Data Science Project B: Social Network Case

Contents

Executive Summary

This report was written on behalf of the E4-2 Consultant Group.

The objective of this report is to present findings regarding the data provided by a social networking site. These findings include helping the client understand its data, and leverage it as a competitive advantage, ultimately leading to an increase in revenue. The two datasets provided, “interests.csv” and “followers.csv” included information regarding the following history of users, as well as their interests. From these files, valuable insights were generated which are expressed in the remainder of this report, which has been segmented into three parts: Parts A, B, and C.

Part A provides an overall understanding of the networks’ users and their interests. As well, it contains social network diagrams and their comparison to a well studied social network, Twitter, done using NetworkX. In this section, metrics are introduced to measure the network connectivity and information flow. This was done so that the social networking site can provide tangible evidence to VC firms that it is possible to increase connectivity through a recommender system.

Part B details a recommendation engine that was created in python to predict which user will follow another user so the company can work towards a “push model” where they can then identify who a user should follow and recommend them. This section also contains information on how this model was validated and assessed to measure performance.

Part C of this report contains recommendations for the social networking site for future data analysis. It also contains a high-level strategy the client could use to evaluate the classifier if it gets deployed in a business setting. It is highly recommended that the client uses the classifier presented in this text so they can achieve the connectivity required by venture capitalists.

Part A: Data Understanding

This data set contains 3,599 unique users and 884 interest categories. The distribution of interests in the entire population can be demonstrated in Figure A.1.A, Figure A.1.B and Figure A.1.C. Figure A.1.A shows the population interest distribution visualized in a bar chart. Certain interest categories show a high number of user interest, which will be explored later in this report.

Figure A.1.A

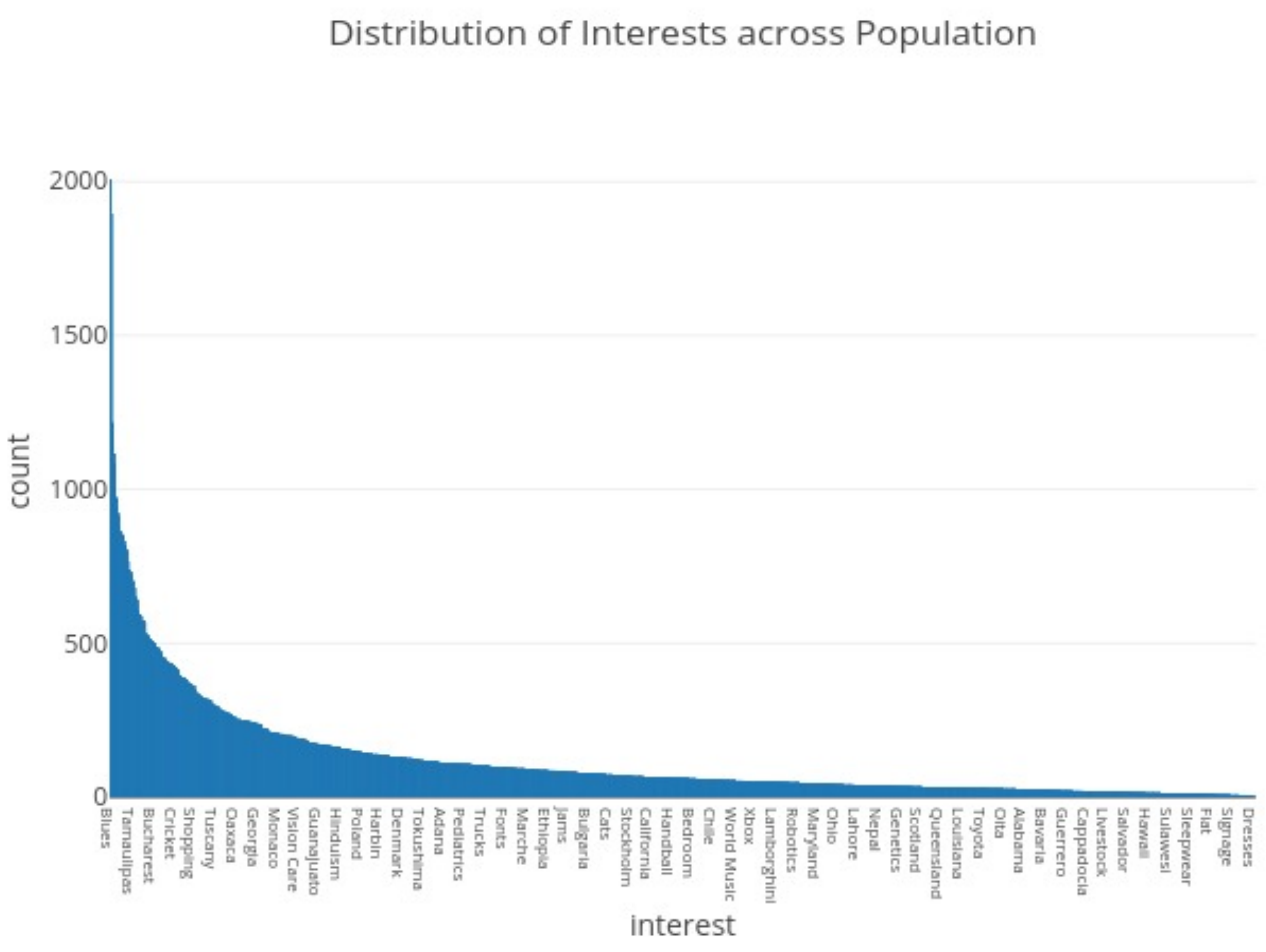


Figure A.1.B. below shows a bubble chart of the number of users per interest category. The darker the bubble, the higher the number of users for that interest. This confirms the result found in Figure A.1.A, showing an uneven distribution of user interest.

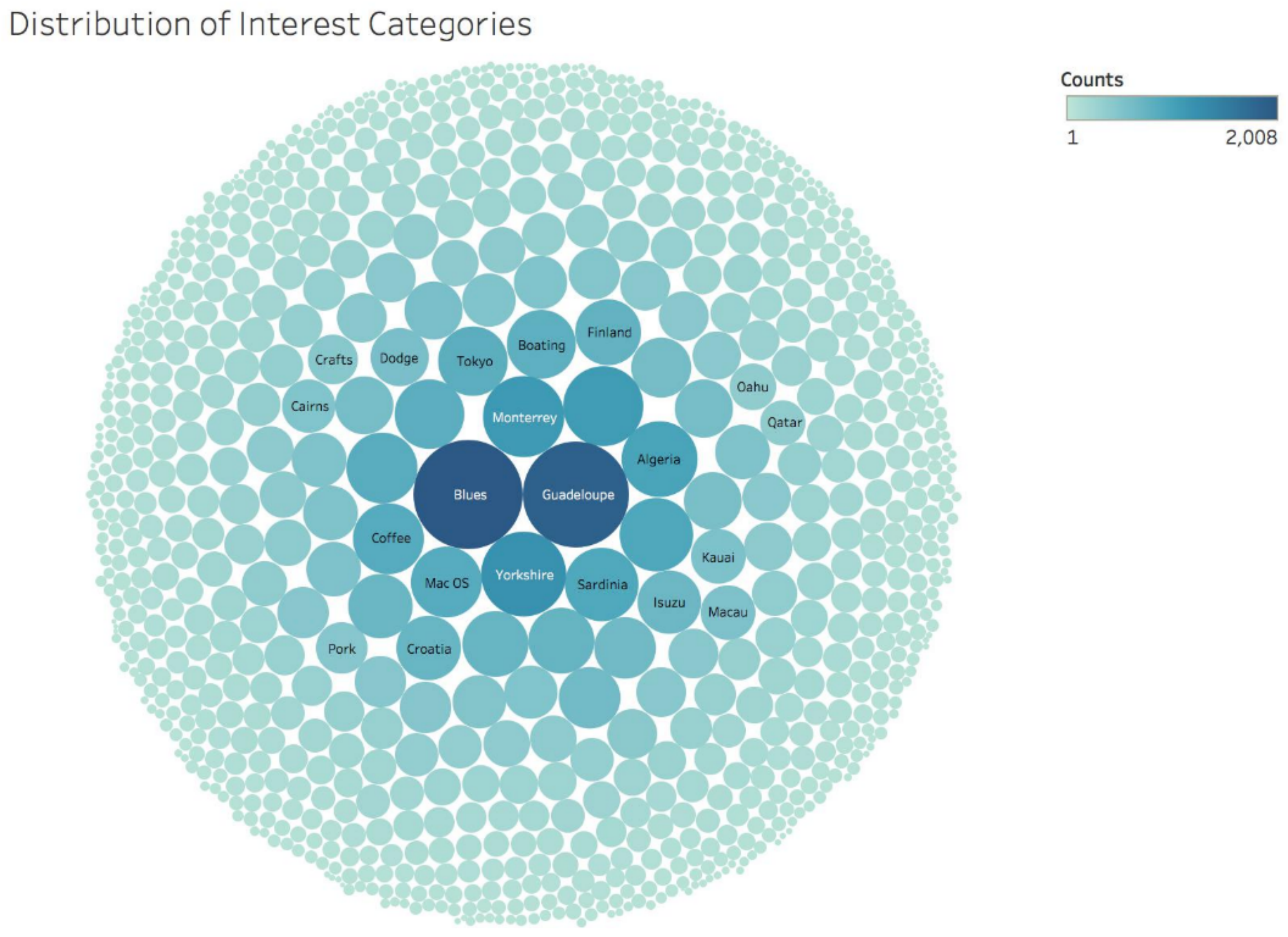
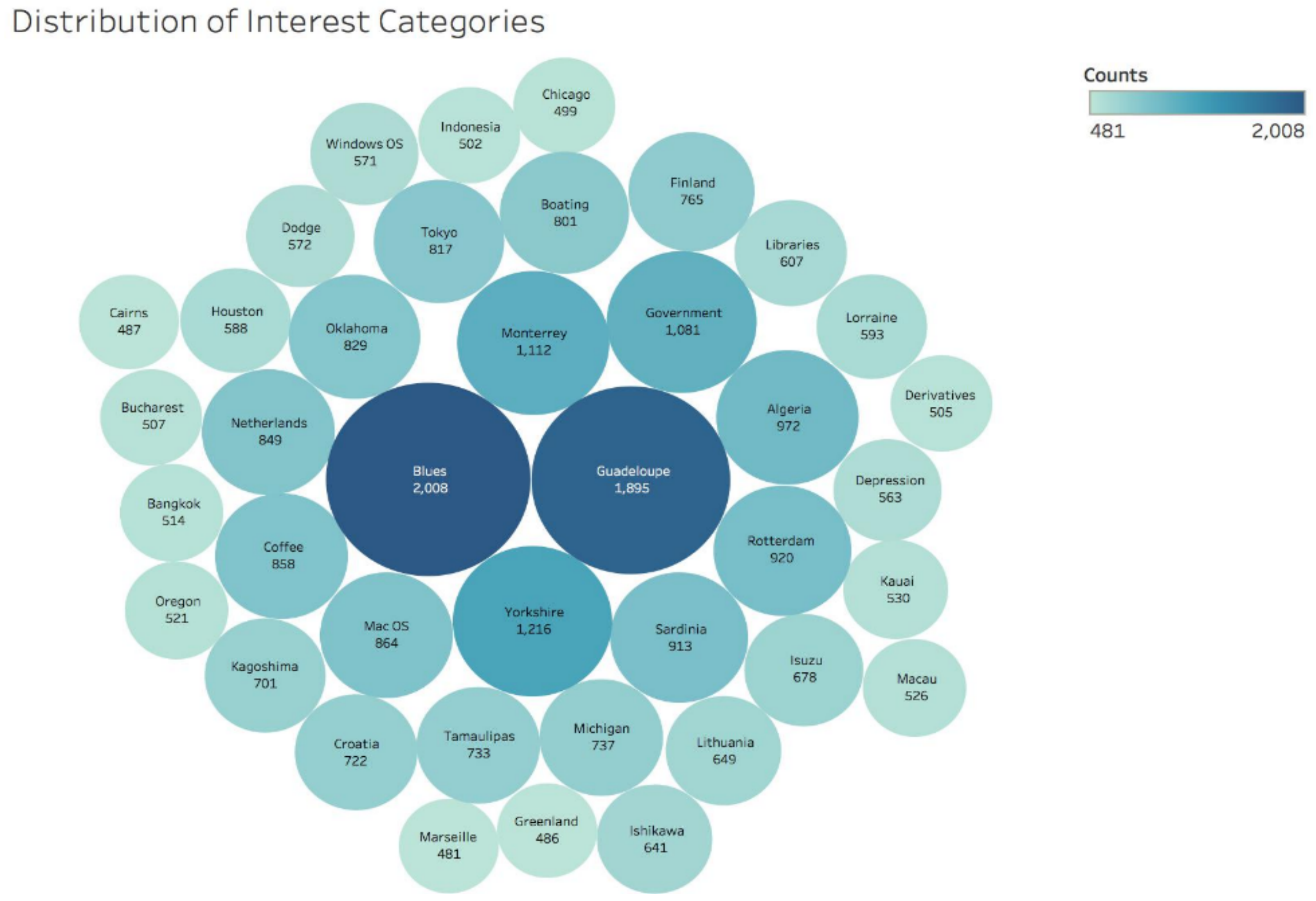


Figure A.1.B

Figure A.1.C below shows the 40 most common interests with counts of 481-2,008. Common interests include: Blues, Guadeloupe, Yorkshire, and Monterrey. The remaining 884 interest categories have counts between 1-480.

Figure A.1.C



The number of declared interests for each user is shown below in Figures A.2.A and A.2.B. A.2.A shows a histogram of interest distribution for each user. As can be seen from the figure, 652 users have 0-10 interests, 768 users have 10-20 interests and as the number of interests increase, the number of users keep on decreasing. The tail of the graph shows some bins which have only 1 user.

Figure A.2.A

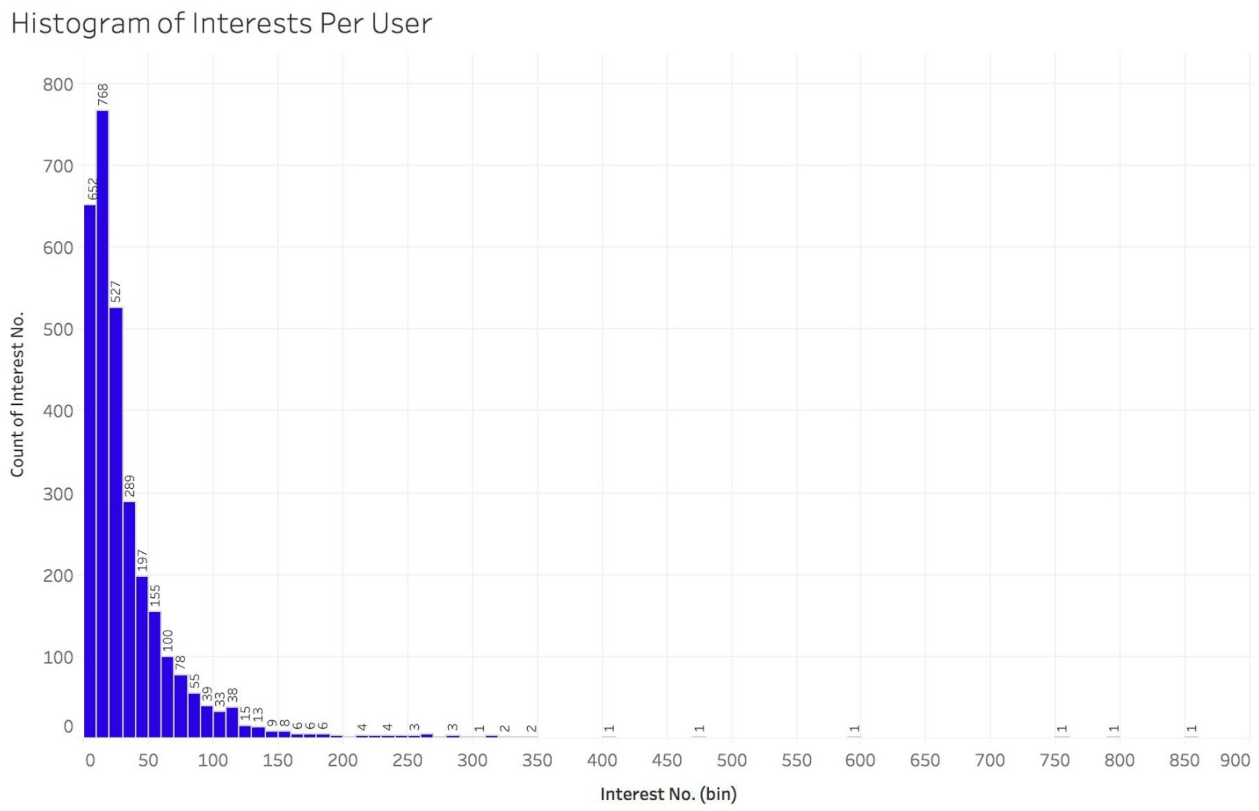
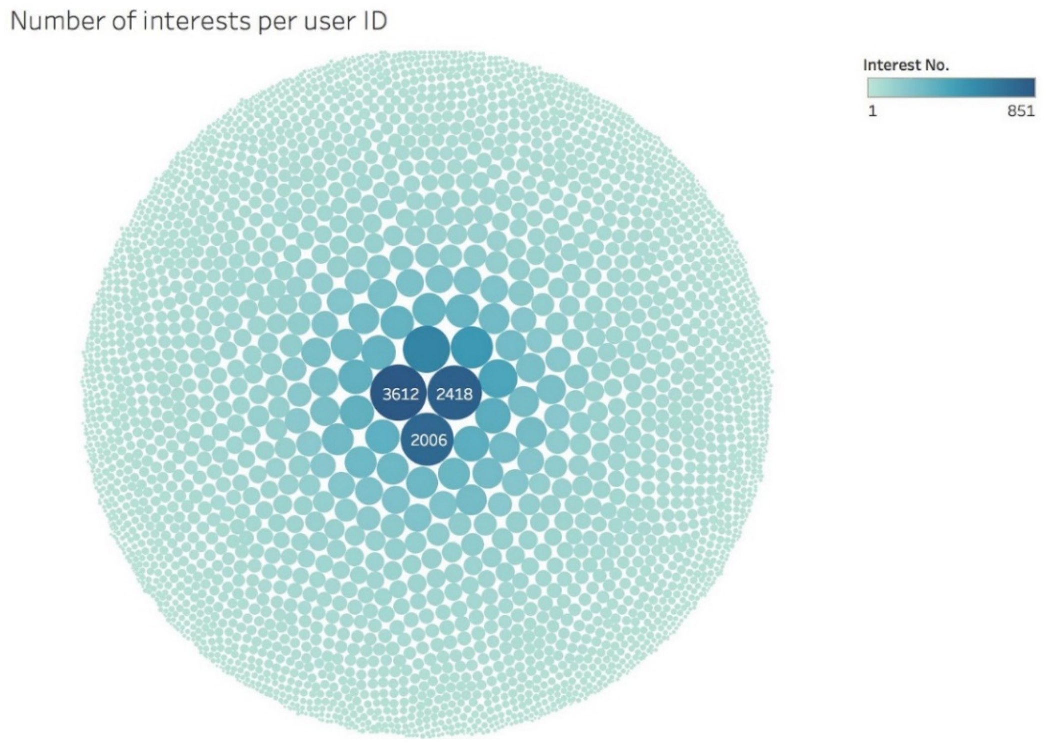


Figure A.2.B shows the number of interests per user visually using a bubble chart. Darker bubbles indicate a higher number of declared interests for that particular user. A few users have a very high number of declared interests. For example, User 3612 has the most declared interests, which is 851.

Figure A.2.B



Social Media Metrics

To visualize the user network and explore metrics relevant in a business context, the NetworkX python package was used. This package is used to create, manipulate, and study the structure and dynamics of complex networks.[1] Trying to plot the entire network (see Figure A.2 and Figure A.3 below) gave very little business insight visually, however the metrics obtained were quite useful.

Figure A.2

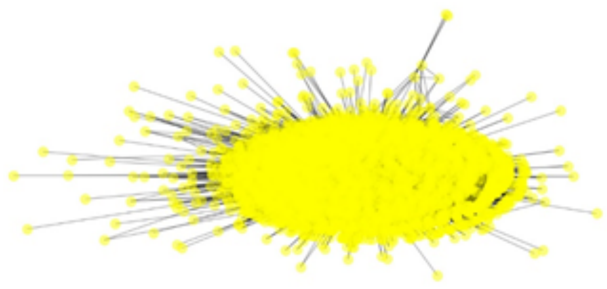
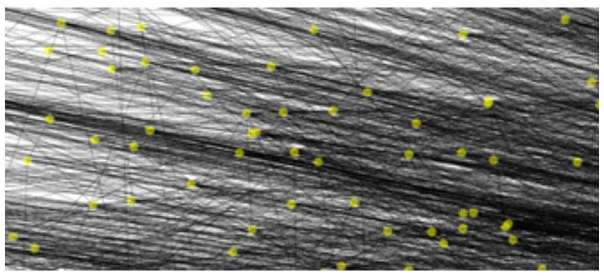


Figure A.3



[1]No Author. (Jan 2018).

Figure A.2 shows the network with user labels not shown. Figure A.3 shows a 30x zoom on the graph, with each yellow node indicating a user and each black line, or edge, indicating a connection between users. The number of nodes (users) was found to be 3599, while the total number of edges (connections) was found to be 37,809. In terms of social network metrics, the business objective is to increase the number of nodes and edges in the network diagram, to increase the total connectivity and number of users. Figure A.4 shows a much simpler network diagram with 23 nodes, to illustrate these concepts, while Figure A.5 shows clusters of users naturally forming (dense areas of the graph).

Figure A.4

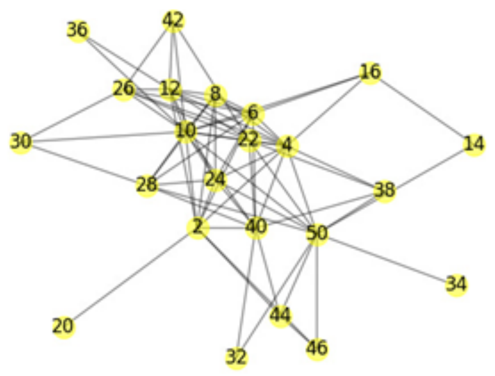
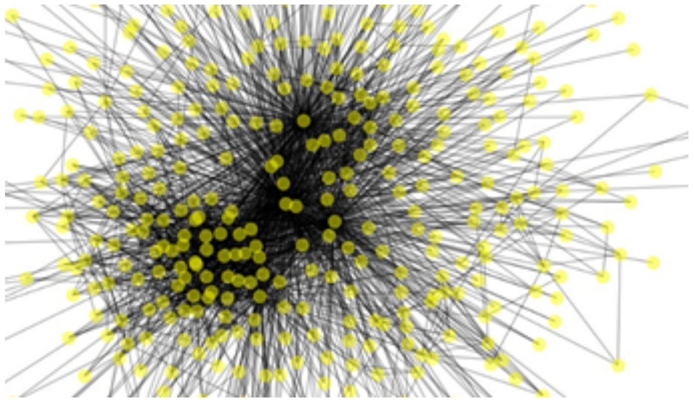


Figure A.5



To better understand the network, a degree distribution was plotted to further explore network characteristics. A node (user) with a degree of one means the user has one connection within the network. To show the degree distribution across the entire network, the Number of Connections (degree) is plotted against the Number of Users in Figure A.6 and Figure A.7.

Figure A.6

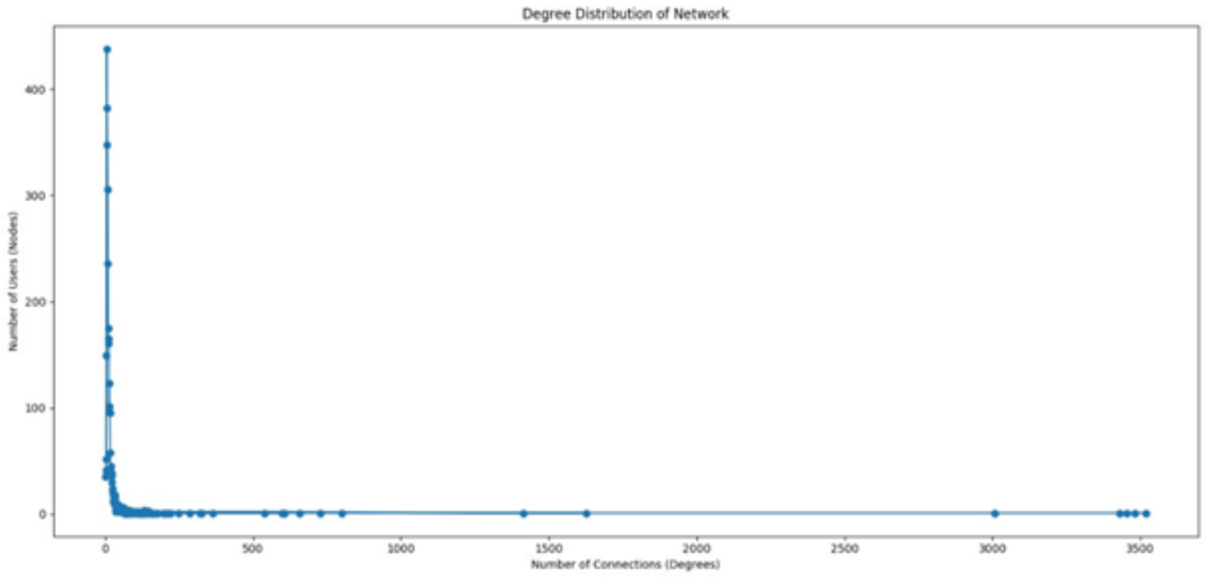


Figure A.7

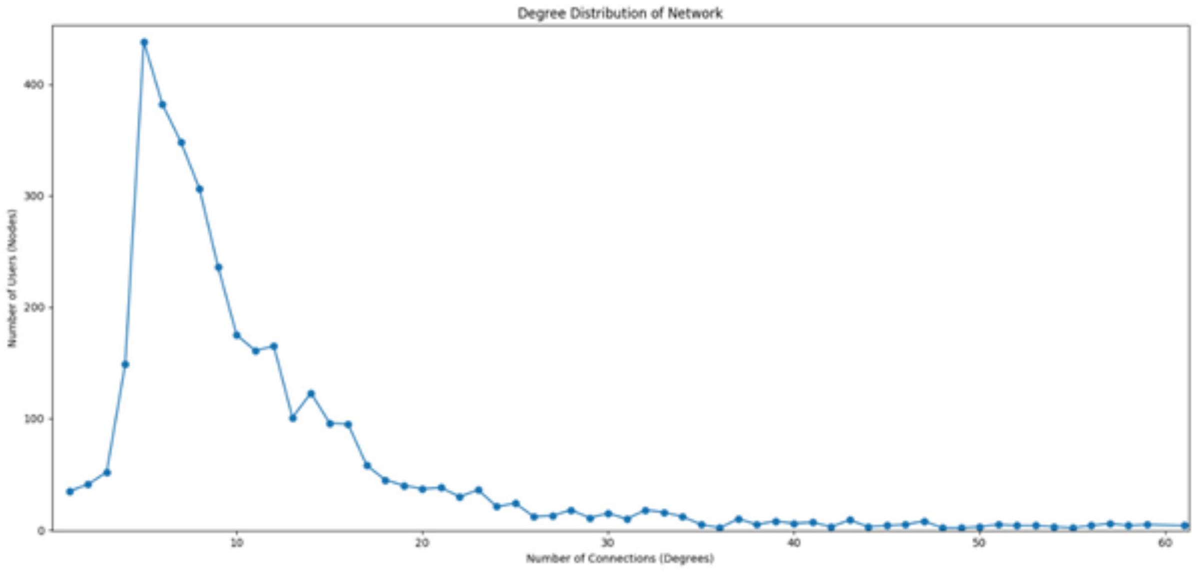
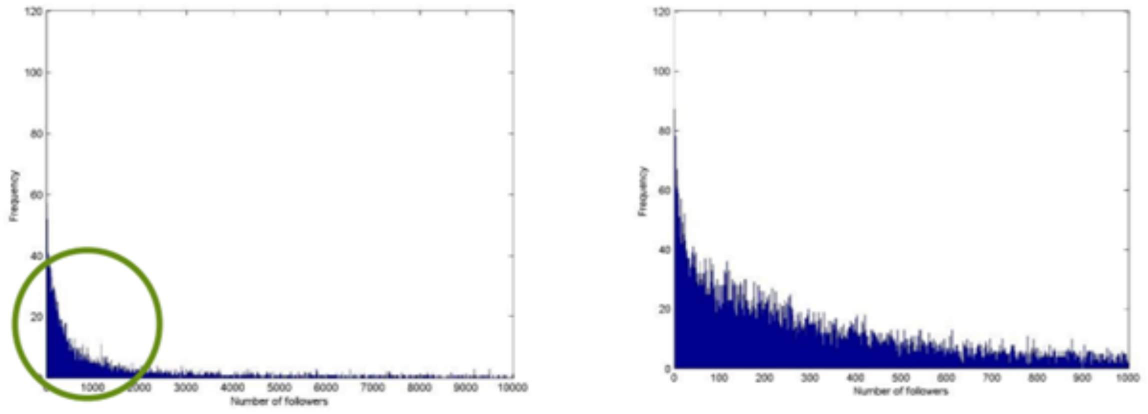


Figure A.6 shows a heavy right skew, meaning there are many users with few connections. For example, the data point at the peak of Figure A.7 (7,440) means there are 440 users with 7 degrees (connections). An interesting note is the 6-7 users who have more than 3000 connections, possibly indicating “social media influencers”.[2] These users likely have influence within the network given the high number of connections, making them prime targets for advertising and close monitoring by the site. Zooming in on the left-hand side of the distribution in Figure A.7 shows the average degree in the above network is 21.01, meaning that on average, each user has 21 connections. As a social media site, a high number of users with a high number of connections (degree distribution pulled to the left) would be ideal, as it indicates more activity in the network. These results are quite similar to Twitter in terms of shape, as seen in Figure A.8. The graph shows a random sample of 15,000 Twitter users, and as expected, shows a high number of users with relatively few connections.[3]

Figure A.8



The average degree in the above network is 6877, as two users have as many as 300,000 followers. Removing those two users gives an average degree of 587. It is interesting to note that the removal of users with a high number of connections can dramatically decrease the average degree, which will be discussed further in Part C. Obviously, an industry leader for social network sites, Twitter, has a much higher average degree than this site (587 vs 21), but these metrics give

[2] Moyer, J. (2012).
[3] Chheda, R. (2016).

a benchmark target for the business to move towards. In order to show increased connectivity and information flow within the network to potential VCs, the average degree measure should be used. This metric directly shows how active users within the site are, which is useful information to attract new users. The number of connections, or edges, is also a useful metric to gauge overall connectivity within the network and can be used to show company success over time.

Part B: Recommendation Engine

The recommendation system was created using two different Jaccard indices as features: a Jaccard index measuring similarity between user interests and a Jaccard index measuring similarity between user people they follow (followees). The Jaccard index is a frequently used metric to measure the similarity or diversity in sample sets. It is calculated as the intersection of either interests or followees over the union of the same between two users. The Jaccard Indices were represented in two large matrices, where each cell contained the Jaccard index for a pair of users.

To appropriately train the model, outliers were discarded from the training and testing set. Outliers were defined as users with an extreme number of followees or users. What constituted as extreme was determined by examining plotted distribution curves. Users with no followees, no followers and/or no interests were not removed as users with these characteristics were common in the dataset. In addition, the recommendation classifier showed unexpected results when such users were removed.

After removing outliers, an “oversampled” balanced sample dataset was created. It was balanced in the sense that it contained an equal number of pairs where the one user was following the other and pairs where the user was not following the other. Since the majority of users in the network are not following each other, a model that was trained on an unbalanced set would encounter pairs that follow each other so infrequently that it would have a hard time learning what causes one user to follow another. Each pair of users in the sample set was assigned a value of zero (not following) or a one (following). This became the target variable.

The balanced sample data set was split into training and tests so that the model could be evaluated on data the model did not see during training. 80% of the data went into the training set and 20% went into the test set. Three models were trained using different methods to find the model with the best accuracy: logistic regression, neural network, and SVM. Each were evaluated and compared using accuracy. The model with the highest accuracy rating was the neural network with 58.57% accuracy. However, there are limitations to assessing the model based on accuracy. Given that the data provided is just for one time-step, accuracy is based on whether or not the user is already following the recommended user, not whether they would or should follow them. Because, up until now, users have found each other through searches, it is quite possible that many people who should or would like to be connected or not. To appropriately evaluate the model, the test model will have to be operationalized. Further discussion on model evaluation is included in Part C.

For each user the model assigns a probability that the user will follow each and every other user in the dataset. The algorithm was designed to recommend the five users with the highest probabilities for them to follow. The model assigns users with no overlapping interests approximately a 50% chance of following each other. This translates into approximately a 0.4% chance of following each other in a dataset that has not been oversampled. This is likely the result of the fact that many of the users following each other have fairly low overlap in terms of both interests and followees, teaching the model that users will follow each other, with little observed basis, about half the time. As the dataset is being oversampled, the probability of the model trained is much higher than the actual probability. Currently, there are 49,506 pairs of following out of 12,960,000 user pairs. So, the prior probability is 0.381% and the data was oversampled to achieve a posterior probability of 50%. For this reason, the model produces higher probabilities for each input. In order to remove the effect of oversampling, and avoid potentially annoying users of the social networking site, a threshold was implemented below which the recommendation of current system will not be used. The problem of not having sufficient information to provide recommendations is referred to as the "cold start" problem. This was addressed by augmenting the recommendation classifier. If there are less than five users with a score of over 53 percent, up to five of the "most popular" users in the network will be recommended. These five users are those with the largest followings: 42, 12, 196, 26, 144. The threshold should be re-evaluated after the recommendation system is put into use. This recommendation model can now be deployed to make appropriate recommendations to increase number of user follows, network connectivity, and revenue.

It is important to note that assumptions were imposed for the model to be developed. These assumptions are:

- The total number of users on the networking site is represented by the users in the follows, followee and the interests lists. This may not be true as there may be users with no followers, no followees and no interests.
- Users with many followers are celebrities or "personalities" that have made themselves known on the networking site.

Part C: Future Data Analysis

One recommendation for additional analysis would be to evaluate the correlations between interest categories, as this might provide better insights for the social network site on what content its users are interested in. One way to analyze these correlations would be a k-means clustering method that would classify similar interests together. For example, if this was done and one interest cluster included countries and travelling, the site could make the hypothesis that users are interested in vacations. This information can then be used to attract advertisers or customize the user experience in some way with the objective of increasing revenue.

As was noted in part A, a small number of users with a large number of connections can greatly increase the average degree across the whole network (remember Twitter's average degree went from 6877 to 587 by removing two users). If average degree is the metric used by VC's to measure network connectivity, attracting a few popular users to the site could be the difference between securing an investment or failing to obtain funds. Understanding which interest categories users on the site are interested in becomes much more critical given this context, as this information can be used to target relevant individuals within that interest category. For example, if users on the site are found to be very interested in sports, the site can try to entice popular users within the sporting world to use the site.

In terms of advanced metrics for popular social media sites, degree distribution and cluster coefficients are two of the most popular methods.[4] For the analysis in Part A, degree distribution was much more useful to determine network insights and prove to VC's connectivity could be increased. Cluster coefficients measure the tendency of a graph to form clusters of highly-related users, which can be used for individual users. The coefficient is the ratio of number of links

[4] Aparicio, S., Alvarez, G. (2015).

between neighbours and maximum number of links possible. Moving forward, it is recommended that the social networking site calculate cluster coefficients for individual nodes, to create a targeted maximum number of connections for each user.

Recommendations for Future Data Dives

To better assess this classifier, one method would be to benchmark the classifier against a simple classifier to see if it has a higher accuracy. If the accuracy result was less than the baseline model, this would be indicative of a poorly trained model. If the accuracy result was greater than the baseline model, it would mean that the classifier is useful.

To increase the performance of the classifier, more information about the users would need to be known. This includes demographic information, similar to what Facebook collects. Its website states they use "mutual friends, work and education information, networks you are part of, contacts you've imported and many other factors"[5] to recommend people a user may know. As well, if the users' location were provided it would improve the classifier because it adds another dimension that shows a higher chance people may know each other in real life. Bao, Zheng, Wilkie, and Mokbel state that "location data bridges the gap between the physical and digital worlds and enables a deeper understanding of users' preferences and behavior."[6] This could potentially increase the connections made as people are more likely to connect online if they know someone in person or have more in common with.

Another component that would be helpful for the future is to have the messages people have broadcasted. From this, a sentiment analysis could have been performed to understand user likes/dislikes, opinions, feelings, and recommend them people with similar mindsets. It is

[5] Facebook (2017).
[6] Bao J, Zheng H, Wilkie D, Mokbel M (2015).

important to note all these methods will likely improve the classifier performance, but they all come with business implications. Customers are more cautious than ever about what data they give up, and trying to obtain this data may hurt the business more than any benefits from a classifier performance increase. The recent example of data issues surrounding Facebook and the subsequent drop in company value is a warning of what can happen when a company is not transparent in data usage.[7]

High Level Strategy to Evaluate the Classifier

One method in which the company can evaluate the classifier's operation is to have ongoing metrics that provide information on how the recommender is doing for the company's KPIs. Looking at the number of edges within the network would be an effective metric to measure the increase in connections per day. "Amount of time gone by between recommendation is shown and it is clicked", and "times recommendation was shown but not clicked" would provide useful information on how the recommender is doing, and if it is improving over time. However, this method entails the company to collect more information regarding the connections and having someone dedicated to evaluating and comparing these numbers regularly.

To accurately measure the recommender performance, it is suggested to deploy the classifier in an A/B testing method. By using this classifier on a test group (50% of the users), and a control group (other 50% of the users), the social network site can evaluate if the model is producing more connections. There are three options to establish a baseline for the A/B testing method: a model which recommends random users, a model which recommends most popular users ("celebrities"), or not recommend any connections and allow the network to continue

[7] Castillo, M. (2018).

growing through customer pull. Given the distribution of users (select few who have many connections), the baseline model chosen is the one that recommends popular users of the site. The evaluation would be done as follows:[8]

1. Randomly split the users into *Group A: Control Group* and *Group B: Test Group*
2. Show Group A recommendations to connect with popular users, and show Group B recommendations with the proposed classifier.
3. Collect results and enter into the graph C.1 (found below)
4. Use Fisher's Exact t-test to evaluate significance of association between results
5. Evaluate results: implement classifier if the test group performs better than the control group

Chart C.1

	Group A: Control Group	Group B: Test Group	Row Total
Times follow occurs			
Time follow does not occur =(Times served - times follow occurs)			
Column Total =(Times served)			

If it is found that the proposed classifier does not perform better than the control group, it is recommended that an investigation into why this occurred be started so the model can be improved and modified accordingly. If it does perform better, full integration and implementation in the website should begin.

[8] Sonberg, J. (2017).

Appendix A

The hardware used for this analysis was XPS13, Dell. It took 2.5 minutes for the programs to run and produce the accuracy rating. For further analysis of large volumes of data, it is recommended to use Hadoop to process the data and train the classifier.

Appendix B

```
#PART-A

#How many users are there?
import csv
import numpy as np
f = open('follows.csv')
csv_f = csv.reader(f)
follower = []
followee = []
for readRow in csv_f:
	follower.append(readRow[0])
	followee.append(readRow[1])
U=list(zip(follower, followee))
UA=np.array(U)
Unique_Users=np.unique(UA) #this has the column headings in it too
Users=Unique_Users[1:3599]#getting rid of the column headings
len(Users)

#How many interest categories were created by users?
import csv
f = open('interests.csv')
csv_f = csv.reader(f)
interests = []
for readRow in csv_f:
	interests.append(readRow[1])
from collections import Counter
from operator import itemgetter
x = Counter(interests)
z = x.most_common()
len(z)

#What is the distribution of interests in the entire population of individuals?
import numpy as np
import matplotlib.pyplot as plt
from collections import Counter
from operator import itemgetter
c = Counter(interests).items()
```

```
c.sort(key=itemgetter(1))
labels, values = zip(*c)
indexes = np.arange(len(labels))
width = 1
plt.bar(indexes, values, width)
plt.xticks(indexes + width * 0.5, labels, rotation=90)
plt.xlabel('Interest Category', fontsize=8)
plt.ylabel('Counts', fontsize=8)
plt.title('Distribution of Interest Categories across Population')
plt.show()

#How many declared interests do individual users have?
import csv
from collections import defaultdict
from collections import Counter
f = open('interests.csv')
csv_f = csv.reader(f)
dataset = list(csv_f)
user=[]
for i in range(1,len(dataset)):
        user.append(dataset[i][0])
people=Counter(user)
import pandas as pd
df = pd.DataFrame.from_dict(people, orient='index').reset_index()
df.to_csv('people.csv')
labels, values= zip(*Counter(people).items())
indexes=np.arange(len(labels))
width=1
plt.bar(indexes, values, width)
plt.xticks(indexes + width * 0.5,labels)
plt.show()
```

#Part B

#This code reads that raw dataset provided by client, calculates the jaccard index for #each pair of users based on their followings and interests and output two csv files #jaccard_follows.csv and jaccard_interest.csv

```
import numpy as np

import csv

from collections import Counter

import pandas as pd

import pprint as pp

import random as rand
```

```
follows = []
with open('follows.csv','r') as csvfile:
    reader = csv.reader(csvfile, delimiter=',')
    for row in reader:
        follows.append(row)
followsTitle = follows[0]
follows=follows[1:]
headers =['follwer_id','followee_id']
followsPD=pd.DataFrame(follows, columns=headers)

interests = []
with open('interests.csv','r') as csvfile:
    reader = csv.reader(csvfile, delimiter=',')
    for row in reader:
        interests.append(row)
interestsTitle = interests[0]
interests=interests[1:]
headers =['user_id','category']
interestsPD=pd.DataFrame(interests, columns=headers)

user1 = [int(item[0]) for item in follows]
user2 = [int(item[1]) for item in follows]
user3 = [int(item[0]) for item in interests]

userlist = list(set(user1+user2+user3))
userlist=sorted(userlist)



shape=len(userlist)
```

```
interestsByUser={}

for row in interests:
    if row[0] in interestsByUser.keys():
        lists = interestsByUser[row[0]]
        lists.append(row[1])
        interestsByUser[row[0]] = lists
    else:
        interestsByUser[row[0]] = [row[1]]

cooccurence_matrix = np.matrix(np.zeros(shape=(shape, shape)), float)
a = set(interestsByUser[str(userI)])
a
i=0
for userI in userlist:
    if str(userI) in interestsByUser.keys():
        j = 0
        for userJ in userlist:
            if str(userJ) in interestsByUser.keys():
                userIInterests = set(interestsByUser[str(userI)])
                userJInterests = set(interestsByUser[str(userJ)])

jaccardScore=float(len(userIInterests.intersection(userJInterests)))/float(len(userIInterests.union
(userJInterests)))
                cooccurence_matrix[i,j]=jaccardScore
            j+=1
    i+=1

index = [key for key in userlist]
columns=index
```

```
cooccurence_pandas=pd.DataFrame(cooccurence_matrix,index=index, columns=columns)
cooccurence_pandas.to_csv('jaccard_interest.csv')

following={}

for row in follows:
    if row[0] in following.keys():
        lists = following[row[0]]
        lists.append(row[1])
        following[row[0]] = lists
    else:
        following[row[0]] = [row[1]]


cooccurence_matrix_follow = np.matrix(np.zeros(shape=(shape, shape)), float)

i=0
for userI in userlist:
    userI = str(userI)
    if userI in following.keys():
        j = 0
        for userJ in userlist:
            userJ = str(userJ)
            if userJ in following.keys():
                userIfollows = set(following[userI])
                userJfollows = set(following[userJ])

jaccardScore_follow=float(len(userIfollows.intersection(userJfollows)))/float(len(userIfollows.unio
n(userJfollows)))
                cooccurence_matrix_follow[i,j]=jaccardScore_follow
```

```
         j+=1
    i+=1


cooccurence_pandas_follow=pd.DataFrame(cooccurence_matrix_follow,index=index,
columns=columns)

cooccurence_pandas_follow.to_csv('jaccard_follows.csv')


totalUsers=len(set(interestsPD['user_id']))


userSet=set(interestsPD['user_id'])

sampleUsers= rand.sample(userSet,100)

predictions={}

for user in sampleUsers:

    row=pd.DataFrame(cooccurence_pandas.loc[[user]],columns=columns)

    sortedRow=row.iloc[:, np.argsort(row.loc[user])[::-1]]

    predictions[user]=sortedRow.columns[1]
```

```
#This code samples user pairs for classfier training. To better train the model, we #oversampled
the data to 50% following and 50% non-following and output it to

#a csv file called balancedSample.csv

import numpy as np

import csv

from collections import Counter

import pandas as pd

import pprint as pp

import random as rand



follows = []

with open('follows.csv','r') as csvfile:

    reader = csv.reader(csvfile, delimiter=',')

    for row in reader:

        follows.append(row)

followsTitle = follows[0]

follows=follows[1:]

headers1 =['follower_id','followee_id']

followsPD=pd.DataFrame(follows, columns=headers1)


interests = []

with open('interests.csv','r') as csvfile:

    reader = csv.reader(csvfile, delimiter=',')

    for row in reader:

        interests.append(row)

interestsTitle = interests[0]

interests=interests[1:]

headers =['user_id','category']

interestsPD=pd.DataFrame(interests, columns=headers)
```

```
userSet=list(set(interestsPD['user_id']))
userBig=userSet*17
followingSet=set(followsPD['follower_id'])
followeeSet=set(followsPD['followee_id'])
user1=rand.sample(userBig, len(userBig))
user2=rand.sample(userBig, len(userBig))
randomPairs=[]
for i in range(len(user1)):
    row=[]
    row.append(user1[i])
    row.append(user2[i])
    randomPairs.append(row)

print(randomPairs)


zeros=[]
ones=[]
for i in range(len(randomPairs)):
    bothSame=0
    for j in range(len(follows)):
        if randomPairs[i][0]==follows[j][0] and randomPairs[i][1]==follows[i][1]:
            bothSame+=1
            print('same: ',randomPairs[i],follows[j])
        else:
            print('not same: ', randomPairs[i], follows[j])
    if bothSame<1:
        zeros.append(randomPairs[i])
    else:
        ones.append(randomPairs[i])
```

```
zerosPD=pd.DataFrame(zeros,columns=headers1)
zerosPD.to_csv('zeros.csv')

zeros0=[]
for row in zeros:
    row.append(0)
    zeros0.append(row)

follows1=[]
for row in follows:
    row.append(1)
    follows1.append(row)

headers2=headers1+['Target']
balancedSample=zeros0+rand.sample(follows1,len(zeros0))
balancedSamplePD=pd.DataFrame(balancedSample,columns=headers2)
balancedSamplePD.to_csv('balancedSample.csv')
```

```
#This code outputs dictionaries to show which user is following which user for future #query. It
also generate a list of most popular users in the network

import numpy as np

import csv

from collections import Counter

import pandas as pd

import pprint as pp

import random as rand

import operator

import statistics

from operator import itemgetter


follows = []

with open('follows.csv','r') as csvfile:

    reader = csv.reader(csvfile, delimiter=',')

    for row in reader:

        follows.append(row)

followsTitle = follows[0]

follows=follows[1:]

headersFoll =['follwer_id','followee_id']

followsPD=pd.DataFrame(follows, columns=headersFoll)


interests = []

with open('interests.csv','r') as csvfile:

    reader = csv.reader(csvfile, delimiter=',')

    for row in reader:

        interests.append(row)

interestsTitle = interests[0]

interests=interests[1:]

headersCat =['user_id','category']
```

```
interestsPD=pd.DataFrame(interests, columns=headersCat)

def aggregateByUser(table):
    interestsByUser={}
    for row in table:
        if row[0] in interestsByUser.keys():
            lists = interestsByUser[row[0]]
            lists.append(row[1])
            interestsByUser[row[0]] = lists
            #print(interestsByUser[row[0]])
        else:
            interestsByUser[row[0]] = [row[1]]
            #print(interestsByUser[row[0]])
    return interestsByUser

#these are the dictionaries that you can plug the userID into!
interestsByUser=aggregateByUser(interests)
followeesByUser=aggregateByUser(follows)

totalUsers=len(set(interestsPD['user_id']))
userSet=set(interestsPD['user_id'])

followers={}
for row in follows:
    if row[1] in followers.keys():
        lists = followers[row[1]]
        lists.append(row[0])
        followers[row[1]] = lists
    else:
        followers[row[1]] = [row[0]]
```

```
count=[]

for key in followers.keys():
    count.append([len(followers[key]),key])

sorted(count,reverse=True)
count[0:5]
countollowers={}
for row in follows:
    if row[1] in followers.keys():
        lists = followers[row[1]]
        lists.append(row[0])
        followers[row[1]] = lists
    else:
        followers[row[1]] = [row[0]]

count=[]

for key in followers.keys():
    count.append([len(followers[key]),key])

count=sorted(count, reverse=True, key=itemgetter(0))
celebrity=count[0:5]
```

```
#This code reads the data of jaccard index on followings and interest as
#well as the sampled user pairs to train the classifer. And then we use the
#classifier to predict 5 users for a target that he will most likely to follow
#based on the probability produced by classfier. We also set a threshold of 53%
#if the probability of the recommendation is lower than the threshold, the
#recommendation will not be made, and will recommmend the target user the most #porpular
user (i.e those who have most followers) instead
import numpy as np
import csv
from collections import Counter
import pandas as pd
import pprint as pp
import random as rand
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn import svm

#read jaccard index for follwings and convert it to a dictionary
jac_follow = []
with open('jaccard_follows.csv','r') as csvfile:
    reader = csv.reader(csvfile, delimiter=',')
    for row in reader:
        jac_follow.append(row)
headers1 = jac_follow[0][1:]
jac_follow=jac_follow[1:]

jacbyfollow={}
for row in jac_follow:
    dict={}
    for i in range(len(headers1)):
        if headers1[i] != row[0]:
            dict[headers1[i]]=float(row[i+1])
    jacbyfollow[row[0]]=dict

#read jaccard index for interests and convert it to a dictionary
jac_interest = []
with open('jaccard_interest.csv','r') as csvfile:
    reader = csv.reader(csvfile, delimiter=',')
    for row in reader:
        jac_interest.append(row)
headers2 = jac_interest[0][1:]
jac_interest=jac_interest[1:]

jacbyinterest={}

for row in jac_interest:
    dict={}
    for i in range(len(headers2)):
         if headers2[i] != row[0]:
            dict[headers1[i]]=float(row[i+1])
    jacbyinterest[row[0]]=dict

#read dataset for traing model
pairs = []
with open('balancedSample.csv','r') as csvfile:
    reader = csv.reader(csvfile, delimiter=',')
    for row in reader:
        pairs.append(row)
pairs=pairs[1:]
```

```
data = []
for row in pairs:
    row.append(jacbyfollow[row[1]][row[2]])
    row.append(jacbyinterest[row[1]][row[2]])
    data.append(row)

X = [[item[4],item[5]] for item in data]
Y = [int(item[3]) for item in data]
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,stratify=Y, test_size=0.2)

#logistic regression
clf_lr = linear_model.LogisticRegression()
clf_lr.fit(X_train, Y_train)
print (clf_lr.score(X_test, Y_test))

#neural network
clf_nn=MLPClassifier()
clf_nn.fit(X_train, Y_train)
print (clf_nn.score(X_test, Y_test))

#svm
clf_svm = svm.SVC()
clf_svm.fit(X_train, Y_train)
print (clf_svm.score(X_test, Y_test))



#recommend 5 users
no = []
recommend = {}
for target in jacbyfollow.keys():
    all_others = list(jacbyfollow[target].keys())
    XX=[]
    for other in jacbyfollow[target].keys():
        XX.append([jacbyfollow[target][other],jacbyinterest[target][other]])
    prediction=list(clf_nn.predict_proba(np.array(XX)))
    for i in range(len(prediction)):
        if prediction[i][0]<0.5:
            no.append([target,i])
    prob= [item[0] for item in prediction]
    score = [item for item in sorted(zip(prob,all_others),reverse=True)]
    recommend[target]=[]
    i=0
    while len(recommend[target])<5 and score[i][0]>0.53:
        if target in followeesByUser.keys():
            if not(score[i][1] in followeesByUser[target]):
                recommend[target].append(score[i][1])
        else:
            recommend[target].append(score[i][1])
        i=i+1

recommend_full=recommend
for key in recommend_full.keys():
    i=0
    while len(recommend_full[key])<5:
        recommend_full[key].append(celebrity[i][1])
        i=i+1
```

Resources Used

Aparicio, S., Alvarez, G. (Aug 2015). *A Model for Scale-Free Networks: Application to Twitter.* Retrieved from: www.mdpi.com/1099-4300/17/8/5848/pdf

Bao, J., Zheng, H., Wilkie. D., Mokbel, M. (2015) *Recommendations in location-based social networks: a survey.* Retrieved from: http://cel.webofknowledge.com/InboundService.do?customersID=SPRINGER&smartRedirect=yes&mode=FullRecord&IsProductCode=Yes&product=CEL&Init=Yes&Func=Frame&action=retrieve&SrcApp=PARTNER_APP&SrcAuth=SPRINGER&SID=5DyDj8ZXvGHnk8kK9be&UT=WOS%3A000353294800004

Castillo, M. (2018). *Facebook's Mark Zuckerberg: 'I'm responsible for what happened' with data privacy issues* https://www.cnbc.com/2018/04/04/mark-zuckerberg-facebook-user-privacy-issues-my-mistake.html

Chheda, R. (2016). *Analyzing Twitter Networks*. Retrieved from: (http://humnet.scripts.mit.edu/wordpress2/wp-content/uploads/2011/09/rinalc@mit.edu_.2.pdf

Linoff, G. (Sept 2009) Adjust for Oversampling. Retribed from: http://blog.data-miners.com/2009/09/adjusting-for-oversampling.html

Moreira, Gabriel. *Recommender Systems in Python 101.* Retrieved from: https://www.kaggle.com/gspmoreira/recommender-systems-in-python-101

Moyer, J. (Aug 2012). *Who are the Social Media Influencers?: A Study of Public Perceptions of Personality*. Retrieved from: https://instituteforpr.org/who-are-the-social-media-influencers-a-study-of-public-perceptions-of-personality/

No Author. (Jan 2018), *Software for Complex Networks.* Retrieved from: https://networkx.github.io/

Sardana, Divya. *RecommenderSystems_PyData_2016*. Retrieved from: https://github.com/dvysardana/RecommenderSystems_PyData_2016

Sonberg, J. (2017). *How to build a strong A/B testing plan that gets results.* Retrieved from:

https://conversionxl.com/blog/how-to-build-a-strong-ab-testing-plan-that-gets-results/