Text analytics and network based analysis solution for airline companies

A project report submitted to the Unstructured Data Management class in partial fulfilment of the class overall requirements

By
Team Conda

Team Members :
Aditya Gogoi
Bhargavi Trivedi
Kinnari Uttarwar
Mandar Oak
Namrata Sharma

Acknowledgement

We must find time to stop and thank the people who make a difference in our lives..
- John F. Kennedy

There are many people that worked hard to make this project something of real value. Even though we worked arduously, however, it is certain that we are in the debt of the following individuals:

The biggest influencer on this project was surely **Prof Ling Xue**. Prof Ling Xue is easily one the most influential person who helped us come up with the outline of this project. He has introduced us to various text analytics tools and technologies such as MongoDB, Neo4j, Py2Neo, PyMongo and Spyder.

Table of Contents

Acknowledgements 1

Table of Contents 2

Abstract: 3

Introduction: 4

Datasets...5

Limitations…………………………..6

Solution 1.a

MongoDB Data Load…………………………...7

Lexicon-based Analysis……....…………………....8

Lexicon-based Analysis Code……..………………..9

Lexicon-based Analysis Result………….……………..…13

Naive base classifier…………..………………....……………15

Naive base code …………………………………...16

Naive base result…………………………..………………21

Solution 1.b

Feature selection using SCIKIT …...……………………22

Solution 1.c

Topic Modelling ………………………………….......…28

Solution 2.a, 2.b, 2.d

Network based analysis in Neo4j....................................…31

Abstract

A dataset of real customer review is collected from Skytrax (www.airlinequality.com), a major website for customers to evaluate various airline companies. This dataset consists of 41,396 review entries and each of the entry has rating of a particular airline company by an author. The customer review data has to be stored in MongoDB database. For further analysis, PyMongo is used to access the data from MongoDB. The next step is to perform sentiment analysis on the data in two ways – lexicon-based way and a machine learning based classification (using nltk).

In the second module, we need to construct a rating network between author and the airlines company they rated. Py2Neo is used to construct the network in Neo4j. Final step is to do a deeper analysis on the airlines and compare each airline with its competitors.

Introduction

In this project, we have a dataset of real customer review collected from Skytrax (www.airlinequality.com), a major website for customers to evaluate various airline companies. This dataset consists of 41,396 review entries and each of the entry has rating of a particular airline company by an author. We have stored this data in MongoDB database. We need to use PyMongo to access the data from MongoDB. The next step is to perform sentiment analysis on the data in two ways – lexicon-based way and a machine learning based classification (using nltk).

Next, we need to construct a rating network between author and the airlines company they rated. Py2Neo is used to construct the network in Neo4j. Final step is to do a deeper analysis on the airlines and compare each airline with its competitors.

Datasets

The data that we have is a real customer review data collected from Skytrax. This dataset contains 41,396 review entries, with each entry capturing the rating of an airline company by an author. In this context, an author is also a rater as well as a customer. The data is stored in a MongoDB database. A sample document (corresponding to a review entry) is as follows:

```
{
    "authorcountry" : "Israel",
    "rating_valuemoney" : 5,
    "recommended" : 1,
    "airlinename" : "el-al-israel-airlines",
    "travellertype" : "FamilyLeisure",
    "cabin" : "Economy",
    "aircraft" : "Boeing 747-400",
    "rating_overall" : 9,
    "reviewcontent" : "Flight was half-full. I had the whole row to myself. ...",
    "rating_inflightEnt" : 4,
    "reviewdate" : "7/4/2015",
    "rating_cabinstaff" : 5,
    "route" : "TLV to JFK",
    "rating_seatcomfort" : 4,
    "rating_foodbeverage" : 5,
    "authorname" : " Moam Ben-Shalom"
}
```

The overall rating is on a scale of 1-10, with a higher level indicating more satisfaction. The ratings of individual aspects (e.g., cabin staff, seat comfort, food & beverage, etc.) are on a scale of 1-5. The property “recommended” indicates whether or not the author would like to recommend this airline to others. The dataset is in bson format.

Limitations

This dataset consists of 41,396 entries and using this dataset will consume a lot of memory which will further lower the performance of our analysis; hence, we are performed our analysis on 1000 entries.

MongoDB Data Load

1.a You need to develop a solution to assess the sentiment of authors in their reviews. Develop your solution to assess review sentiment in two ways: a lexicon-based way and a machine-learning-based classification (you can use *rating_overall* for learning);

```
mongorestore -d test2 C:\Airline
```

Lexicon-based Analysis

Lexicon based analysis is the process of converting a sequence of characters (such as in a computer program or web page) into a sequence of tokens (strings with an assigned and thus identified meaning). Roughly the equivalent of splitting ordinary text written in a natural language into a sequence of words and punctuation symbols. A program that performs lexical analysis may be termed a lexer, tokenizer, or scanner, though scanner is also a term for the first stage of a lexer. A lexer is generally combined with a parser, which together analyze the syntax of programming languages, web pages, and so forth.

Lexicon-based Analysis Code

```
# -*- coding: utf-8 -*-

import pandas as pd

# Import csv package to convert pandas dataframe to csv file
import csv

# Import Counter package to do counting
from collections import Counter

# Import operator package to sort a dictoinary by its values
import operator

# Import NTLK package after the installation
from nltk import sent_tokenize,word_tokenize
from nltk.corpus import stopwords
from nltk.stem.snowball import SnowballStemmer

import nltk
#nltk.download()

from pymongo import MongoClient

client = MongoClient()

db = client.Airline

Airline_reviews = db.Airline

result = Airline_reviews.find({})

Airlinedf =  pd.DataFrame(list(result))

del Airlinedf['_id']

Airlinedf.head(5)

# ### **Lexicon-based Word Counting**
# We use [General Inquirer](http://www.wjh.harvard.edu/~inquirer/), a free dictionary, to analyze reveiw
sentiment.
```

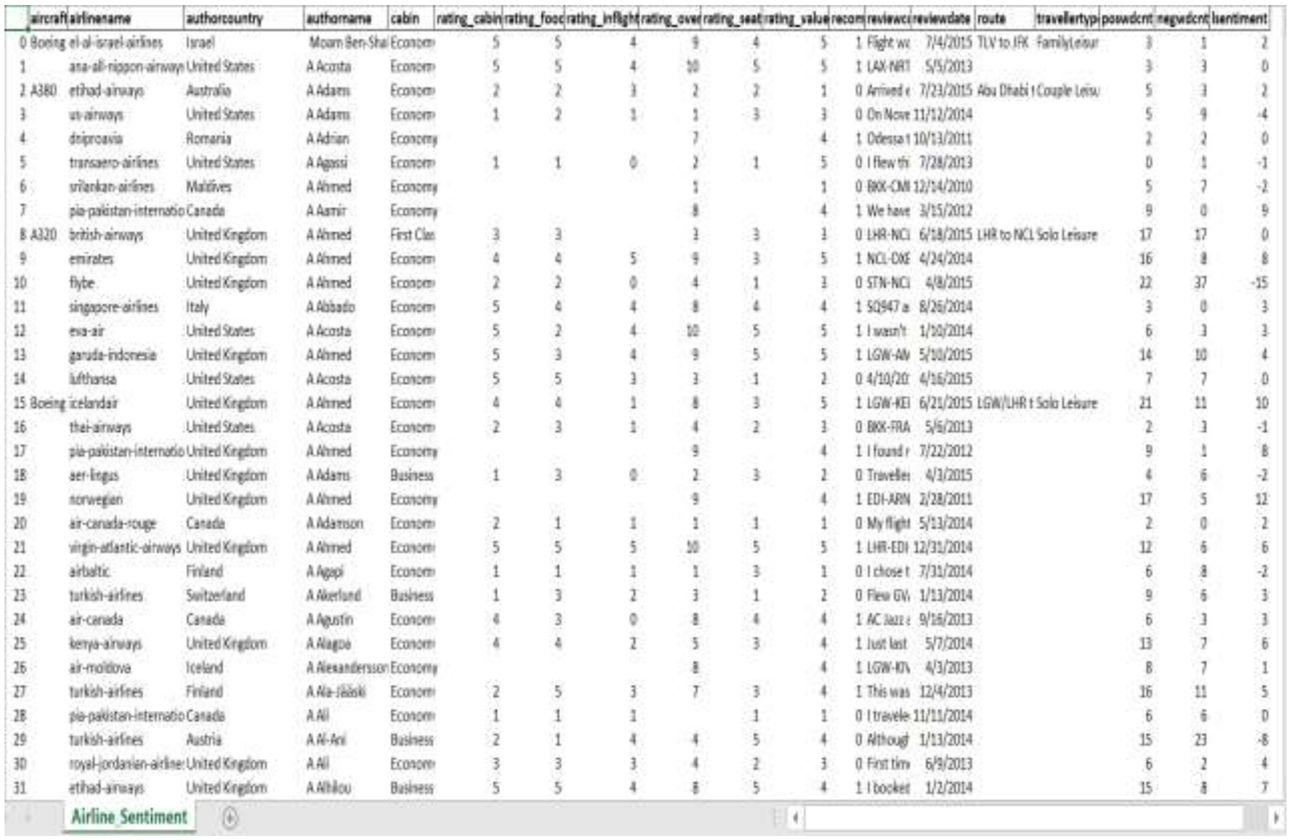
```
# **Import positive and negative words from General Inquirer Dictionary**
positive=[]
negative=[]
keys_to_ignore = ['Entry','Source','Defined']
with open('general_inquirer_dict.txt') as fin:
    reader = csv.DictReader(fin,delimiter='\t')
    for i,line in enumerate(reader):
        if line['Negativ']=='Negativ':
            if line['Entry'].find('#')==-1:
                negative.append(line['Entry'].lower())
            if line['Entry'].find('#')!=-1: #In General Inquirer, some words have multiple senses. Combine all
tags for all senses.
                negative.append(line['Entry'].lower()[:line['Entry'].index('#')])
        if line['Positiv']=='Positiv':
            if line['Entry'].find('#')==-1:
                positive.append(line['Entry'].lower())
            if line['Entry'].find('#')!=-1: #In General Inquirer, some words have multiple senses. Combine all
tags for all senses.
                positive.append(line['Entry'].lower()[:line['Entry'].index('#')])

fin.close()

# Store positive words and negative words from the dictionary in two lists
pvocabulary=sorted(list(set(positive)))
nvocabulary=sorted(list(set(negative)))

#review_original = Airlinedf[['reviewcontent','airlinename','authorname','rating_overall','recommended']]

review = Airlinedf.iloc[:1000,:]

#print(review)

review['poswdcnt']=0
review['negwdcnt']=0
review['lsentiment']=0
review_index=0

# Tokenize the words from the review documents to a word list
def getWordList(text,word_proc=lambda x:x):
    word_list=[]
    for sent in sent_tokenize(text):
        for word in word_tokenize(sent):
            word_list.append(word)
    return word_list# If needed, a dictionary can be used for stemming
stemmer = SnowballStemmer("english")

# The lists are used for storing the # of positive words, the # of negative words,
# and the overall sentiment level for all the documents
# The length of each list is equal to the total number of review document
pcount_list=[]
ncount_list=[]
lsenti_list=[]
```

```
# Iterate all review documents
# For each word, look it up in the positive word list and the negative word list
# If found in any list, update the corresponding counts
for text in review['reviewcontent']:
    vocabulary=getWordList(text,lambda x:x.lower())

    # Remove words with a length of 1
    vocabulary=[word for word in vocabulary if len(word)>1]

    # Remove stopwords
    vocabulary=[word for word in vocabulary
                if not word in stopwords.words('english')]

    # Stem words
    vocabulary=[stemmer.stem(word) for word in vocabulary]

    pcount=0
    ncount=0
    for pword in pvocabulary:
        pcount += vocabulary.count(pword)
    for nword in nvocabulary:
        ncount += vocabulary.count(nword)

    pcount_list.append(pcount)
    ncount_list.append(ncount)
    lsenti_list.append(pcount-ncount)

    #review.loc[review_index,'poswdcnt']=pcount
    #review.loc[review_index,'negwdcnt']=ncount
    #review.loc[review_index,'lsentiment']=pcount-ncount

    #review_index += 1
    #print(review_index)

# Storing word counts and overall sentiment into the dataframe
# So that we know the # of positive words, # of negative words, and sentiment
# for each review document
se=pd.Series(pcount_list)
review['poswdcnt']=se
se=pd.Series(ncount_list)
review['negwdcnt']=se
se=pd.Series(lsenti_list)
review['lsentiment']=se
#print(review)

review.to_csv('Airline_Sentiment.csv')

# Run an OLS regression
import statsmodels.formula.api as sm
result=sm.ols(formula="rating_overall~lsentiment",data=review).fit()
print(result.summary())
```

```
# Run a logistic regression
import statsmodels.api as sm2
logit=sm2.Logit(review['recommended'],review['lsentiment'])
result=logit.fit()
print(result.summary())
```

Lexicon-Based Analysis Results

Result of lexicon-based analysis in csv format



	aircraft	airlinename	authorcountry	authorname	cabin	rating_cabin	rating_food	rating_inflight	rating_over	rating_seat	rating_value	recom	reviewc	reviewdate	route	travellertyp	poswdcnt	negwdcnt	lsentiment
0	Boeing	el-al-israel-airlines	Israel	Moam Ben-Shal	Econom	5	5	4	9	4	5	1	Flight wa	7/4/2015	TLV to JFK	FamilyLeisur	3	1	2
1		ana-all-nippon-airways	United States	A Acosta	Econom	5	5	4	10	5	5	1	LAX-NRT	5/5/2013			3	3	0
2	A380	etihad-airways	Australia	A Adams	Econom	2	2	3	2	2	1	0	Arrived e	7/23/2015	Abu Dhabi t	Couple Leisu	5	3	2
3		us-airways	United States	A Adams	Econom	1	2	1	1	3	3	0	On Nove	11/12/2014			5	9	-4
4		dniproavia	Romania	A Adrian	Economy				7		4	1	Odessa t	10/13/2011			2	2	0
5		transaero-airlines	United States	A Agassi	Econom	1	1	0	2	1	5	0	I flew thi	7/28/2013			0	1	-1
6		srilankan-airlines	Maldives	A Ahmed	Economy				1		1	0	BKK-CMI	12/14/2010			5	7	-2
7		pia-pakistan-internatio	Canada	A Aamir	Economy				8		4	1	We have	3/15/2012			9	0	9
8	A320	british-airways	United Kingdom	A Ahmed	First Clas	3	3		3	3	3	0	LHR-NCL	6/18/2015	LHR to NCL	Solo Leisure	17	17	0
9		emirates	United Kingdom	A Ahmed	Econom	4	4	5	9	3	5	1	NCL-DXE	4/24/2014			16	8	8
10		flybe	United Kingdom	A Ahmed	Econom	2	2	0	4	1	3	0	STN-NCL	4/8/2015			22	37	-15
11		singapore-airlines	Italy	A Abbado	Econom	5	4	4	8	4	4	1	SQ947 a	8/26/2014			3	0	3
12		eva-air	United States	A Acosta	Econom	5	2	4	10	5	5	1	I wasn't	1/10/2014			6	3	3
13		garuda-indonesia	United Kingdom	A Ahmed	Econom	5	3	4	9	5	5	1	LGW-AM	5/10/2015			14	10	4
14		lufthansa	United States	A Acosta	Econom	5	5	3	3	1	2	0	4/10/20	4/16/2015			7	7	0
15	Boeing	icelandair	United Kingdom	A Ahmed	Econom	4	4	1	8	3	5	1	LGW-KEI	6/21/2015	LGW/LHR t	Solo Leisure	21	11	10
16		thai-airways	United States	A Acosta	Econom	2	3	1	4	2	3	0	BKK-FRA	5/6/2013			2	3	-1
17		pia-pakistan-internatio	United Kingdom	A Ahmed	Economy				9		4	1	I found r	7/22/2012			9	1	8
18		aer-lingus	United Kingdom	A Adams	Business	1	3	0	2	3	2	0	Travelle	4/3/2015			4	6	-2
19		norwegian	United Kingdom	A Ahmed	Economy				9		4	1	EDI-ARN	2/28/2011			17	5	12
20		air-canada-rouge	Canada	A Adamson	Econom	2	1	1	1	1	1	0	My flight	5/13/2014			2	0	2
21		virgin-atlantic-airways	United Kingdom	A Ahmed	Econom	5	5	5	10	5	5	1	LHR-EDI	12/31/2014			12	6	6
22		airbaltic	Finland	A Agapi	Econom	1	1	1	1	3	1	0	I chose t	7/31/2014			6	8	-2
23		turkish-airlines	Switzerland	A Akerlund	Business	1	3	2	3	1	2	0	Flew GV	1/13/2014			9	6	3
24		air-canada	Canada	A Agustin	Econom	4	3	0	8	4	4	1	AC Jazz	9/16/2013			6	3	3
25		kenya-airways	United Kingdom	A Alagoa	Econom	4	4	2	5	3	4	1	Just last	5/7/2014			13	7	6
26		air-moldova	Iceland	A Alexandersson	Economy				8		4	1	LGW-KIV	4/3/2013			8	7	1
27		turkish-airlines	Finland	A Ala-Jääski	Econom	2	5	3	7	3	4	1	This was	12/4/2013			16	11	5
28		pia-pakistan-internatio	Canada	A Ali	Econom	1	1	1		1	1	0	I travele	11/11/2014			6	6	0
29		turkish-airlines	Austria	A Al-Ani	Business	2	1	4	4	5	4	0	Althoug	1/13/2014			15	23	-8
30		royal-jordanian-airline	United Kingdom	A Ali	Econom	3	3	3	4	2	3	0	First tim	6/9/2013			6	2	4
31		etihad-airways	United Kingdom	A Alhilou	Business	5	5	4	8	5	4	1	I booked	1/2/2014			15	8	7

Airline_Sentiment

Lexicon -based analysis on FULL DATASET

```
                            OLS Regression Results
==============================================================================
Dep. Variable:          rating_overall   R-squared:                       0.169
Model:                             OLS   Adj. R-squared:                  0.169
Method:                  Least Squares   F-statistic:                     7509.
Date:                 Tue, 21 Feb 2017   Prob (F-statistic):               0.00
Time:                         21:40:22   Log-Likelihood:                -91929.
No. Observations:                36861   AIC:                         1.839e+05
Df Residuals:                    36859   BIC:                         1.839e+05
Df Model:                            1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [95.0% Conf. Int.]
------------------------------------------------------------------------------
Intercept      5.2146      0.018    289.911      0.000         5.179     5.250
lsentiment     0.3240      0.004     86.655      0.000         0.317     0.331
==============================================================================
Omnibus:                      6020.864   Durbin-Watson:                   1.966
Prob(Omnibus):                   0.000   Jarque-Bera (JB):             1913.044
Skew:                           -0.321   Prob(JB):                         0.00
Kurtosis:                        2.086   Cond. No.                         5.74
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
Optimization terminated successfully.
         Current function value: 0.616695
         Iterations 5
                           Logit Regression Results
==============================================================================
Dep. Variable:             recommended   No. Observations:                41396
Model:                           Logit   Df Residuals:                    41395
Method:                            MLE   Df Model:                            0
Date:                 Tue, 21 Feb 2017   Pseudo R-squ.:                  0.1073
Time:                         21:40:23   Log-Likelihood:                -25529.
converged:                        True   LL-Null:                       -28599.
                                         LLR p-value:                       nan
==============================================================================
                 coef    std err          z      P>|z|      [95.0% Conf. Int.]
------------------------------------------------------------------------------
lsentiment     0.1915      0.003     70.397      0.000         0.186     0.197
==============================================================================
```

Naive Base Classifier

In machine learning, naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

Naive Bayes has been studied extensively since the 1950s. It was introduced under a different name into the text retrieval community in the early 1960s, and remains a popular (baseline) method for text categorization, the problem of judging documents as belonging to one category or the other (such as spam or legitimate, sports or politics, etc.) with word frequencies as the features. With appropriate pre-processing, it is competitive in this domain with more advanced methods including support vector machines. It also finds application in automatic medical diagnosis.

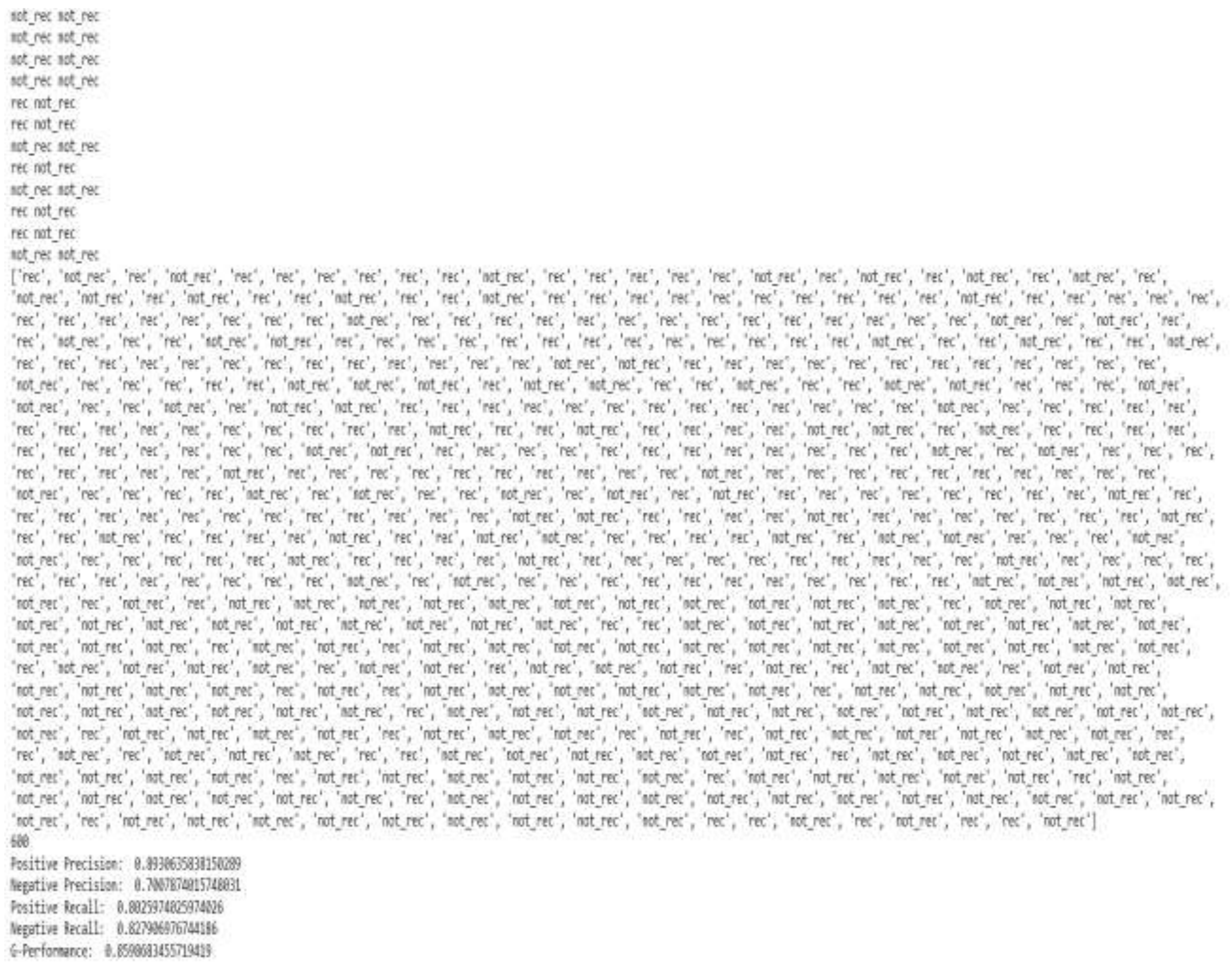
Naive Base Code

```
import pandas as pd
#mongorestore --db Airline --collecton Airline reviews.bson
# Import csv package to convert pandas dataframe to csv file
import csv

# Import Counter package to do counting
from collections import Counter

# Import operator package to sort a dictoinary by its values
import operator

# Import NTLK package after the installation
from nltk import sent_tokenize,word_tokenize
from nltk.corpus import stopwords
from nltk.stem.snowball import SnowballStemmer
from nltk.util import ngrams

import nltk
#nltk.download()

from pymongo import MongoClient

client = MongoClient()

db = client.Airline

Airline_reviews = db.Airline

result = Airline_reviews.find({})

Airlinedf =  pd.DataFrame(list(result))

del Airlinedf['_id']

def word_feats(words):
    return dict([(word,True) for word in words])

# Store positive words and negative words from the dictionary in two lists
#pvocabulary=sorted(list(set(positive)))
#nvocabulary=sorted(list(set(negative)))

review_test = Airlinedf[['reviewcontent','airlinename','authorname','rating_overall','recommended']]

#review = review_original.iloc[:1000,:]
```

```
#print(review)

recc_feat=[]
not_recc_feat= []

review = review_test.iloc[:2000,:]

#print(review)

import re

for row in range(0,len(review)):
    #print(review.iloc[row]['recommended'])

    # Reset the review variable
    review1=""

    if review.iloc[row]['recommended'] == 1:


    # Reset the review variable
        review1 = review.iloc[row]['reviewcontent']

    # Remove punctuation
        #review1 = re.sub(r'[^\w\s]0123456789.-,()','',review1)
        review1=re.sub(r'[^\w\s]','',review1)
    # Review words with only 1 letter
        review1=" ".join(
                        word for word in review1.split()
                        if len(word)>1
                       )

    # Remove Stopwords
        review1=" ".join(
                        word for word in review1.split()
                        if not word in stopwords.words('english')
                        )

    # Stemming
        stemmer=SnowballStemmer('english')
        review1=" ".join(
                        [stemmer.stem(word)
                        for word in review1.split()]
                       )

        ngramsize=2

        if ngramsize > 1:
            review1=[word for word in ngrams(
                                    review1.split(),ngramsize)]
            review1=(word_feats(review1),'rec')
        else:
            review1=(word_feats(review1.split()),'rec')
        recc_feat.append(list(review1))

#print(recc_feat)
for row in range(0,len(review)):
    #print(review.iloc[row]['recommended'])

    # Reset the review variable
    review2=""

    if review.iloc[row]['recommended'] == 0:

    # Reset the review variable
        review2 = review.iloc[row]['reviewcontent']

    # Remove punctuation
        #review2 = re.sub(r'[^\w\s]0123456789.-,()','',review2)
        review2=re.sub(r'[^\w\s]','',review2)
    # Review words with only 1 letter
        review2=" ".join(
                    word for word in review2.split()
                    if len(word)>1
                   )

    # Remove Stopwords
        review2=" ".join(
                    word for word in review2.split()
                    if not word in stopwords.words('english')
                    )

    # Stemming
        stemmer=SnowballStemmer('english')
        review2=" ".join(
                    [stemmer.stem(word)
                    for word in review2.split()]
                   )

        ngramsize=2
        if ngramsize>1:
            review2=[word for word in ngrams(
                                    review2.split(),ngramsize)]
            review2=(word_feats(review2),'not_rec')
        else:
            review2=(word_feats(review2.split()),'not_rec')
        not_recc_feat.append(list(review2))


print(recc_feat)
print(not_recc_feat)



train1=recc_feat[:700]+not_recc_feat[:700]
test1=recc_feat[700:]+not_recc_feat[700:]

print(len(train1))
print(len(test1))

print(train1)
print(test1)

# Train a Naive Bayesian Classifier
from nltk.classify import NaiveBayesClassifier
classifier=NaiveBayesClassifier.train(train1)

#==============================================================================
# print(classifier)
#==============================================================================
# refsets will include the original pos/neg classification from the original data
# testsets will include the predicted classification results using the trained
# Naive Bayesian Classifier

import collections

refsets=collections.defaultdict(set)
testsets=collections.defaultdict(set)

print(refsets['rec'])
print(testsets['not_rec'])
print(refsets['rec'])
print(testsets['not_rec'])



# Iterate the 20 reviews in the test dataset
# For each review, record the original class in the refset
# Put the predicted class in the testset

result=[]

for i,(feats,label) in enumerate(test1):
    refsets[label].add(i)
    observed = classifier.classify(feats)
    testsets[observed].add(i)
    result.append(observed)
    print(observed,label)

print(result)

print(len(result))
# Performance evaluation

import nltk.metrics
import math
from nltk.metrics import precision
from nltk.metrics import recall


pos_pre=precision(refsets['rec'],testsets['rec'])
neg_pre=precision(refsets['not_rec'],testsets['not_rec'])
pos_rec=recall(refsets['rec'],testsets['rec'])
neg_rec=recall(refsets['not_rec'],testsets['not_rec'])
gperf=math.sqrt(pos_pre*neg_rec)


print('Positive Precision: ',pos_pre)
print('Negative Precision: ',neg_pre)
print('Positive Recall: ',pos_rec)
print('Negative Recall: ',neg_rec)
print('G-Performance: ',gperf)
```

Naive Base Results

We can see the original and predicted result below. We have calculated the value for positive precision, negative precision, positive recall and negative recall. We got the overall performance of 85%.



```
not_rec not_rec
not_rec not_rec
not_rec not_rec
not_rec not_rec
rec not_rec
rec not_rec
not_rec not_rec
rec not_rec
not_rec not_rec
rec not_rec
rec not_rec
not_rec not_rec
['rec', 'not_rec', 'rec', 'not_rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'not_rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'not_rec', 'rec', 'not_rec', 'rec', 'not_rec', 'rec', 'not_rec', 'rec',
'not_rec', 'not_rec', 'rec', 'not_rec', 'rec', 'rec', 'not_rec', 'rec', 'rec', 'not_rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'not_rec', 'rec', 'rec', 'rec', 'rec', 'rec',
'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'not_rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'not_rec', 'rec', 'not_rec', 'rec',
'rec', 'not_rec', 'rec', 'rec', 'not_rec', 'not_rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'not_rec', 'rec', 'rec', 'not_rec', 'rec', 'rec', 'not_rec',
'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'not_rec', 'not_rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec',
'not_rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'not_rec', 'not_rec', 'not_rec', 'rec', 'not_rec', 'not_rec', 'rec', 'rec', 'not_rec', 'rec', 'rec', 'not_rec', 'not_rec', 'rec', 'rec', 'rec', 'not_rec',
'not_rec', 'rec', 'rec', 'not_rec', 'rec', 'not_rec', 'not_rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'not_rec', 'rec', 'rec', 'rec', 'rec', 'rec',
'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'not_rec', 'rec', 'rec', 'not_rec', 'rec', 'rec', 'rec', 'rec', 'not_rec', 'not_rec', 'rec', 'not_rec', 'rec', 'rec', 'rec', 'rec',
'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'not_rec', 'not_rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'not_rec', 'rec', 'not_rec', 'rec', 'rec', 'rec',
'rec', 'rec', 'rec', 'rec', 'rec', 'not_rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'not_rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec',
'not_rec', 'rec', 'rec', 'rec', 'rec', 'not_rec', 'rec', 'not_rec', 'rec', 'rec', 'not_rec', 'rec', 'not_rec', 'rec', 'not_rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'not_rec', 'rec',
'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'not_rec', 'not_rec', 'rec', 'rec', 'rec', 'rec', 'not_rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'not_rec',
'rec', 'rec', 'not_rec', 'rec', 'rec', 'rec', 'rec', 'not_rec', 'rec', 'rec', 'not_rec', 'not_rec', 'rec', 'rec', 'rec', 'rec', 'not_rec', 'rec', 'not_rec', 'not_rec', 'rec', 'rec', 'rec', 'not_rec',
'not_rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'not_rec', 'rec', 'rec', 'rec', 'rec', 'not_rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'not_rec', 'rec', 'rec', 'rec', 'rec',
'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'not_rec', 'rec', 'not_rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec',
'not_rec', 'rec', 'not_rec', 'rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'rec', 'not_rec', 'not_rec', 'not_rec',
'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'rec', 'rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec',
'not_rec', 'not_rec', 'not_rec', 'rec', 'not_rec', 'not_rec', 'rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec',
'rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'rec', 'not_rec', 'not_rec', 'rec', 'not_rec', 'not_rec', 'not_rec', 'rec', 'not_rec', 'rec', 'not_rec', 'not_rec', 'rec', 'not_rec', 'not_rec',
'not_rec', 'not_rec', 'not_rec', 'not_rec', 'rec', 'not_rec', 'rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec',
'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec',
'not_rec', 'rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'rec', 'not_rec', 'not_rec', 'not_rec', 'rec', 'not_rec', 'rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'rec',
'rec', 'not_rec', 'rec', 'not_rec', 'not_rec', 'not_rec', 'rec', 'rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec',
'not_rec', 'not_rec', 'not_rec', 'not_rec', 'rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'rec', 'not_rec',
'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec',
'not_rec', 'rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'not_rec', 'rec', 'rec', 'not_rec', 'rec', 'not_rec', 'rec', 'rec', 'not_rec']
600
Positive Precision:  0.8930635838150289
Negative Precision:  0.7007874015748031
Positive Recall:  0.8025974025974026
Negative Recall:  0.827906976744186
G-Performance:  0.8590683455719419
```

Feature selection using SCIKIT

1.b We need to develop a model to explain what factors influence authors' likelihood to recommend an airline. Your model should include the information retrieved from review content. You can use your own judgement to decide which text-related factors, alongside with other available variables from the data, to include in the model, and you need to justify them in your project reports;

To explain what factors, influence authors' likelihood to recommend an airline, we cab use feature selection concept of machine learning.

Feature Selection is a method in which we select those features in data that are most useful or most relevant for the problem we are working on.

In this case, we are trying to find the influence of various types of ratings given by the author on the recommendation he/she gives.

For our analysis we are looking at the influence of -
'rating_cabinstaff','rating_foodbeverage','rating_inflightEnt','rating_seatcomfort','rating_valuemoney' and 'rating_overall' on 'recommended' column

The first step is to upload the Airline data to python. For simplicity, we are using only 1000 records. Below is the code for the same

```
 #mongorestore --db Airline --collecton Airline reviews.bson

# import pymongo to get the mongodb data into python
from pymongo import MongoClient

client = MongoClient()
db = client.Airline
Airline_reviews = db.Airline
result = Airline_reviews.find({})
Airlinedf =  pd.DataFrame(list(result))

#taking only the inteded rows from the airlines data and slcing it to 1000 records
df1 = Airlinedf[['rating_cabinstaff','rating_foodbeverage','rating_inflightEnt','rating_seatcomfort','
rating_valuemoney','rating_overall','recommended']]

df1=df1.iloc[:1000,:]
df1.fillna(0,inplace=True)
```

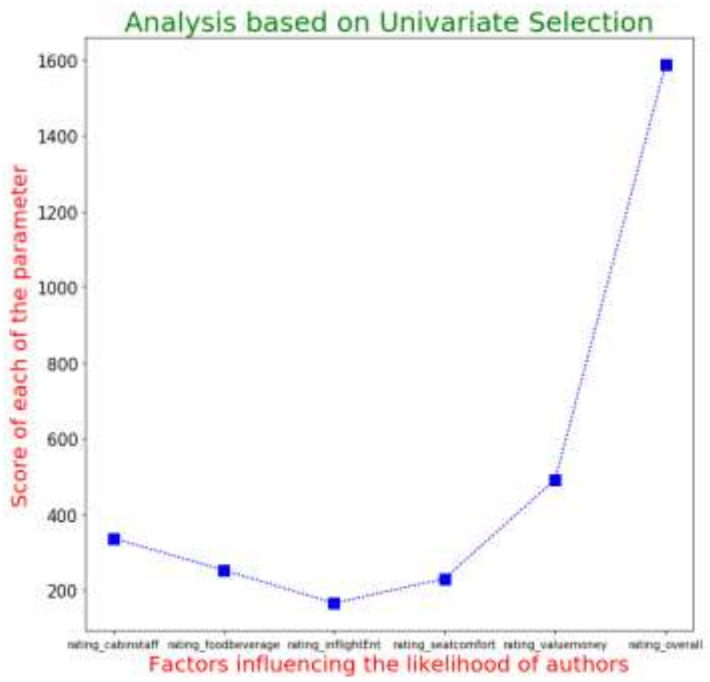
Once the data is loaded, we can use below are the 3 methods of feature Selection using scikit which shows the influence of various factors on recommendation.

1. **Univariate Selection**
   Univariate feature selection examines each feature individually to determine the strength of the relationship of the element with the response variable.

The scikit-learn library provides the SelectKBest class that can be used with a suite of different statistical tests to select a specific number of features.
Below is the code to do the same:

```
import pandas as pd
import numpy
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression

#converting dataframe created above into array
array = df1.values
print(array)
# taking X as the 6 factors
X = array[:,0:6]
# taking y as the prediction variable whiche here is 'recommendation'
Y = array[:,6]

#1st methond of Feature selection
# Univariated Selection

# feature extraction
test = SelectKBest(score_func=chi2, k=4)
fit = test.fit(X, Y)
# summarize scores
numpy.set_printoptions(precision=3)

#ploting a graph in order to represent the factors influencing recommendation
scores=numpy.array(fit.scores_)
plt.rc('font', size=20)          # controls default text sizes
plt.rc('axes', labelsize=20)    # fontsize of the x and y labels
plt.rc('xtick', labelsize=10)    # fontsize of the tick labels
plt.rc('ytick', labelsize=15)
plt.figure(figsize=(10,10))     #control size of the frame
x_tiks=['rating_cabinstaff','rating_foodbeverage','rating_inflightEnt','rating_seatcomfort',
'rating_valuemoney','rating_overall']
plt.xticks(range(6),x_tiks) #
plt.plot(scores,':s',markersize=10,color='b')
plt.ylabel('Score of each of the parameter',color='red')
plt.xlabel('Factors influencing the likelihood of authors',color='red')
plt.title('Analysis based on Univariate Selection',fontsize=25,color='green')
plt.show()
```

The graph plot looks like:



Looking at the graph, we can say that based on Univariate Selection, ‘rating_overall’ is the most influential factor followed by ‘rating_valuemoney’ and ‘rating_cabinstaff’ for the airline to be ‘recommended .'

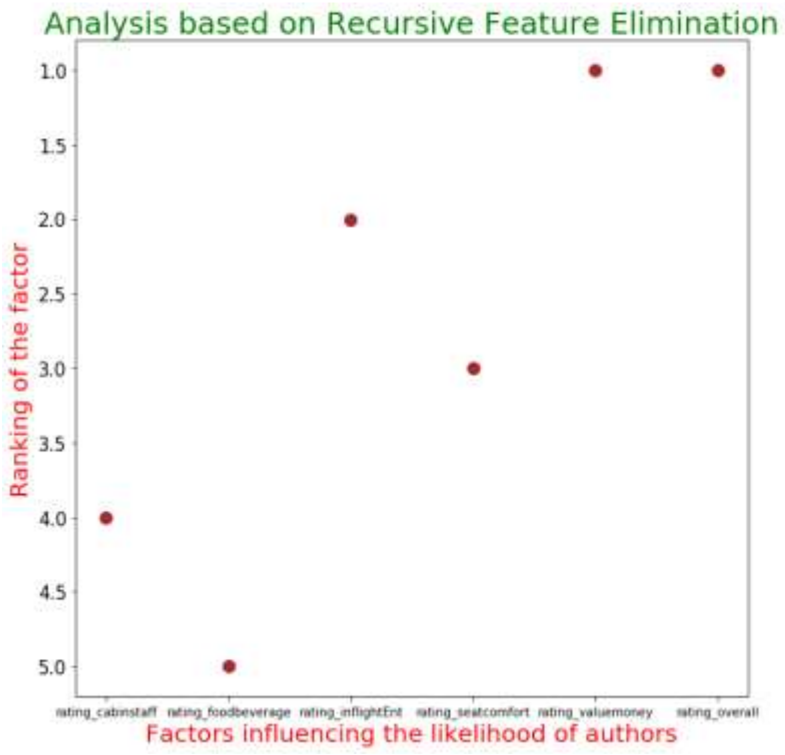
2. **Recursive Feature Elimination**
   The Recursive Feature Elimination (or RFE) works by recursively removing attributes and building a model on those attributes that remain.
   It uses the model accuracy to identify which attributes (and combination of attributes) contribute the most to predicting the target attribute.
   Below is the code to do the same:

```
import pandas as pd
import numpy
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression

#converting dataframe created above into array
array = df1.values
print(array)
# taking X as the 6 factors
X = array[:,0:6]
# taking y as the prediction variable whiche here is 'recommendation'
Y = array[:,6]
#2nd method of Feature Selection
#Recursive Feature Elimination

# feature extraction
model = LogisticRegression()
rfe = RFE(model, 2)
fit = rfe.fit(X, Y)
print("Feature Ranking: ",fit.ranking_)

#ploting a graph in order to represent the factors influencing recommendation
ranking=numpy.array(fit.ranking_)
```

```
        plt.rc('font', size=20)          # controls default text sizes
        plt.rc('axes', labelsize=20)    # fontsize of the x and y labels
        plt.rc('xtick', labelsize=10)    # fontsize of the tick labels
        plt.rc('ytick', labelsize=15)
        plt.figure(figsize=(10,10))     #control size of the frame
        x_tiks=['rating_cabinstaff','rating_foodbeverage','rating_inflightEnt',
        'rating_seatcomfort','rating_valuemoney','rating_overall']
        plt.xticks(range(6),x_tiks) #
        plt.plot(ranking,'ro',markersize=10,color='brown')
        plt.ylabel('Ranking of the factor',color='red')
        plt.xlabel('Factors influencing the likelihood of authors',color='red')
        plt.title('Analysis based on Recursive Feature Elimination',fontsize=25,color='green')
        ax = plt.gca()
        ax.invert_yaxis()
plt.show()
```

The graph plot looks like:

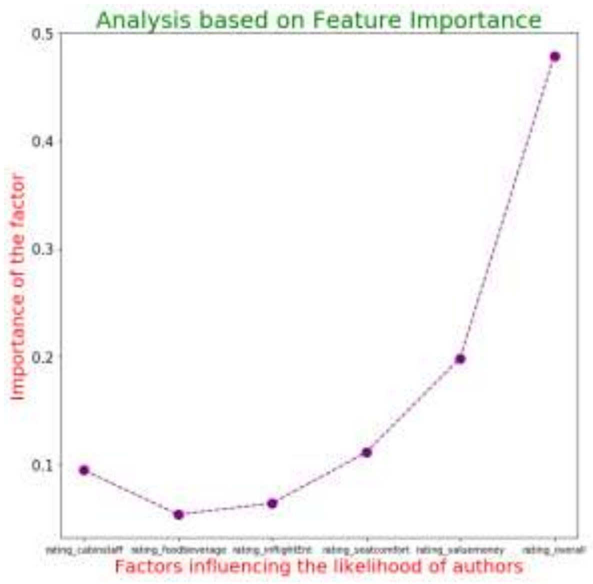
Looking at the graph we can deduce that based on Recursive Feature Elimination, most influential factor for ‘recommended’ is ‘rating_overall’ and ‘rating_valuemoney’. And the 2nd ranked factor is ‘rating_inflightEnt’



3. **Feature Importance**
   Bagged decision trees like Random Forest and Extra Trees can be used to estimate the importance of features.
   Here we are constructing, ExtraTreesClassifier classifier .
   Below is the code for the same:

```
        #3rd method of Feature Selection
        #Feature Importance
        import pandas as pd
        import numpy
        from sklearn.ensemble import ExtraTreesClassifier

        #converting dataframe created above into array
        array = df1.values
        print(array)
        # taking X as the 6 factors
        X = array[:,0:6]
        # taking y as the prediction variable whiche here is 'recommendation'
        Y = array[:,6]


        model = ExtraTreesClassifier()
        model.fit(X, Y)
        print(model.feature_importances_)

        #ploting a graph in order to represent the factors influencing recommendation
        plt.rc('font', size=20)          # controls default text sizes
        plt.rc('axes', labelsize=20)    # fontsize of the x and y labels
        plt.rc('xtick', labelsize=10)    # fontsize of the tick labels
        plt.rc('ytick', labelsize=15)
        plt.figure(figsize=(10,10))     #control size of the frame
        x_tiks=['rating_cabinstaff','rating_foodbeverage','rating_inflightEnt',
        'rating_seatcomfort','rating_valuemoney','rating_overall']
        plt.xticks(range(6),x_tiks) #
        plt.plot(importance,'--bo',markersize=10,color='purple')
        plt.ylabel('Importance of the factor',color='red')
        plt.xlabel('Factors influencing the likelihood of authors',color='red')
        plt.title('Analysis based on Feature Importance',fontsize=25,color='green')
plt.show()
```

The graph plot looks like:



Looking at the graph we can say that based on Feature Importance, ‘rating_overall’ is the most influential factor follwed by ‘rating_valuemoney’ and ‘rating_seatcomfort’ for the airline to be ‘recommended ‘.

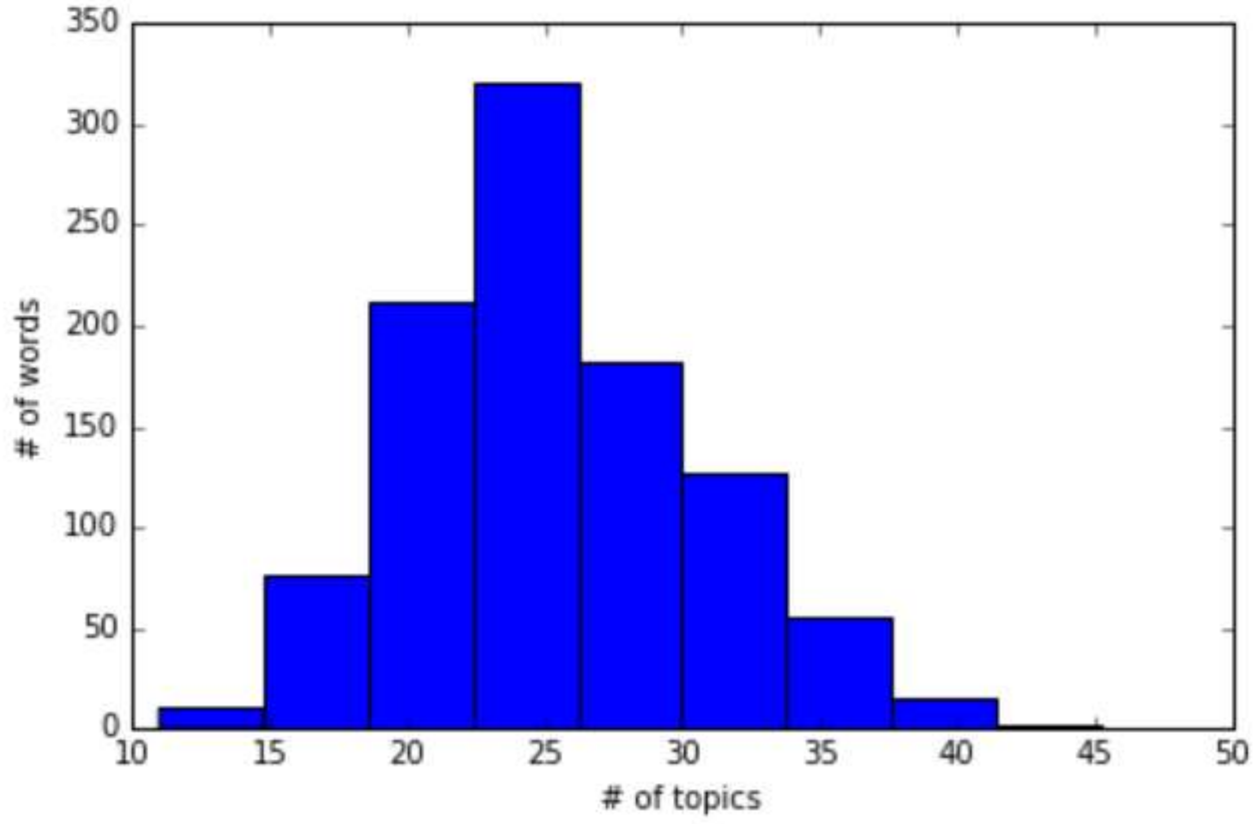
Looking at the result of all the 3 analysis we can say that ‘rating_overall’ ,’rating_cabinstaff’,’rating_inflightEnt’ and ‘rating_seatcomfort’ are the 4 major factors influencing the author to recommend any airline

Topic Modelling

1.c (Optional) You may consider enriching your above analysis (in part b) by extracting the topics from review content and using some topic measures (e.g., the number of topics and topic loadings).

```
# -*- coding: utf-8 -*-

import pandas as pd
# Import csv package to convert pandas dataframe to csv file
import csv
import matplotlib.pyplot as plt
import numpy as np


from pymongo import MongoClient

client = MongoClient()

db = client.Airline

Airline_reviews = db.Airline

result = Airline_reviews.find({})

Airlinedf =  pd.DataFrame(list(result))

del Airlinedf['_id']

review = Airlinedf.iloc[:1000,:]

topic_doc = review['reviewcontent'].tolist()


from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer
import string
stop = set(stopwords.words('english'))
exclude = set(string.punctuation)
lemma = WordNetLemmatizer()
def clean(doc):
    stop_free = " ".join([i for i in doc.lower().split() if i not in stop])
    punc_free = ''.join(ch for ch in stop_free if ch not in exclude)
    normalized = " ".join(lemma.lemmatize(word) for word in punc_free.split())
    return normalized

doc_clean = [clean(doc).split() for doc in topic_doc]
```

```
# Importing Gensim
import gensim
from gensim import corpora, models, matutils

# Creating the term dictionary of our courpus, where every unique term is assigned an index.
dictionary = corpora.Dictionary(doc_clean)

print(dictionary)

# Converting list of documents (corpus) into Document Term Matrix using dictionary prepared above.
doc_term_matrix = [dictionary.doc2bow(doc) for doc in doc_clean]

print(doc_term_matrix)


# Creating the object for LDA model using gensim library
Lda = gensim.models.ldamodel.LdaModel

# Running and Trainign LDA model on the document term matrix.
ldamodel = Lda(doc_term_matrix, num_topics=100, id2word = dictionary, alpha = 1.0)

# Find the most import topic
topics=matutils.corpus2dense(ldamodel[doc_term_matrix],
                        num_terms = ldamodel.num_topics)
weight=topics.sum(1)
max_topic=weight.argmax()

words=ldamodel.show_topic(max_topic,64)
print(words)

# Plot a histogram to display the distribution of topics by documents
num_topics_used= [len(ldamodel[doc]) for doc in doc_term_matrix]

fig,ax = plt.subplots()
ax.hist(num_topics_used)
ax.set_ylabel('# of words')
ax.set_xlabel('# of topics')
fig.tight_layout()
fig.savefig('Figure_Topics_By_Words.png')
```



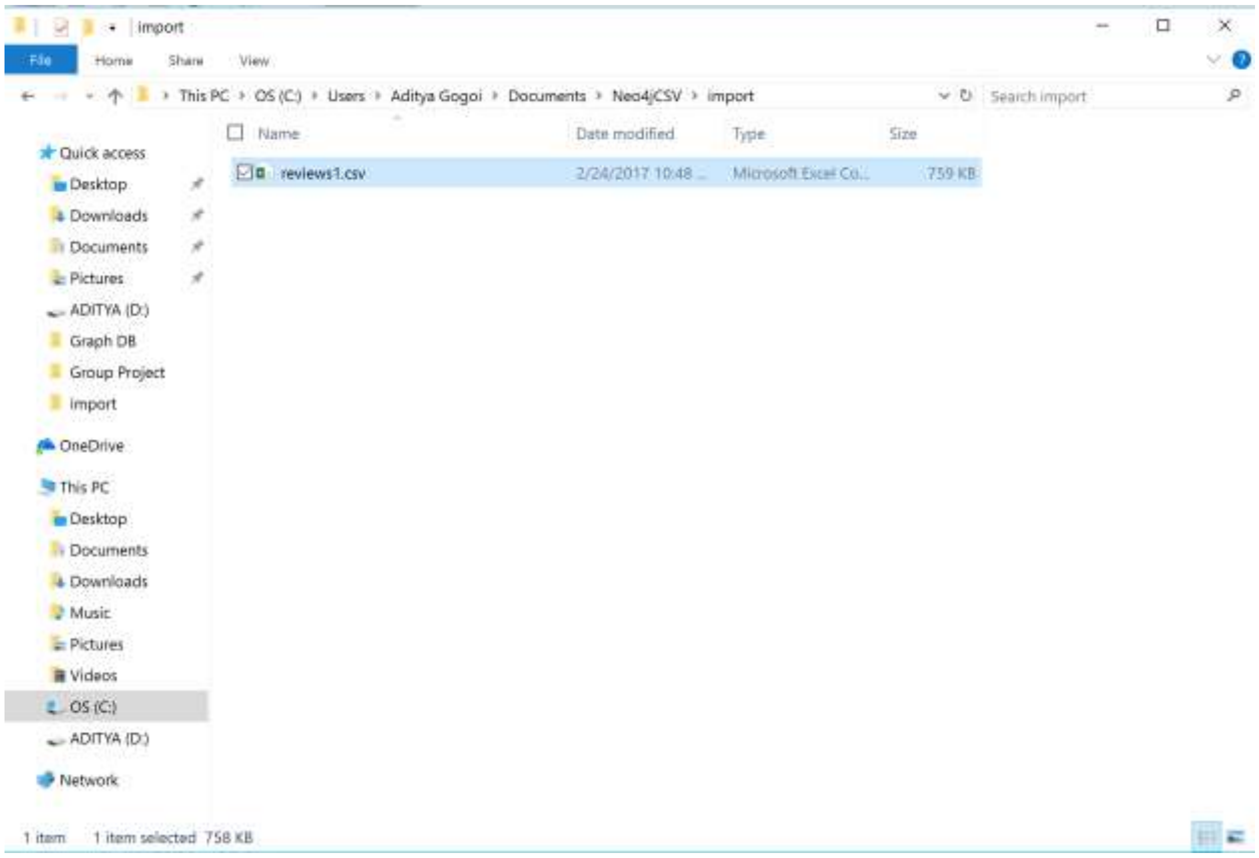
# of words
# of topics

Network based analysis in Neo4j

2.a Second, we constructed a rating network between author and the airline companies they rated. Such rating network captures who rated which airline companies and can be used to realize the network-based analyses:

1. **Code to construct this network in Neo4j;**

→ This is the first step. We have to keep the reviews1 CSV file in the target directory.

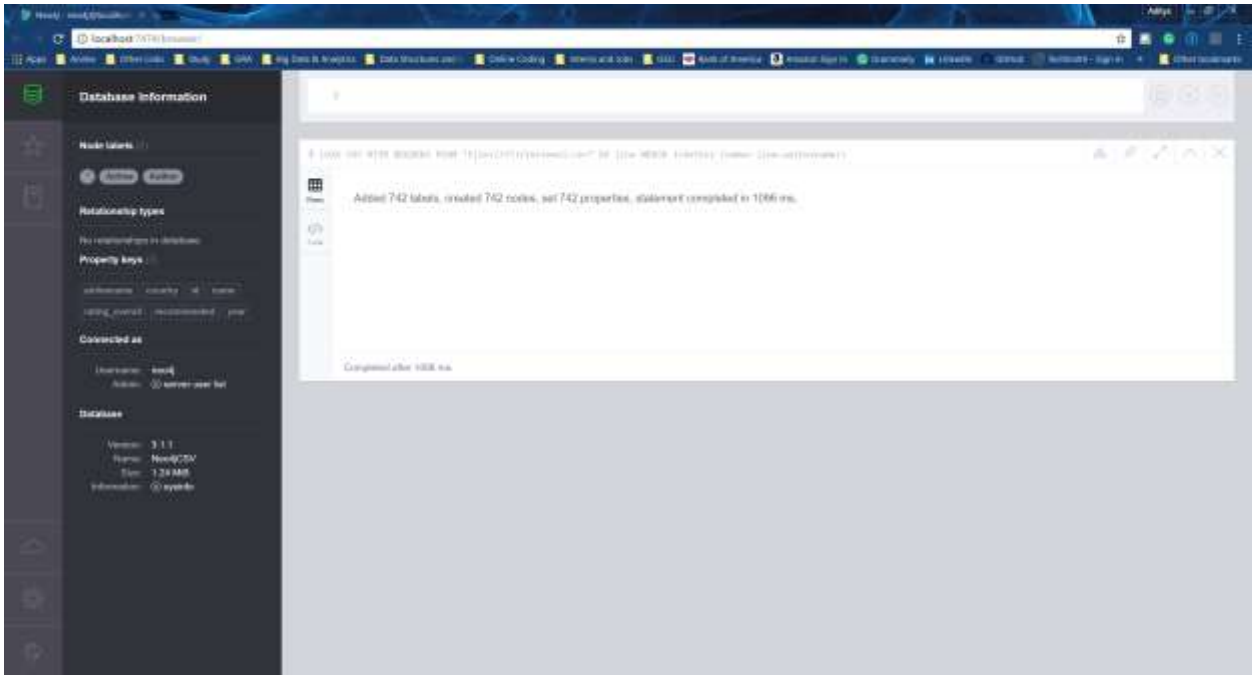


Name	Date modified	Type	Size
reviews1.csv	2/24/2017 10:48 ...	Microsoft Excel Co...	759 KB

1 item 1 item selected 758 KB

1. Loading Author Nodes:-

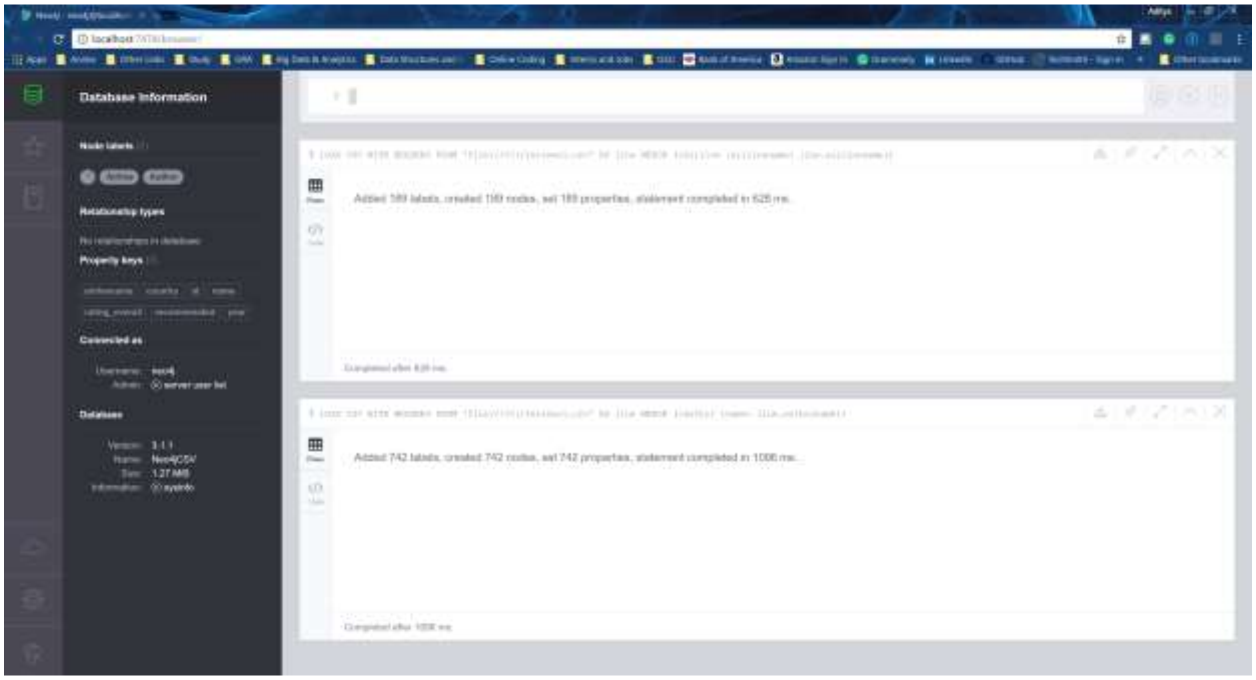
```
Query:
LOAD CSV WITH HEADERS FROM 'file:///C:/reviews1.csv' AS line
MERGE (:Author {name: line.authorname})
```



```
Added 742 labels, created 742 nodes, set 742 properties, statement completed in 1066 ms.
```

2. Loading Airline Nodes:-

```
Query:
LOAD CSV WITH HEADERS FROM 'file:///C:/reviews1.csv' AS line
MERGE (:Airline {airlinename: line.airlinename})
```



Database Information

Node labels

Relationship types

No relationships in database

Property keys

Connected as

Username: neo4j

Admin: server user list

Database

Version: 3.1.1

Name: Neo4jCSV

Size: 1.27 MiB

Information: sysinfo

```
Added 189 labels, created 189 nodes, set 189 properties, statement completed in 628 ms.
```

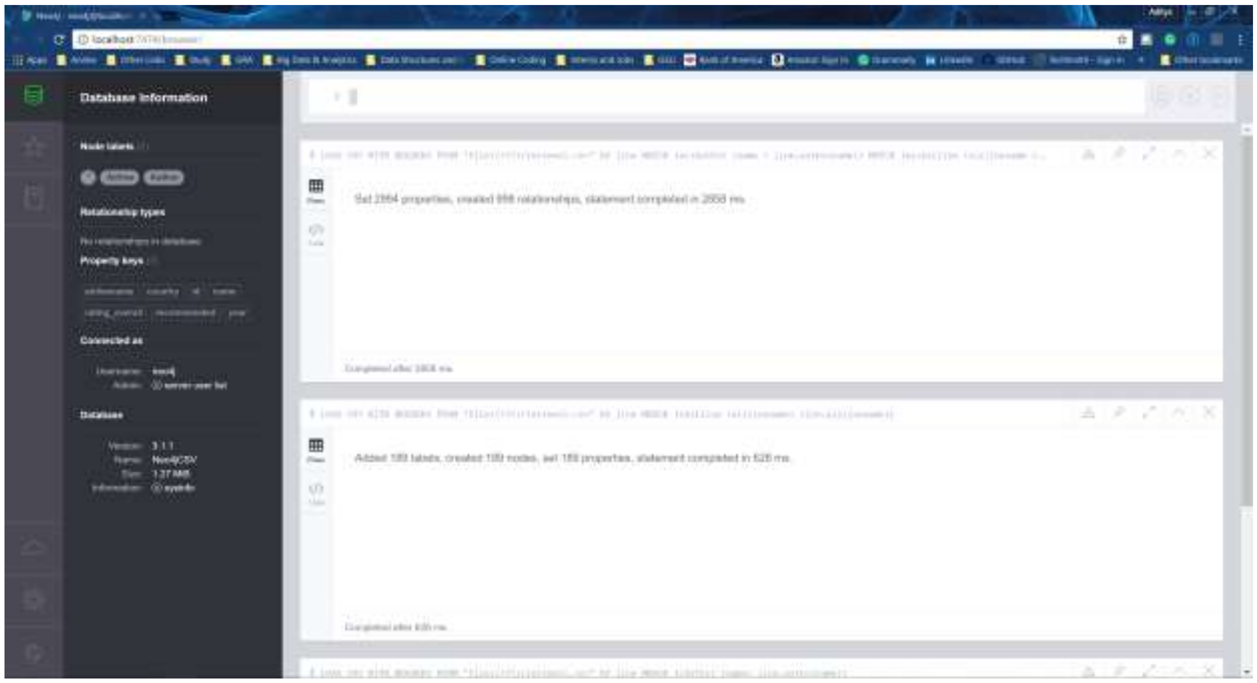
Completed after 628 ms.

```
Added 742 labels, created 742 nodes, set 742 properties, statement completed in 1006 ms.
```

Completed after 1006 ms.

3. Making Relationships:-

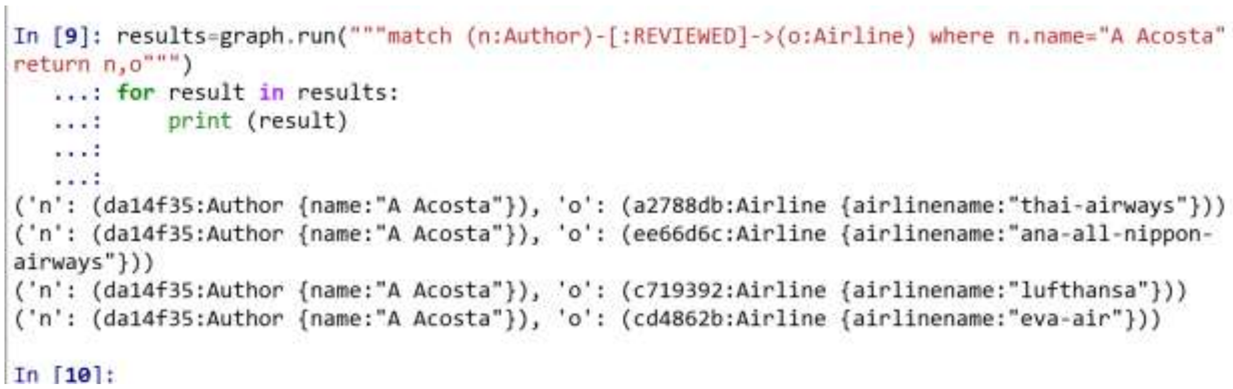
```
Query:
LOAD CSV WITH HEADERS FROM 'file:///C:/reviews1.csv' AS line
MATCH (at:Author {name : line.authorname})
MATCH (ar:Airline {airlinename : line.airlinename})
MERGE (at)-[:REVIEWED {ratingoverall:toInt(line.rating_overall),recommended:
toInt(line.recommended),lsentiment:toInt(line.lsentiment)}]->(ar)
```



```
Set 2994 properties, created 998 relationships, statement completed in 2858 ms.
```

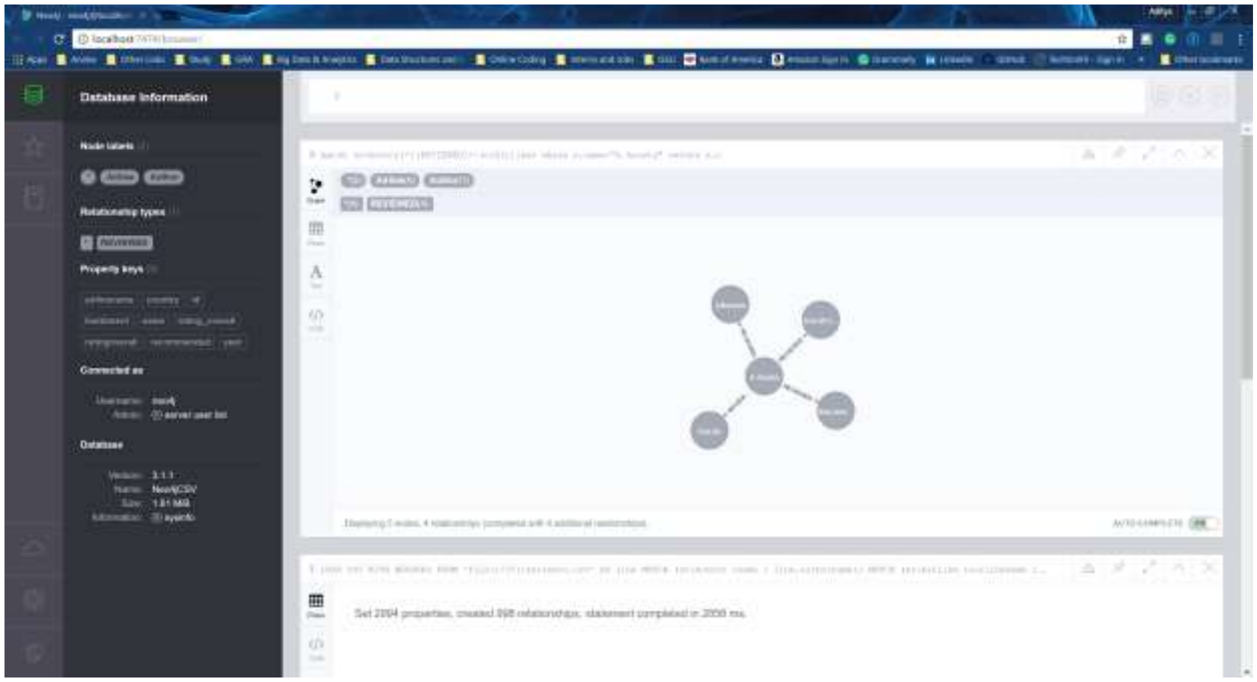
```
Added 189 labels, created 189 nodes, set 189 properties, statement completed in 628 ms.
```

4. Making a trial query.



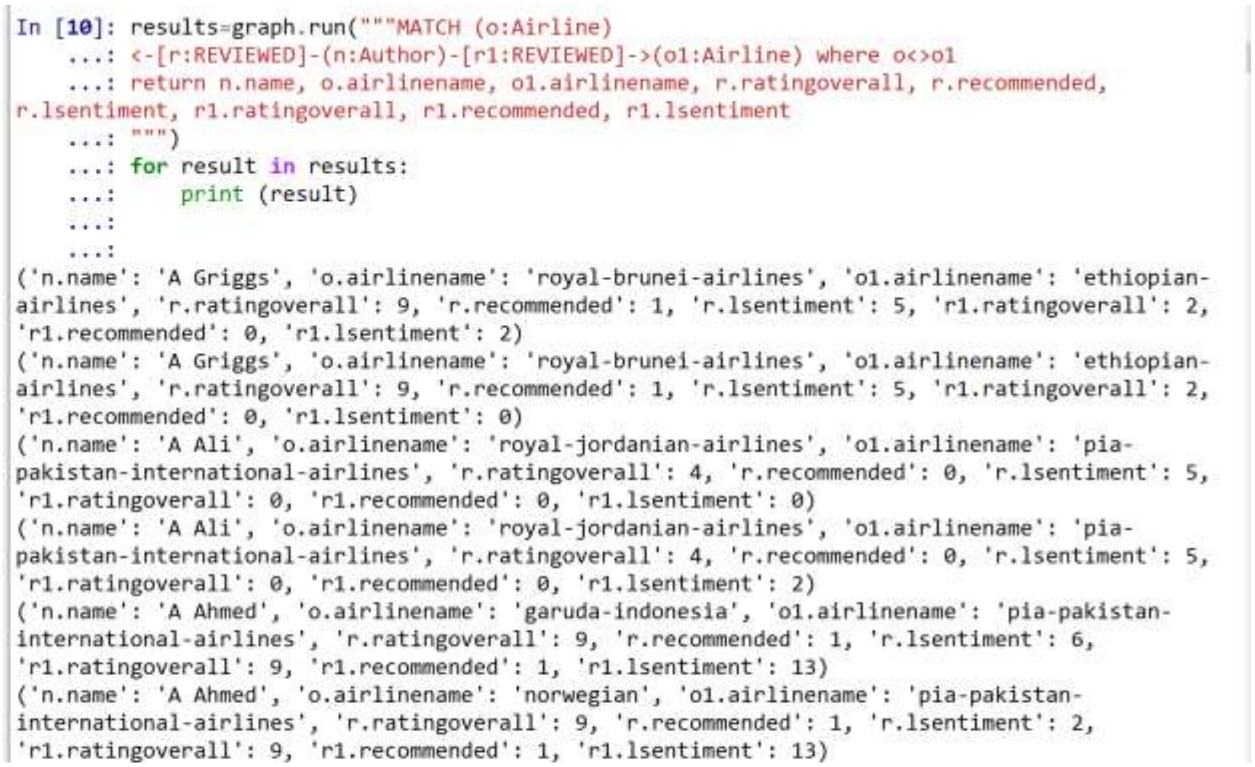
```
In [9]: results=graph.run("""match (n:Author)-[:REVIEWED]->(o:Airline) where n.name="A Acosta"
return n,o""")
   ...: for result in results:
   ...:     print (result)
   ...:
   ...:
('n': (da14f35:Author {name:"A Acosta"}), 'o': (a2788db:Airline {airlinename:"thai-airways"}))
('n': (da14f35:Author {name:"A Acosta"}), 'o': (ee66d6c:Airline {airlinename:"ana-all-nippon-
airways"}))
('n': (da14f35:Author {name:"A Acosta"}), 'o': (c719392:Airline {airlinename:"lufthansa"}))
('n': (da14f35:Author {name:"A Acosta"}), 'o': (cd4862b:Airline {airlinename:"eva-air"}))

In [10]:
```

Query:
match (n:Author)-[:REVIEWED]->(o:Airline) where n.name="A Acosta" return n,o



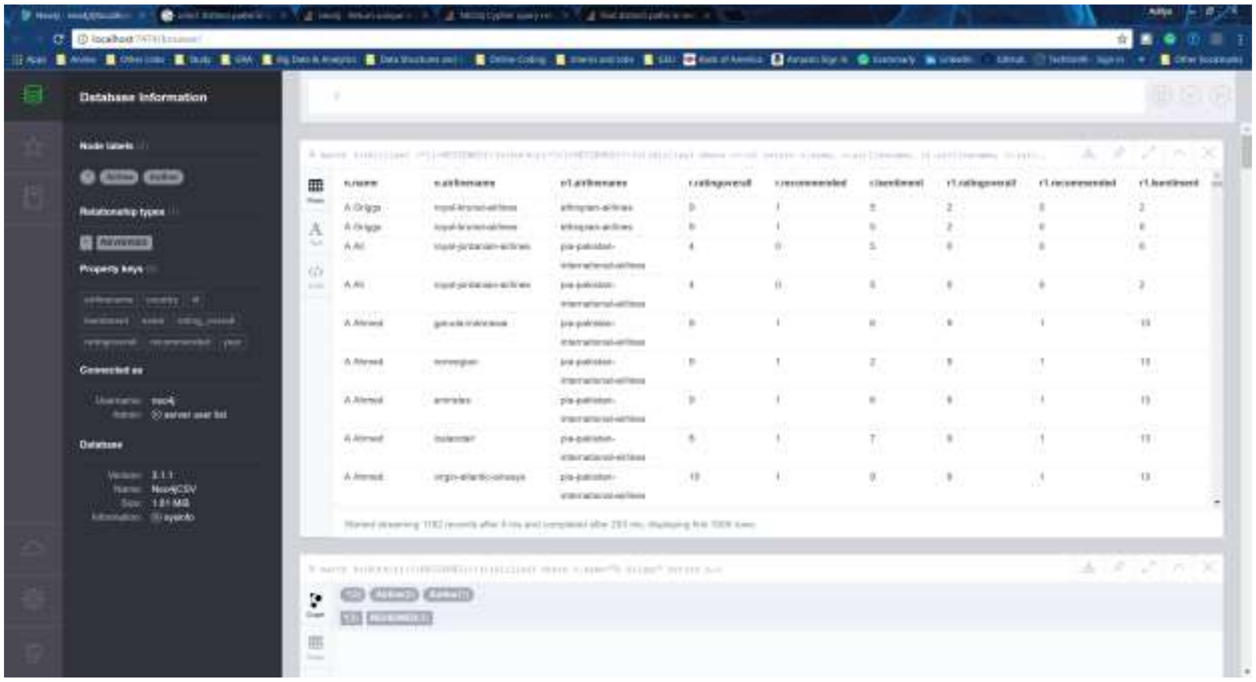
2.b For any airline company, it is important compare itself with its direct competitors based on customer rating and sentiment. Two airlines are considered direct competitors only when there are some customers rating both of them. So you need to develop a solution to compare airlines with their competitors. In terms of what to compare, it is usually important see how direct competitors are rated and perceived (in terms of sentiment) differently by their common raters;

5. Getting competing airlines based on common authors. Sometimes, same author gives different ratings for same airline.



```
In [10]: results=graph.run("""MATCH (o:Airline)
    ...: <-[r:REVIEWED]-(n:Author)-[r1:REVIEWED]->(o1:Airline) where o<>o1
    ...: return n.name, o.airlinename, o1.airlinename, r.ratingoverall, r.recommended,
r.lsentiment, r1.ratingoverall, r1.recommended, r1.lsentiment
    ...: """)
    ...: for result in results:
    ...:     print (result)
    ...:
    ...:
('n.name': 'A Griggs', 'o.airlinename': 'royal-brunei-airlines', 'o1.airlinename': 'ethiopian-
airlines', 'r.ratingoverall': 9, 'r.recommended': 1, 'r.lsentiment': 5, 'r1.ratingoverall': 2,
'r1.recommended': 0, 'r1.lsentiment': 2)
('n.name': 'A Griggs', 'o.airlinename': 'royal-brunei-airlines', 'o1.airlinename': 'ethiopian-
airlines', 'r.ratingoverall': 9, 'r.recommended': 1, 'r.lsentiment': 5, 'r1.ratingoverall': 2,
'r1.recommended': 0, 'r1.lsentiment': 0)
('n.name': 'A Ali', 'o.airlinename': 'royal-jordanian-airlines', 'o1.airlinename': 'pia-
pakistan-international-airlines', 'r.ratingoverall': 4, 'r.recommended': 0, 'r.lsentiment': 5,
'r1.ratingoverall': 0, 'r1.recommended': 0, 'r1.lsentiment': 0)
('n.name': 'A Ali', 'o.airlinename': 'royal-jordanian-airlines', 'o1.airlinename': 'pia-
pakistan-international-airlines', 'r.ratingoverall': 4, 'r.recommended': 0, 'r.lsentiment': 5,
'r1.ratingoverall': 0, 'r1.recommended': 0, 'r1.lsentiment': 2)
('n.name': 'A Ahmed', 'o.airlinename': 'garuda-indonesia', 'o1.airlinename': 'pia-pakistan-
international-airlines', 'r.ratingoverall': 9, 'r.recommended': 1, 'r.lsentiment': 6,
'r1.ratingoverall': 9, 'r1.recommended': 1, 'r1.lsentiment': 13)
('n.name': 'A Ahmed', 'o.airlinename': 'norwegian', 'o1.airlinename': 'pia-pakistan-
international-airlines', 'r.ratingoverall': 9, 'r.recommended': 1, 'r.lsentiment': 2,
'r1.ratingoverall': 9, 'r1.recommended': 1, 'r1.lsentiment': 13)
```

Query:
MATCH (o:Airline)
<-[r:REVIEWED]-(n:Author)-[r1:REVIEWED]->(o1:Airline) where o<>o1
return n.name, o.airlinename, o1.airlinename, r.ratingoverall, r.recommended, r.lsentiment,
r1.ratingoverall, r1.recommended, r1.lsentiment



Database Information

Node labels

Relationship types

Property keys

Connected as

Username: neo4j

Database

Version: 3.1.1

Name: Neo4jCSV

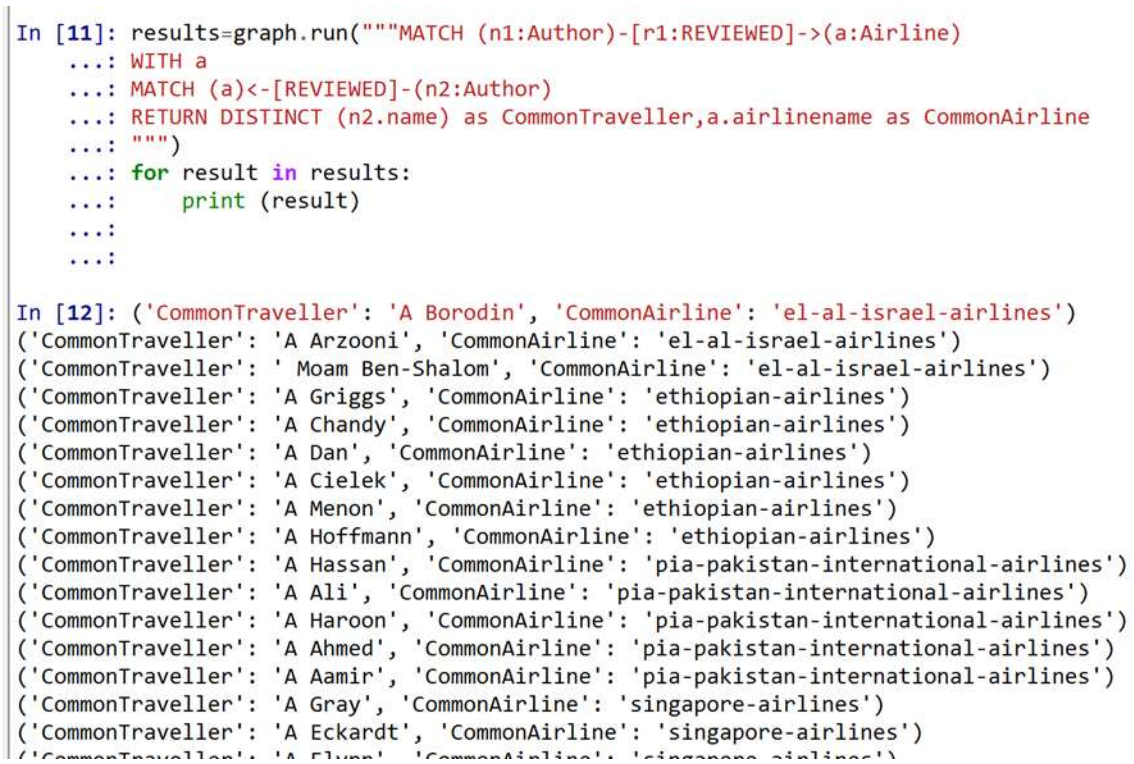
Size: 1.81 MiB

n.name	n.airlinename	r1.airlinename	r.ratingoverall	r.recommended	r.sentiment	r1.ratingoverall	r1.recommended	r1.sentiment
A Griggs	royal-brunei-airlines	ethiopian-airlines	9	1	5	2	0	2
A Griggs	royal-brunei-airlines	ethiopian-airlines	9	1	5	2	0	0
A Ali	royal-jordanian-airlines	pia-pakistan-international-airlines	4	0	5	8	0	6
A Ali	royal-jordanian-airlines	pia-pakistan-international-airlines	4	0	5	8	0	2
A Ahmed	garuda-indonesia	pia-pakistan-international-airlines	8	1	8	9	1	13
A Ahmed	norwegian	pia-pakistan-international-airlines	8	1	2	9	1	13
A Ahmed	emirates	pia-pakistan-international-airlines	9	1	6	9	1	13
A Ahmed	icelandair	pia-pakistan-international-airlines	6	1	7	9	1	13
A Ahmed	virgin-atlantic-airways	pia-pakistan-international-airlines	10	1	9	9	1	13

Started streaming 1182 records after 4 ms and completed after 233 ms, displaying first 1000 rows.

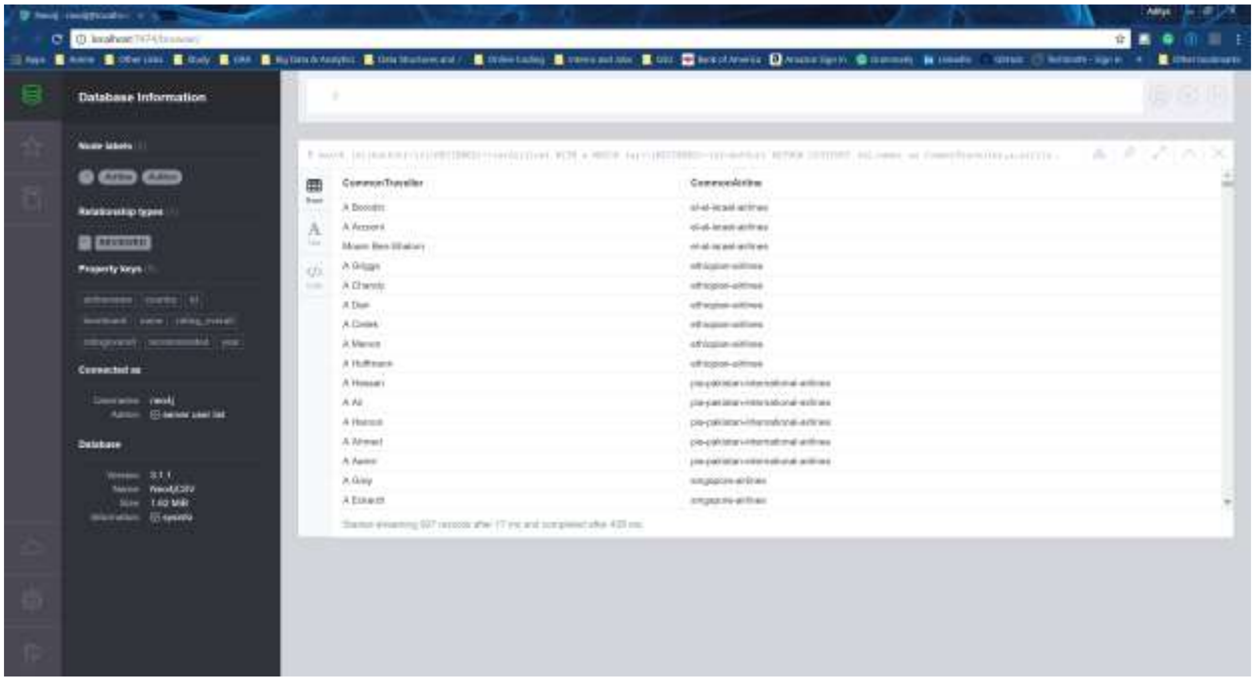
2.d Regarding raters, it is often useful to find raters with common interests. A basic analysis can be to find raters who rated and commented common airlines. You can develop a solution of your own to perform this analysis and generate some insights.

6. Getting Authors with similar interests, in this case, similar airlines.



```
In [11]: results=graph.run("""MATCH (n1:Author)-[r1:REVIEWED]->(a:Airline)
    ...: WITH a
    ...: MATCH (a)<-[REVIEWED]-(n2:Author)
    ...: RETURN DISTINCT (n2.name) as CommonTraveller,a.airlinename as CommonAirline
    ...: """)
    ...: for result in results:
    ...:     print (result)
    ...:
    ...:

In [12]: ('CommonTraveller': 'A Borodin', 'CommonAirline': 'el-al-israel-airlines')
('CommonTraveller': 'A Arzooni', 'CommonAirline': 'el-al-israel-airlines')
('CommonTraveller': ' Moam Ben-Shalom', 'CommonAirline': 'el-al-israel-airlines')
('CommonTraveller': 'A Griggs', 'CommonAirline': 'ethiopian-airlines')
('CommonTraveller': 'A Chandy', 'CommonAirline': 'ethiopian-airlines')
('CommonTraveller': 'A Dan', 'CommonAirline': 'ethiopian-airlines')
('CommonTraveller': 'A Cielek', 'CommonAirline': 'ethiopian-airlines')
('CommonTraveller': 'A Menon', 'CommonAirline': 'ethiopian-airlines')
('CommonTraveller': 'A Hoffmann', 'CommonAirline': 'ethiopian-airlines')
('CommonTraveller': 'A Hassan', 'CommonAirline': 'pia-pakistan-international-airlines')
('CommonTraveller': 'A Ali', 'CommonAirline': 'pia-pakistan-international-airlines')
('CommonTraveller': 'A Haroon', 'CommonAirline': 'pia-pakistan-international-airlines')
('CommonTraveller': 'A Ahmed', 'CommonAirline': 'pia-pakistan-international-airlines')
('CommonTraveller': 'A Aamir', 'CommonAirline': 'pia-pakistan-international-airlines')
('CommonTraveller': 'A Gray', 'CommonAirline': 'singapore-airlines')
('CommonTraveller': 'A Eckardt', 'CommonAirline': 'singapore-airlines')
('CommonTraveller': 'A Flynn', 'CommonAirline': 'singapore-airlines')
```

Query:-
MATCH (n1:Author)-[r1:REVIEWED]->(a:Airline)
WITH a
MATCH (a)<-[REVIEWED]-(n2:Author)
RETURN DISTINCT (n2.name) as CommonTraveller,a.airlinename as CommonAirline



Database Information

Node labels

Airline Author

Relationship types

REVIEWED

Property keys

Connected as

Username: neo4j

Database

Version: 3.1.1

Size: 1.60 MiB

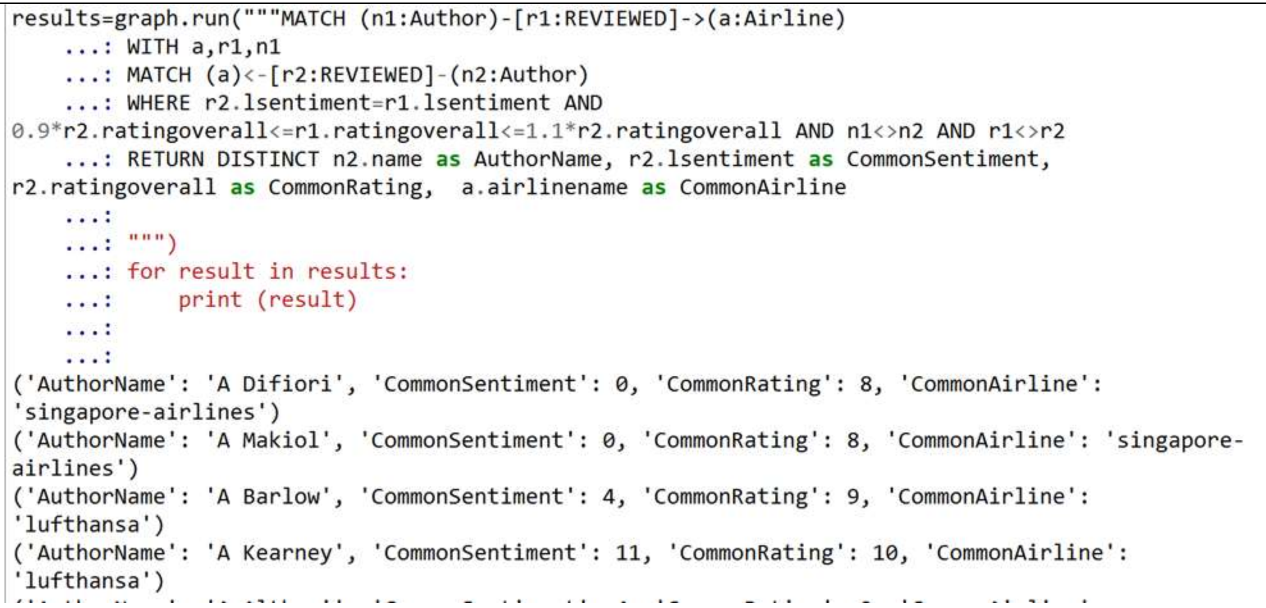
[illegible]

CommonTraveller	CommonAirline
A Boodin	el-al-israel-airlines
A Account	el-al-israel-airlines
Moam Ben Shalom	el-al-israel-airlines
A Griggs	ethiopian-airlines
A Chandy	ethiopian-airlines
A Dan	ethiopian-airlines
A Cohen	ethiopian-airlines
A Manon	ethiopian-airlines
A Huffmann	ethiopian-airlines
A Hassan	pia-pakistan-international-airlines
A Ali	pia-pakistan-international-airlines
A Haroon	pia-pakistan-international-airlines
A Ahmed	pia-pakistan-international-airlines
A Aamir	pia-pakistan-international-airlines
A Gray	singapore-airlines
A Eckardt	singapore-airlines

Started streaming 937 records after 17 ms and completed after 435 ms.

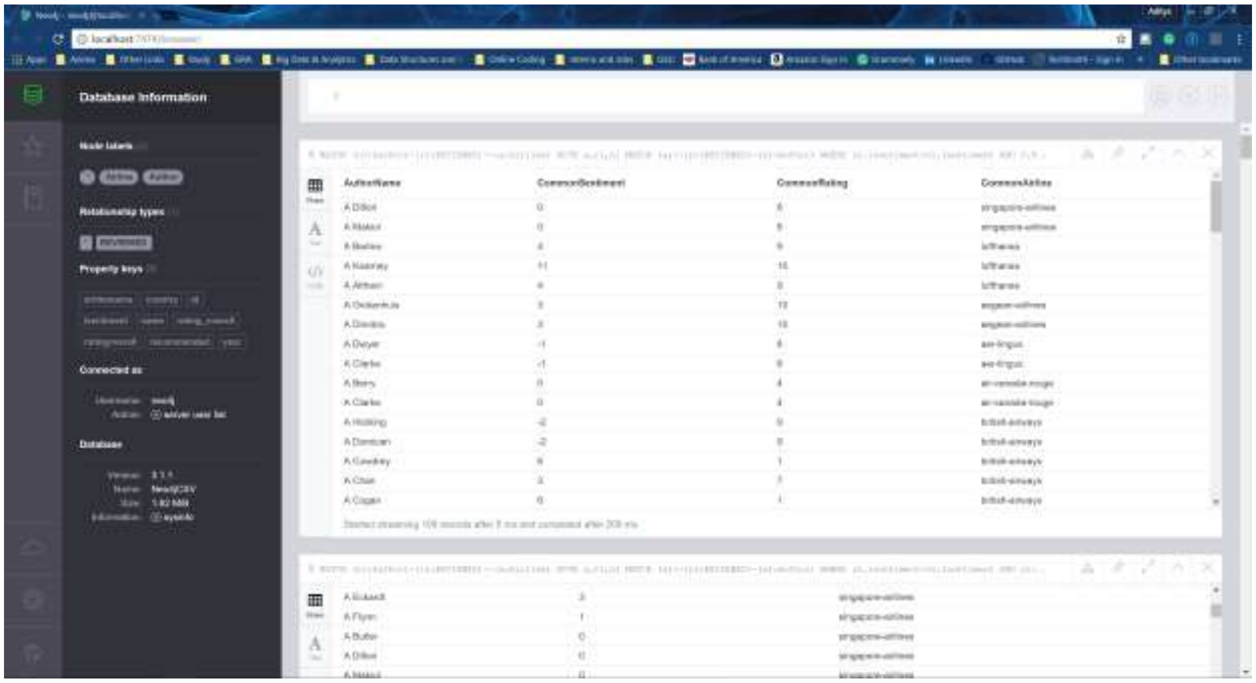
2.d Then (optionally), you can further analyze how similar these raters are with each other in terms of their ratings of the common airlines and their sentiment toward the common airlines.

7. Getting common users based on their common rating and Sentiment towards common airlines. Authors may have common rating or common sentiments.



```
results=graph.run("""MATCH (n1:Author)-[r1:REVIEWED]->(a:Airline)
    ...: WITH a,r1,n1
    ...: MATCH (a)<-[r2:REVIEWED]-(n2:Author)
    ...: WHERE r2.lsentiment=r1.lsentiment AND
0.9*r2.ratingoverall<=r1.ratingoverall<=1.1*r2.ratingoverall AND n1<>n2 AND r1<>r2
    ...: RETURN DISTINCT n2.name as AuthorName, r2.lsentiment as CommonSentiment,
r2.ratingoverall as CommonRating,  a.airlinename as CommonAirline
    ...:
    ...: """)
    ...: for result in results:
    ...:     print (result)
    ...:
    ...:
('AuthorName': 'A Difiori', 'CommonSentiment': 0, 'CommonRating': 8, 'CommonAirline':
'singapore-airlines')
('AuthorName': 'A Makiol', 'CommonSentiment': 0, 'CommonRating': 8, 'CommonAirline': 'singapore-
airlines')
('AuthorName': 'A Barlow', 'CommonSentiment': 4, 'CommonRating': 9, 'CommonAirline':
'lufthansa')
('AuthorName': 'A Kearney', 'CommonSentiment': 11, 'CommonRating': 10, 'CommonAirline':
'lufthansa')
```

Query:
MATCH (n1:Author)-[r1:REVIEWED]->(a:Airline)
WITH a,r1,n1
MATCH (a)<-[r2:REVIEWED]-(n2:Author)
WHERE r2.lsentiment=r1.lsentiment AND 0.9*r2.ratingoverall<=r1.ratingoverall<=1.1*r2.ratingoverall AND n1<>n2 AND r1<>r2
RETURN DISTINCT n2.name as AuthorName, r2.lsentiment as CommonSentiment, r2.ratingoverall as CommonRating, a.airlinename as CommonAirline



Database Information

Node labels

Airline Author

Relationship types

Property keys

Connected as

Username: neo4j

Database

Version: 3.1.1

Name: Neo4jCSV

Size: 1.82 MiB

AuthorName	CommonSentiment	CommonRating	CommonAirline
A Dillon	0	6	singapore-airlines
A Mabol	0	8	singapore-airlines
A Bailey	4	9	lufthansa
A Kearney	11	16	lufthansa
A Airtson	4	8	lufthansa
A Ockenfuls	3	13	singapore-airlines
A Dimitris	3	13	singapore-airlines
A Dwyer	-1	6	aer-lingus
A Clarke	-1	8	aer-lingus
A Berry	0	4	air-canada-rouge
A Clarke	0	4	air-canada-rouge
A Hotting	-2	5	british-airways
A Donovan	-2	8	british-airways
A Cowdrey	6	1	british-airways
A Chan	3	7	british-airways
A Cogan	6	1	british-airways

Started streaming 108 records after 3 ms and completed after 208 ms

A Eckardt	3	singapore-airlines
A Flynn	1	singapore-airlines
A Butler	0	singapore-airlines
A Dillon	0	singapore-airlines
A Mabol	0	singapore-airlines