user's lifetime in Venmo (from 0 up to 12 months).

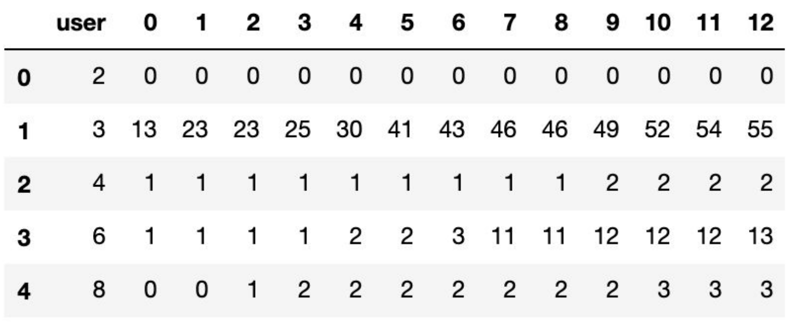
i) Number of friends and number of friends of friends [very easy, 2pts].

Similar to what we did in Q5, but this time we splitted the transactions data in a monthly interval, and calculated the number of friends and friends of friends for each user's lifetime in Venmo(from 0 up to 12 months). Noticed that social networks are evolving over time, so we computed the cumulated numbers at each lifetimes.



	user	0	1	2	3	4	5	6	7	8	9	10	11	12
0	2	1	1	1	1	1	1	1	1	1	1	1	1	1
1	3	3	4	4	5	7	8	8	8	8	8	8	8	8
2	4	5	5	5	5	5	5	5	5	5	5	5	5	5
3	6	2	2	2	2	2	2	2	3	3	3	3	3	3
4	8	2	2	2	2	2	2	2	2	2	2	2	2	2

Figure 7: Number of friends



	user	0	1	2	3	4	5	6	7	8	9	10	11	12
0	2	0	0	0	0	0	0	0	0	0	0	0	0	0
1	3	13	23	23	25	30	41	43	46	46	49	52	54	55
2	4	1	1	1	1	1	1	1	1	1	2	2	2	2
3	6	1	1	1	1	2	2	3	11	11	12	12	12	13
4	8	0	0	1	2	2	2	2	2	2	2	3	3	3

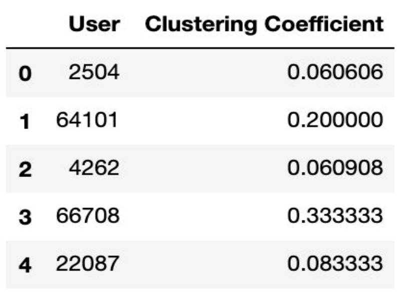
Figure 8: Number of friends of friends

ii) Clustering coefficient of a user's network [easy, 3 pts]. (Hint: the easiest way to calculate this is to program it yourselves. Alternatively, you can use the "networkx" python package. The latter approach will slow down your script significantly).

We used the "networkx" python package to calculate the clustering coefficient of a user's network. In graph theory, a clustering coefficient is a measure of the degree to which nodes in a graph tend to cluster together. Here the local clustering coefficient of a user is computed as the proportion of connections among her neighbours. The local clustering coefficient C_i for a vertex V_i is then given by the proportion of links between the vertices within its neighborhood divided by the number of links that could possibly exist between them. For a directed graph, e_{ji} is distinct from e_{ij} , and therefore for each neighborhood N_i there are $k_i(k_{i-1})$ links that could exist among the vertices within the neighborhood.

$$C_i = \frac{|\{e_{jk} : v_j, v_k \in N_i, e_{jk} \in E\}|}{k_i(k_{i-1})}$$

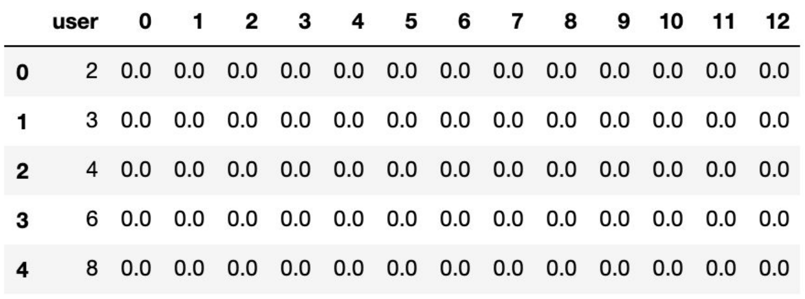
Firstly, we calculated the clustering coefficients of a user for her whole lifetime:



	User	Clustering Coefficient
0	2504	0.060606
1	64101	0.200000
2	4262	0.060908
3	66708	0.333333
4	22087	0.083333

Figure 9. Users' clustering coefficients for their whole lifetime

Next, we calculated the clustering coefficients of users across their lifetime in Venmo (from 0 up to 12 months):



	user	0	1	2	3	4	5	6	7	8	9	10	11	12
0	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figure 10. Users' clustering coefficients across their one year lifetime

Some users' clustering coefficients were 0 because no vertex that is connected to V_i connects to any other vertex that is connected to V_i, which means their friends and friends of friends were not connected to each other.

iii) Calculate the page rank of each user (hard, 5 pts). (Hint: First of all, you need to use GraphX to do this. Moreover, notice that page rank is a global social network metric. If you go ahead and calculate the page rank for each user at each of her lifetime points, you will soon realize it will be a dead end. Can you think of a smart way to do this?)

We used the GraphFrame API to calculate the page rank on spark dataframes, which was fully integrated with GraphX (for RDD and written in Scala). PageRank can be used in any graph to identify the most influential and important nodes/vertices. Here, we want to use PageRank to identify who is the most influential user in venmo circles.

First we constructed the graph, which needed to prepare two dataframes, one for edges and one for vertices(nodes). We unioned the user1 and user2 to create a user_id node dataframe. Then we took the transactions and transaction type to construct the edges table with new column names of "src", "dst", and "relationship".

Next, we constructed the graph and calculated the global pagerank for each user. PageRank algorithm in PySpark will generate a graph with a new attribute/weight assigned to its nodes called PageRank where calculated PageRank scores are stored. And edge weights are normalized based on the number of edges.

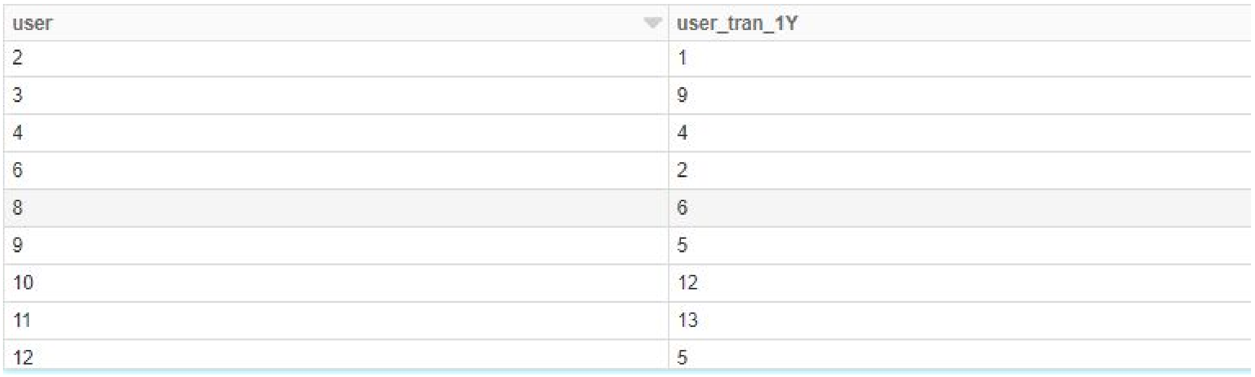
```
+-------+------------------+
|     id|          pagerank|
+-------+------------------+
|1002607|503.38417178353365|
| 351854|173.63907884676289|
| 295387|147.94588892896627|
| 376776|135.30803222562156|
|1307021|   92.372340309255|
|2526756| 73.37582737926496|
|8584533| 62.70260762553058|
|1170172| 60.66482439125626|
|2075857|   58.425662665936|
|5628611|  53.2388011548595|
| 726096| 52.32944170487281|
| 289657| 50.19968973587706|
| 658967| 48.02559045277043|
| 922886|45.376970719229355|
| 726776| 45.20305134902603|
|4108992| 44.53517061616728|
| 517582|43.862404317897195|
|4548542| 39.43101440858216|
|3648238| 39.00613717175025|
|1426491|38.595799846284415|
+-------+------------------+
only showing top 20 rows
```

```
+-----+-------+------------+--------------------+
|  src|    dst|relationship|              weight|
+-----+-------+------------+--------------------+
|   19|  85471|     payment|                0.25|
|  173|  87794|     payment|                 0.2|
| 1009|   6675|     payment|0.030303030303030304|
| 2794|  66024|     payment|0.047619047619047616|
| 4262|  69280|     payment|0.017543859649122806|
| 6172|2373924|     payment|                 0.2|
|24277|  65200|     payment|                0.25|
|26470| 940997|      charge|0.045454545454545456|
|26876|  31061|     payment|  0.3333333333333333|
|37907|  70026|     payment|               0.125|
|39108| 209534|      charge| 0.05263157894736842|
|42801|  73618|      charge| 0.14285714285714285|
|43223|2049989|     payment| 0.09090909090909091|
|44861| 129338|     payment|              0.0625|
|44933|5167380|     payment|                 1.0|
|45008|1692271|      charge|                 0.2|
|46627|  68239|      charge|              0.0625|
|47021|  47694|     payment|  0.3333333333333333|
|48485|  47283|     payment|  0.3333333333333333|
|49753|  52685|      charge| 0.06666666666666667|
+-----+-------+------------+--------------------+
only showing top 20 rows
```

Figure 11. Pagerank and edge weights of Users

Q7 [1 pt]: First, create your dependent variable Y, i.e. the total number of transactions at lifetime point 12. In other words, for every user, you need to count how many transactions s/he had committed during her/his twelve months in Venmo.

First we selected all the unique users, either it's user1 or user2. Then count the number of transactions each user made during his first year. The table below shows a sample of 9 users ordered by user id.



user	user_tran_1Y
2	1
3	9
4	4
6	2
8	6
9	5
10	12
11	13
12	5

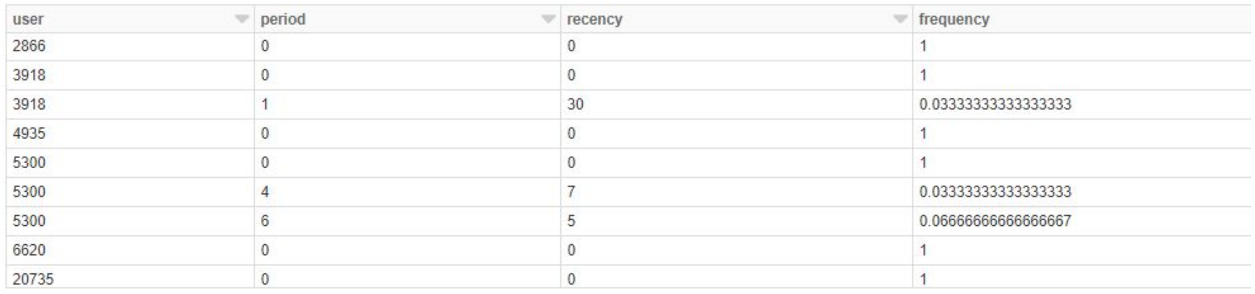
Figure 12. Total number of transactions during user's first year

Q8 [2 pts]: Create the recency and frequency variables. In CRM, this predictive framework is known as RFM. Here, you don't have monetary amounts, so we will focus on just RF. Recency refers to the last time a user was active, and frequency is how often a user uses Venmo in a month. You need to compute these metrics across a user's lifetime in Venmo (from 0 up to 12).

For example, if a user has used Venmo twice during her first month in Venmo with the second time being on day x, then her recency in month 1 is "30-x" and her frequency is 2/30.

First, we defined period 0 as only the first transaction, period 1-12 as 30x days after the first transaction.

For period0, we assume it's only the first transaction that a user made in Venmo, so recency should be 0 and frequency should be 1 for all users in their period 0. For period 1-12 frequency, we counted the number of transactions the user made during that period and then divided by 30. For period 1-12 recency, we first calculated the date difference of each transaction date and first date of that period, then get the minimum date difference within that period as recency. The table below shows a sample of 9 users.

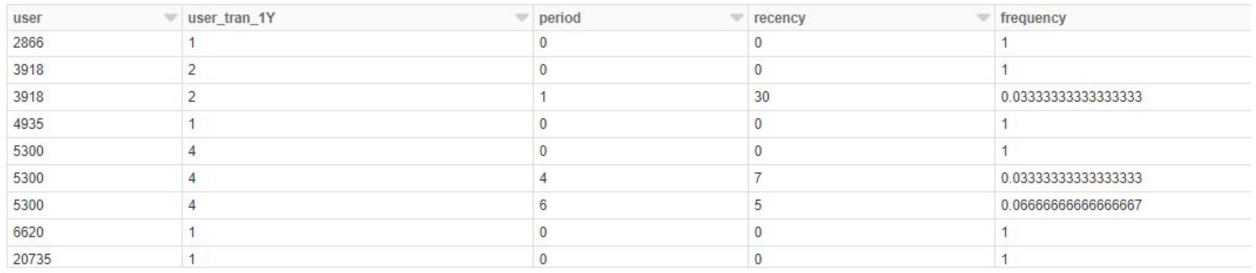


user	period	recency	frequency
2866	0	0	1
3918	0	0	1
3918	1	30	0.03333333333333333
4935	0	0	1
5300	0	0	1
5300	4	7	0.03333333333333333
5300	6	5	0.06666666666666667
6620	0	0	1
20735	0	0	1

Figure 13. User's recency and frequency in each period

Q9 [2 pts]: For each user's lifetime point, regress recency and frequency on Y. Plot the MSE for each lifetime point. In other words, your x-axis will be lifetime in months (0-12), and your y axis will be the MSE. (Hint: Don't forget to split your data into train and test sets).

We joined the dependent variable Y (user_tran_1Y) with recency and frequency and got the regression dataset. In this table, each user has at most one entry in each period.



user	user_tran_1Y	period	recency	frequency
2866	1	0	0	1
3918	2	0	0	1
3918	2	1	30	0.03333333333333333
4935	1	0	0	1
5300	4	0	0	1
5300	4	4	7	0.03333333333333333
5300	4	6	5	0.06666666666666667
6620	1	0	0	1
20735	1	0	0	1

Figure 14. Regression dataset

Then we created subsets for each period and regressed for each period.

In terms of linear regression on PySpark, we need to first assemble all the independent variables then combine the assembler with the dependent variable. After splitting train and test data, feeding in regression, we got a vector with 13 MSEs from 13 periods.

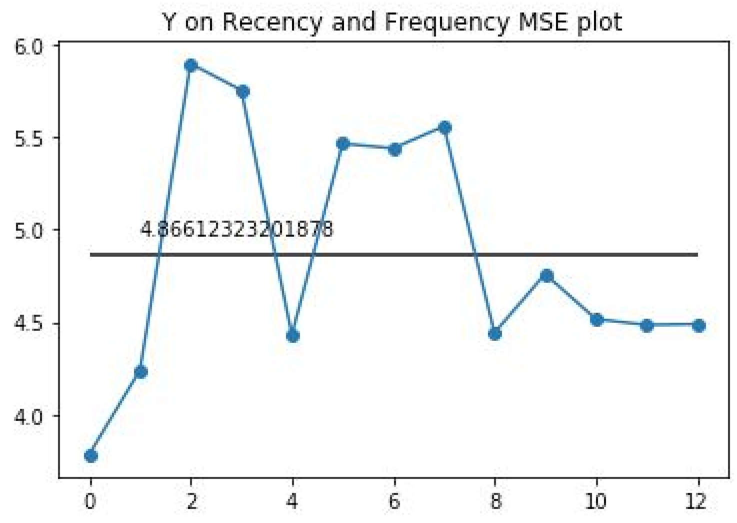


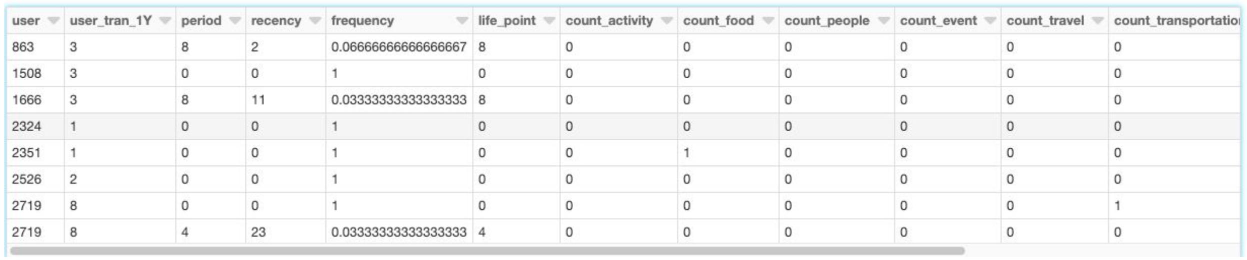
Figure 15. MSE for each period of regressing recency and frequency on Y

Overall, MSE is between 4 to 6. Period 0 has the lowest MSE, 3.77. Period 3 has the highest MSE, 5.73. MSE fluctuates during period 2 to period 8 and stabilizes at around 4.4 after that.

Q10 [5 pts] : For each user's lifetime point, regress recency, frequency AND her spending behavior profile on Y. Plot the MSE for each lifetime point like above. Did you get any improvement?

After mapping the user spending behavior data from part 1 to user level, we again regressed the data for time period 0 to 13. Based on the result of MSE plot shown in the graph below, the mean square error in the initial months (0-5) has dropped significantly. In other words, the user spending behaviors in each time period jointly with user recency and frequency become better at explaining the variance in their transaction numbers. In the later stages, the MSE remains similar to the model with only RF framework containing the Recency and Frequency metrics.

Overall, the average MSE for all 0-12 time periods during the user lifetime has improved compared to the model with only RF metrics. That is to say, the social spending behavior of the users is also contributing to explaining the variance in their transaction frequency.



user	user_tran_1Y	period	recency	frequency	life_point	count_activity	count_food	count_people	count_event	count_travel	count_transportatio
863	3	8	2	0.06666666666666667	8	0	0	0	0	0	0
1508	3	0	0	1	0	0	0	0	0	0	0
1666	3	8	11	0.03333333333333333	8	0	0	0	0	0	0
2324	1	0	0	1	0	0	0	0	0	0	0
2351	1	0	0	1	0	0	1	0	0	0	0
2526	2	0	0	1	0	0	0	0	0	0	0
2719	8	0	0	1	0	0	0	0	0	0	1
2719	8	4	23	0.03333333333333333	4	0	0	0	0	0	0

Figure 16. Regression dataset

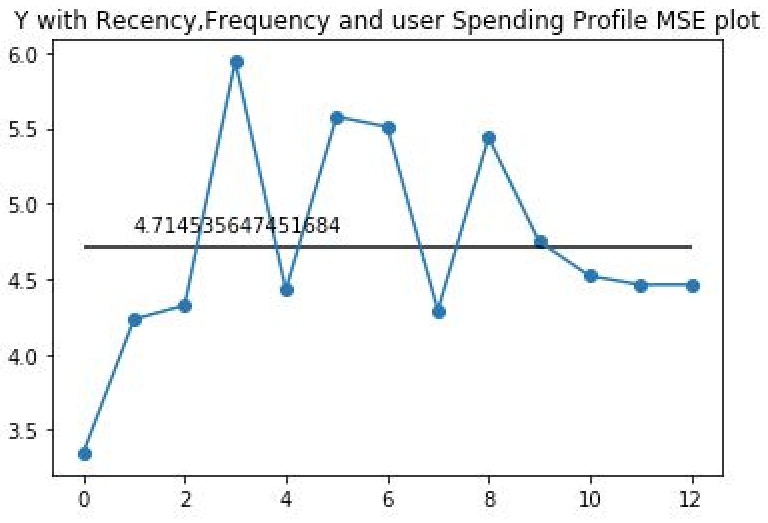


Figure 17. MSE for each period of regressing recency, frequency and spending profile on Y

Q10 [5 pts] : For each user's lifetime point, regress her social network metrics on Y. Plot the MSE for each lifetime point like above. What do you observe? How do social network metrics compare with the RF framework? What are the most informative predictors?

The social network metrics, compared with the RF framework, is performing better in the linear regression model based on the MSE plot shown below. The average MSE for the models across different months in user lifetime dropped from about 4.8 in the RF framework model to 3.3 in the social network metrics model. Thus, we can conclude that the social network metrics is better at explaining the variance in user's transaction frequency.

As for the most informative predictors, we looked at the coefficient of each feature in the linear regression model output. The bigger coefficients include the friend, coefficient and pagerank metrics. As shown in the below table.

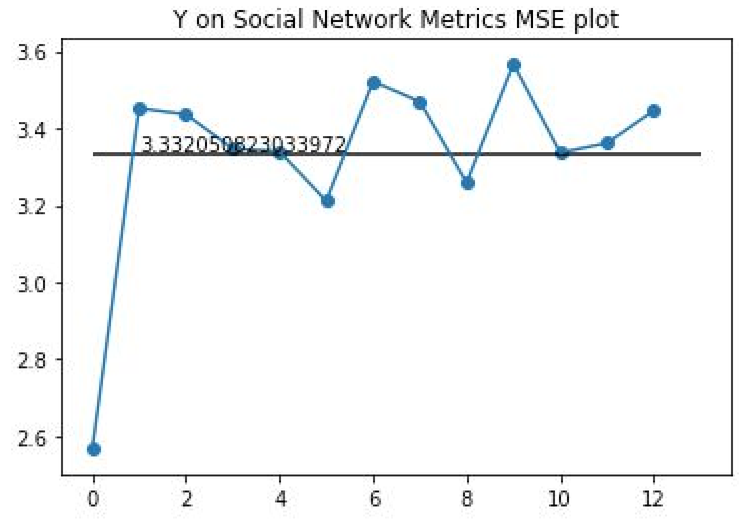
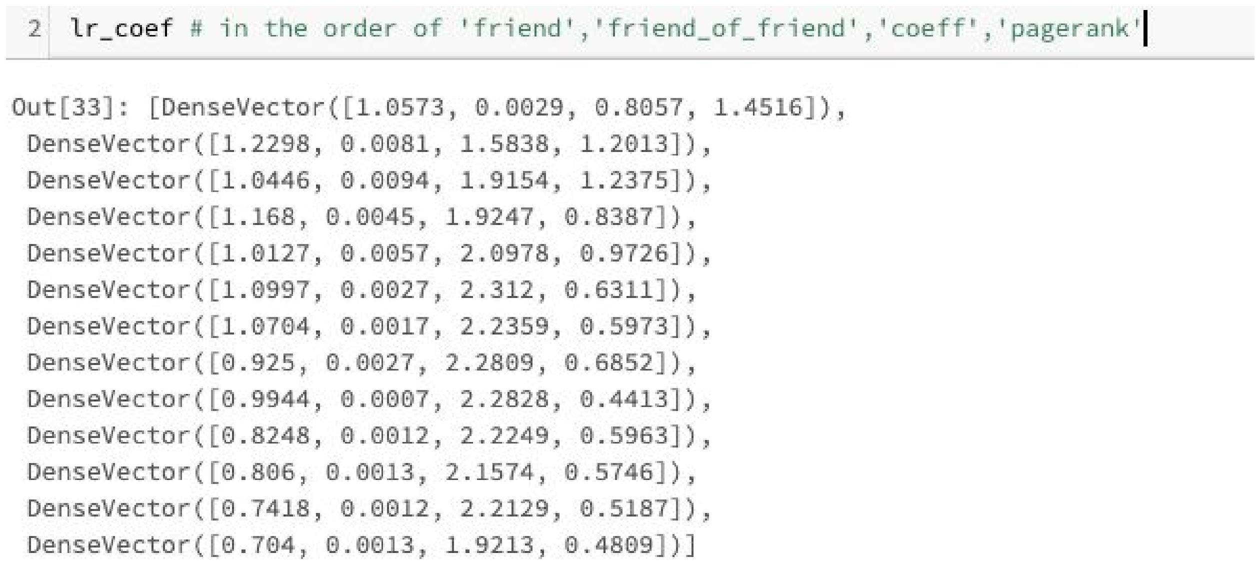


Figure 18. MSE for each period of regressing social network metrics on Y

As can be seen from the coefficients table of all the models, the friend, clustering coefficient and page rank metrics have larger coefficients thus are more important in the model. Among the three metrics, the clustering coefficient metric in social network analysis has the largest coefficients thus is greater at generating predictive power for the Y variables. With one unit of change in these variables, it will lead to bigger change in the Y variable. Thus, these metrics are more significant in explaining the variance in the transaction frequency Y variable.



```
lr_coef # in the order of 'friend','friend_of_friend','coeff','pagerank'
```

```
Out[33]: [DenseVector([1.0573, 0.0029, 0.8057, 1.4516]),
 DenseVector([1.2298, 0.0081, 1.5838, 1.2013]),
 DenseVector([1.0446, 0.0094, 1.9154, 1.2375]),
 DenseVector([1.168, 0.0045, 1.9247, 0.8387]),
 DenseVector([1.0127, 0.0057, 2.0978, 0.9726]),
 DenseVector([1.0997, 0.0027, 2.312, 0.6311]),
 DenseVector([1.0704, 0.0017, 2.2359, 0.5973]),
 DenseVector([0.925, 0.0027, 2.2809, 0.6852]),
 DenseVector([0.9944, 0.0007, 2.2828, 0.4413]),
 DenseVector([0.8248, 0.0012, 2.2249, 0.5963]),
 DenseVector([0.806, 0.0013, 2.1574, 0.5746]),
 DenseVector([0.7418, 0.0012, 2.2129, 0.5187]),
 DenseVector([0.704, 0.0013, 1.9213, 0.4809])]
```

Figure 19. Regression coefficients of the variable 'friend','friend_of_friend','coeff','pagerank' between month 0-12.

Q11 [5 pts] : For each user's lifetime point, regress her social network metrics and the spending behavior of her social network on Y. Plot the MSE for each lifetime point like above. Does the spending behavior of her social network add any predictive benefit compared to Q10?

We mapped the data of social network metrics and user's spending profile together based on each individual month of every user. And then regress these metrics on the number of transactions. The result MSE plot compared to the previous ones, are similar and did not result in a huge drop in the MSE. Thus, the spending behavior and social network metrics together did not add too much predictive power to the model.

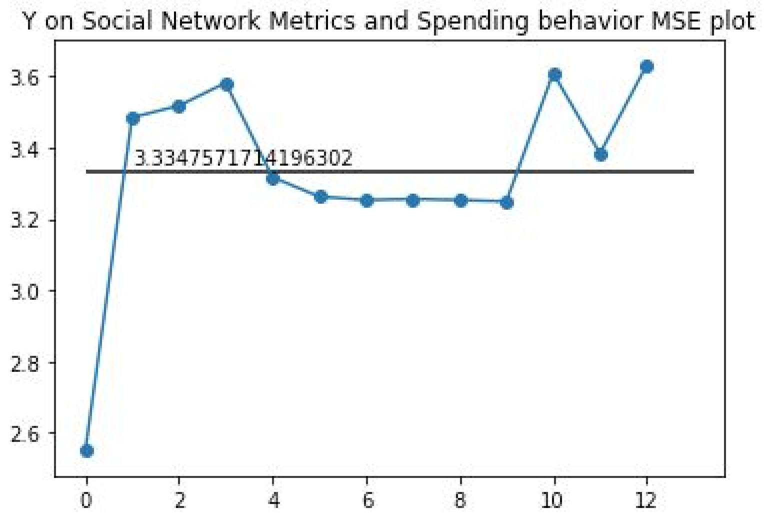


Figure 20. MSE for each period of regressing social network metrics and spending profile on Y