extended_case_2

April 22, 2020

1 How is employee happiness related to their risk of leaving?

[ ]:
```
### Load relevant packages
import pandas                   as pd
import numpy                    as np
import matplotlib.pyplot        as plt
import seaborn                  as sns
import statsmodels.formula.api as sm
import os
import warnings
warnings.filterwarnings("ignore")  # Suppress all warnings

%matplotlib inline
plt.style.use('ggplot')
```

1.1 Introduction

Business Context. Companies generally like to retain their employees, since hiring and training new employees is costly and risky. The rate at which employees are quitting is called the turnover rate, and the high cost of employee turnover has been pointed out extensively in management literature. A high turnover rate not only increases human resource costs, which can reach up to 150% of the annual salary per replaced employee, but it also has social costs, since it is correlated with lower wages, lower productivity per employee, and not surprisingly, a less loyal workforce. For instance, in 2006, turnover at Walmart's Sam's Club was 44% with an average hourly pay of \$10.11, while at Costco it was a much lower 17% with a higher \$17.00 hourly wage [2]. In addition, a more recent study correlated companies with low turnover with a series of socially positive characteristics dubbed "high-involvement work practices".

Studying employee happiness at workplaces is an important area of investigation in the modern economic market. This enables company managers get a sense of which features are related to turnover. You work in your company's new talent analytics department. Your company has thousands of employees and is interested in reducing the time spent recruiting and hiring new employees to replace the people that quit.

Business Problem. Your company has tasked you to answer the following question: **"Does employee happiness predict turnover?"**

Analytical Context. In the last cases, you practiced data literacy, information gathering, and information sufficiency. You have also practiced basic exploratory data analysis (EDA). In this case, you will continue to practice applying these tools, but you will also see more advanced EDA. In this case, there are so many variables that it is difficult to meaningfully explore all of them at the same time. As a result, we will focus on exploring small subsets of the variables. The dataset we will be working with is from 34 different Spanish companies that recorded employee data as part of a corporate feedback program.

1.2 Understanding the data

The employee dataset was collected across 34 different companies from May 10, 2014 to March 8, 2017. The companies belong to one of the following sectors:

1. E-payments
2. IT consulting services
3. Retail
4. Manufacturing
5. Services
6. Tourism
7. Education

About half of the companies are multinational and the other half are Barcelona-based companies. The bulk of the employees used the app in Spain (Barcelona area), and more than 90% are Spanish nationals. The comments are written in various languages: 97% in Spanish, 2% in English, and 1% Catalan. 1.6% of the comment file (measured in bytes) corresponded to emoji symbols. Employee turnover was detected when the employee was removed from the Enterprise Resource Planning (ERP) system which is linked to backend of the app provider.

The data consists of four tables: `votes`, `comments`, `interactions`, and `churn` (employee turnover). A vote was obtained when an employee opened the app and answered the question: > How happy are you at work today?

To vote, the employee indicates their feeling by touching one of four icons that appeared on the screen:

- **4: Great**
- **3: Good**
- **2: So-so**
- **1: Pretty Bad**

After the employee indicates their happiness level, a second screen appears where they can input a text explanation (usually a complaint, suggestion, or comment); this is the `comments` table. Out of 4,356 employees, 2,638 employees commented at least once. Finally, in a third screen the employee can see their peers' comments and like or dislike them; this data is stored in the `interactions` table. 3,516 employees liked or disliked at least one of their peers' comments.

Let's take a look at the raw tables:

[21]:
```
# Load in the raw dataframes.
votes_df    = pd.read_csv("files/votes.csv")
comments_df = pd.read_csv("files/comments.csv")
likes_df    = pd.read_csv("files/interactions.csv")
churn_df    = pd.read_csv("files/churn.csv")
```

[22]:
```
votes_df.head(10)
```

[22]:

	employee	companyAlias	voteDate	vote
0	31	56aec740f1ef260003e307d6	Mon Feb 01 01:00:00 CET 2016	4
1	33	56aec740f1ef260003e307d6	Mon Feb 01 01:00:00 CET 2016	4
2	79	56aec740f1ef260003e307d6	Mon Feb 01 01:00:00 CET 2016	4
3	94	56aec740f1ef260003e307d6	Mon Feb 01 01:00:00 CET 2016	4
4	16	56aec740f1ef260003e307d6	Mon Feb 01 01:00:00 CET 2016	2
5	20	56aec740f1ef260003e307d6	Mon Feb 01 01:00:00 CET 2016	2
6	22	56aec740f1ef260003e307d6	Mon Feb 01 01:00:00 CET 2016	2
7	41	56aec740f1ef260003e307d6	Mon Feb 01 01:00:00 CET 2016	2
8	83	56aec740f1ef260003e307d6	Mon Feb 01 01:00:00 CET 2016	2
9	12	56aec740f1ef260003e307d6	Mon Feb 01 01:00:00 CET 2016	3

[23]:
```
comments_df.head(10)
```

[23]:

	employee	companyAlias	commentId	\
0	307	56aec740f1ef260003e307d6	58d018d7e010990004e38070	
1	382	56aec740f1ef260003e307d6	58d0179ae010990004e3806d	
2	172	56aec740f1ef260003e307d6	58cff8cde010990004e37f6a	
3	135	56aec740f1ef260003e307d6	58cfefeee010990004e37f60	
4	225	56aec740f1ef260003e307d6	58cfd9b4e010990004e37f52	
5	18	56aec740f1ef260003e307d6	58cfd7e2e010990004e37f4f	
6	175	56aec740f1ef260003e307d6	58cfbdb53047cd000460dbcd	
7	491	56aec740f1ef260003e307d6	58cfbc333047cd000460dbcc	
8	158	56aec740f1ef260003e307d6	58cfb1ee3047cd000460dbb9	
9	124	56aec740f1ef260003e307d6	58cfa0af3047cd000460dba7	

	txt	likes	dislikes	\
0	**…	4.0	0.0	
1	******************************	1.0	2.0	
2	**************************	3.0	0.0	
3	**************************	1.0	1.0	
4	**********************************	3.0	2.0	
5	**…	1.0	0.0	
6	**…	13.0	1.0	
7	**…	4.0	0.0	
8	**…	5.0	0.0	
9	********************	2.0	0.0	

	commentDate

```
0  Mon Mar 20 19:00:17 CET 2017
1  Mon Mar 20 18:55:16 CET 2017
2  Mon Mar 20 16:44:02 CET 2017
3  Mon Mar 20 16:06:08 CET 2017
4  Mon Mar 20 14:30:50 CET 2017
5  Mon Mar 20 14:23:02 CET 2017
6  Mon Mar 20 12:28:45 CET 2017
7  Mon Mar 20 12:23:54 CET 2017
8  Mon Mar 20 11:39:14 CET 2017
9  Mon Mar 20 10:27:51 CET 2017
```

[24]:
```
likes_df.head(10)
```

[24]:

	employee	companyAlias	liked	disliked	commentId
0	307	56aec740f1ef260003e307d6	True	False	58d018d7e010990004e38070
1	36	56aec740f1ef260003e307d6	True	False	58d018d7e010990004e38070
2	276	56aec740f1ef260003e307d6	True	False	58d018d7e010990004e38070
3	24	56aec740f1ef260003e307d6	True	False	58d018d7e010990004e38070
4	382	56aec740f1ef260003e307d6	True	False	58d0179ae010990004e3806d
5	24	56aec740f1ef260003e307d6	False	True	58d0179ae010990004e3806d
6	217	56aec740f1ef260003e307d6	False	True	58d0179ae010990004e3806d
7	164	56aec740f1ef260003e307d6	True	False	58cff8cde010990004e37f6a
8	34	56aec740f1ef260003e307d6	True	False	58cff8cde010990004e37f6a
9	152	56aec740f1ef260003e307d6	True	False	58cff8cde010990004e37f6a

[25]:
```
churn_df.head(10)
```

[25]:

	employee	companyAlias	numVotes	lastParticipationDate
0	512	56aec740f1ef260003e307d6	4	Thu Feb 23 12:48:04 CET 2017
1	-2	56aec740f1ef260003e307d6	0	Wed Jan 18 14:00:55 CET 2017
2	2	56aec740f1ef260003e307d6	72	Fri Mar 17 01:00:00 CET 2017
3	487	56aec740f1ef260003e307d6	14	Sat Nov 19 15:02:14 CET 2016
4	3	56aec740f1ef260003e307d6	22	Thu Feb 16 01:00:00 CET 2017
5	-4	56aec740f1ef260003e307d6	0	Mon Nov 07 17:41:56 CET 2016
6	4	56aec740f1ef260003e307d6	195	Mon Mar 20 01:00:00 CET 2017

```
7       516  56aec740f1ef260003e307d6        29  Mon Mar 20 12:28:45 CET 2017
8       475  56aec740f1ef260003e307d6        15  Sun Nov 06 19:38:30 CET 2016
9         5  56aec740f1ef260003e307d6        42  Tue Mar 14 01:00:00 CET 2017
```

```
   stillExists
0         True
1        False
2         True
3        False
4         True
5        False
6         True
7         True
8        False
9         True
```

1.2.1 Exercise 1:

1. What variables can you use as a measure of employee happiness? What variables can you use as a measure of employee retention?
2. What other factors can contribute to employee retention that are not related to happiness? Are these factors included in any of the datasets above? Which ones are, and which ones are missing?

Answer. We can use employees' like/dislike votes from `likes_df` as a measure of their happiness. We can use employees' `stillExists` entries as a measure of their retention.

The set of **confounding factors** is much larger. Some examples, which are not included in this dataset, include the employee's compensation level, feelings of community/beloging in the workplace, satisfaction with their life outside work, relationship with their spouse or significant other, and their performance on the job.

One thing to note is that most factors which affect happiness at work also affect retention (because employees often leave because they are not happy). However, there are some factors that affect retention that have nothing to do with happiness, such as life events which require the employee to move (e.g. spouse's job changed location) or desire to change career paths.

1.3 Exploring employee happiness votes (40 minutes)

We will begin by exploring the first dataset, containing data about employees' votes using the app. We will get a rough sense of the size of the data available to us and employee engagement on the app, which is the measurement tool that was used to the collect the data we are analyzing.

1.3.1 Data Cleaning (`votes_df`)

While other cases will focus on teaching you proper data cleaning, here the code is provided for you. Feel free to try and understand what the code does and why this cleaning is necessary. Oftentimes

we remove erroneous, superfluous, or cumbersome data.

[26]:
```
# Remove companies that have low engagement numbers and no comments.
votes_df = votes_df[(votes_df['companyAlias'] != "5474b9cde4b0bf7614b2c66f") &
                    (votes_df['companyAlias'] != "58bf03e5cff4fa0004dd44ef") &
                    (votes_df['companyAlias'] != "573a0671b5ec330003add34a")]
```

[27]:
```
# Replace companyAlias column with shorter labels for readability.
labels = ["A","B","C","D","E","F","G","H","I","J","K","L",
          "M","N","O","P","Q","R","S","T","V","W","X","Y",
          "Z","AA","AB","AC","AD","AE","AF","AG","AH","AI","AJ"]
unique_company_alias = votes_df['companyAlias'].unique()
mapAlias2Labels = dict(zip(unique_company_alias, labels))
votes_df['companyAlias'] = votes_df['companyAlias'].replace(mapAlias2Labels)
```

[28]:
```
# Clean dates from timezone.
votes_df['voteDate'] = votes_df['voteDate'].map(lambda date: str(date).strip()
                        .replace('CEST', '')
                        .replace('CET', ''))

# Parse date string to date time with format: Mon Dec 25 19:23:59 2019.
votes_df['voteDate'] = pd.to_datetime(votes_df['voteDate'], format="%a %b %d %H:
 ↪%M:%S %Y")
```

1.3.2 Exercise 2:

2.1 How generalizable would any analysis on this dataset be? What other features of the companies would be important to gather to make our analysis more generalizable?

Answer. A simple `companyAlias` count reveals that there are only 34 total companies represented. This is enough to get a sense of what is going on, but the sample size may be too small to generalize. Also of importance are segmenting variables such as company size (head count, number of cities, revenue, etc.), headquarters location (Spain, Europe, multinational, etc.), and sector (agriculture, IT, finance, etc.). All of these things could have a material impact on employee happiness and need to be taken into account.

2.2 What issues could there be with the integrity of the data itself?

Answer. One thing that is not addressed in this dataset is that of **sampling bias**. We don't know how this feedback was even collected. For example, were employees able to submit feedback truly anonymously? Were they required to submit feedback? These are two major factors that can skew the results.

1.3.3 Exercise 3:

3.1 For now, we'll keep the points in Exercise 2 in mind, and proceed ahead. To make our analysis easier, create a new attribute `UID` that is a unique value for each data value by combining `employee` and `companyAlias`.

Answer. One possible solution is given below:

[29]:
```
votes_df['uid'] = (votes_df[['employee', 'companyAlias']]
                   .apply(lambda x: '{}{}'.format(x[0], x[1]).replace('\n',␣
 ↪''), axis=1))
```

3.2 Since we have identified voluntary responsiveness as a potential confounding factor in the sampling design, let's look at how many employees per company actually used the app to give a response. Using your newly created `uid`, draw a barplot showing the number of employees who voted on happiness at least once for each company alias.

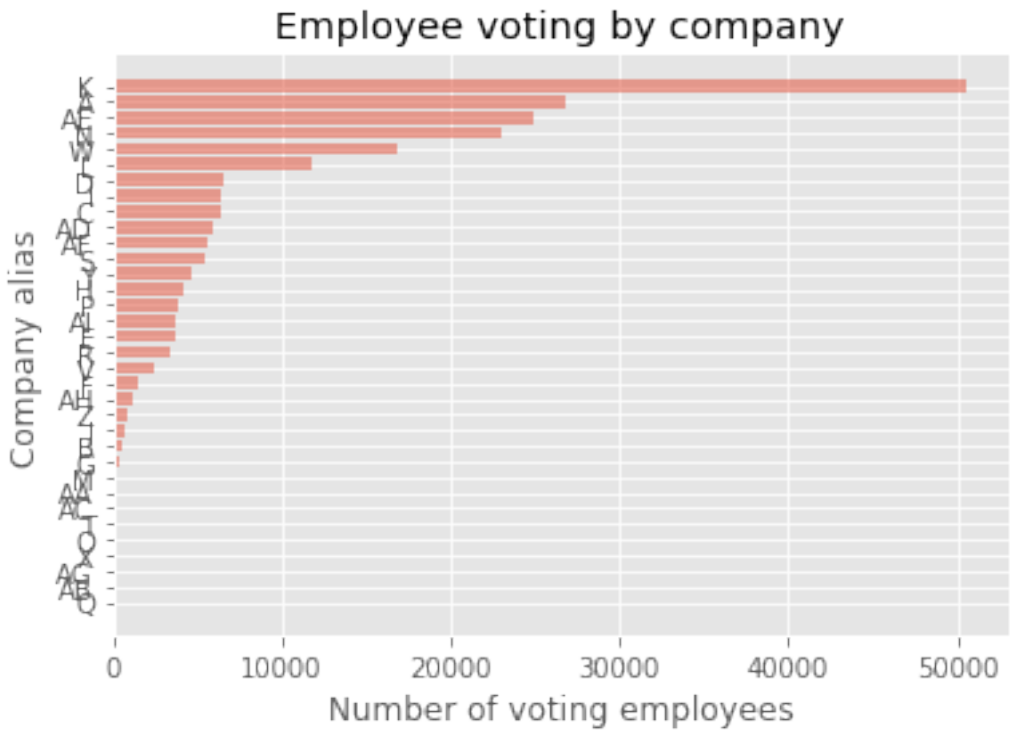
Answer. One possible solution is given below:

[30]:
```
# First find the vote frequency per company alias.
votes_frequency_table = (votes_df.groupby('companyAlias')['uid']
                           .count()
                           .sort_values(ascending=True)
                           .reset_index(name='n_employees'))
votes_frequency_table.head(10)
```

[30]:

	companyAlias	n_employees
0	Q	28
1	AB	42
2	AG	53
3	X	63
4	O	92
5	T	97
6	AC	153
7	AA	162
8	M	191
9	G	351

[31]:
```
# Then plot number of voting employees vs. company alias
objects = (votes_frequency_table["companyAlias"])
y_pos = np.arange(len(objects))
performance = votes_frequency_table["n_employees"]

plt.barh(y_pos, performance, align='center', alpha=0.5)
plt.yticks(y_pos, objects)
plt.xlabel('Number of voting employees')
plt.ylabel('Company alias')
plt.title('Employee voting by company')

plt.show()
```



1.4 Exploring employee feedback comments

Let's now explore the feedback comments.

1.4.1 Data Cleaning (`comments_df`)

Run the lines below to clean the `comments_df` table.

[32]:
```
# Rename columns to something shorter and lowercase.
col_rename_map = {
    'commentDate': 'date',
    'dislikes': 'nolike',
    'likes': 'like',
    'commentId': 'commentid',
    'companyAlias': 'coa',
    'employee': 'id'
}

comments_df.rename(columns=col_rename_map, inplace = True)
```

[33]:
```
# We remove the companies that have low numbers and no comments.
comments_df = comments_df[(comments_df['coa'] != "5474b9cde4b0bf7614b2c66f") &
```

```
                          (comments_df['coa'] != "58bf03e5cff4fa0004dd44ef")
                         ]
```

[34]:
```
# Clean dates from timezone.
comments_df['date'] = comments_df['date'].map(lambda date: str(date).strip()
                        .replace('CEST', '')
                        .replace('CET', ''))

# Parse date string to date time with format: Mon Dec 25 19:23:59 2019.
comments_df['date'] = pd.to_datetime(comments_df['date'], format="%a %b %d %H:
 ↪%M:%S %Y")
```

[35]:
```
# Replace companyAlias column with shorter labels.
unique_company_alias = comments_df['coa'].unique()
mapAlias2Labels = dict(zip(unique_company_alias, labels))
comments_df['coa'] = comments_df['coa'].replace(mapAlias2Labels)
```

[36]:
```
# Make a unique ID for every entry by concatenating employee ID and company␣
 ↪alias.
comments_df['uid'] = (comments_df[['id', 'coa']]
                    .apply(lambda x: '{}{}'.format(x[0], x[1]).replace('\n',␣
 ↪''), axis=1))
```

1.4.2 Exercise 4:

Notice that something challenging about the `comments_df` table is that the `text` column, containing the text of the employee comments themselves, have been de-identified. This means a comment like `I hate my company` is reduced to `**********`: one `*` for every character in the comment. While this reduces the amount of information on employee comments substantially, is there anything we can still gain from analyzing this data?

Answer. Yes - even though the comments themselves have been removed, we can still gain some insight by looking at the length of each comment. There are two reasonable hypotheses we can make that, if true, lead to metrics of employee happiness given the data available to us: (1) *happier employees are more likely to vote positively on peers' comments* and (2) *unhappier employees tend to write longer comments*. There is evidence in psychological literature that both of these phenomena occur, and are likely present in our dataset as well.

1.4.3 Exercise 5:

Let's go through and create features related to our thoughts above.

5.1 Count how many likes there are for each company alias.

Answer. One possible solution is given below:

[37]:
```
comments_df.groupby('coa')['like'].nunique().sort_values(ascending=True).
↪head(15)
```

[37]:
```
coa
AB     2
AC     3
G      3
T      3
AG     4
B      5
J      5
Q      5
M      5
AA     5
O      7
P     13
AH    13
Y     14
V     16
Name: like, dtype: int64
```

5.2 Add a column `lcomment` that counts the length of the comment in `txt`.

Answer. One possible solution is given below:

[38]:
```
# Replace NaN with empty string, so that it is correctly counted as zero length.
comments_df['txt'].fillna('', inplace=True)

# Count comment text length and insert into column called `lcomment`.
comments_df['lcomment'] = comments_df['txt'].apply(lambda txt: len(str(txt)))
```

5.3 Determine the average comment length per company. Call this feature `n_comments`. Plot the distribution of `n_comments` across the companies.

Answer. One possible solution is given below:

[40]:
```
comment_freq_table = (comments_df
                      .groupby('coa')['uid']
                      .nunique()
                      .sort_values(ascending=True))
```

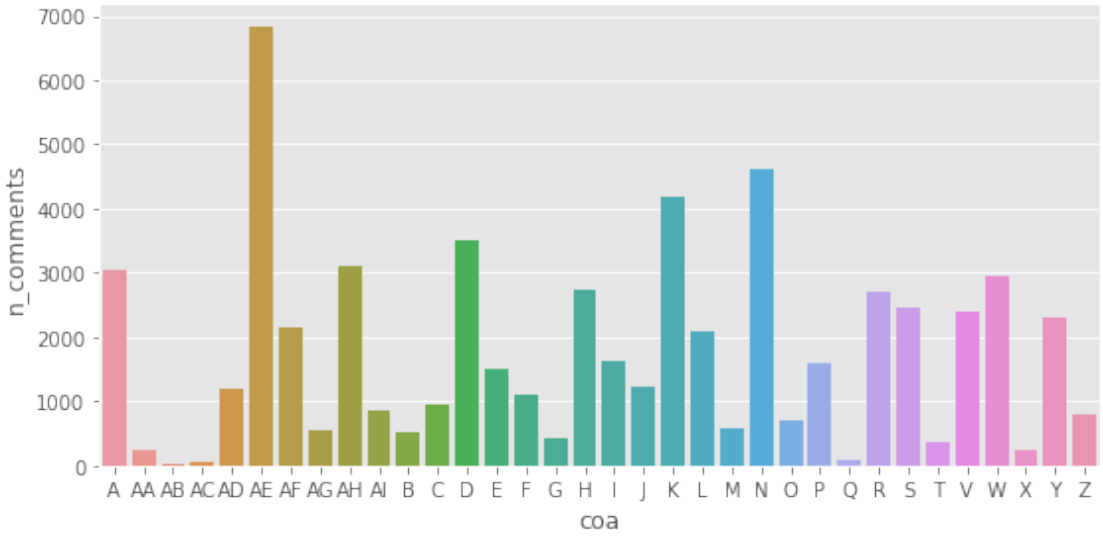
[41]:
```
# Take the sum of `lcomment`.
text_length_table = (comments_df
                      .groupby('coa')['lcomment']
                      .sum()
                      .sort_values(ascending=True))
```

```
# Normalize total text length by employee count from the previous frequency␣
↪table.
text_length_table_norm = text_length_table / comment_freq_table
text_length_table_norm = text_length_table_norm.reset_index(name="n_comments")

text_length_table_norm.head(10)
```

[41]:

	coa	n_comments
0	A	3043.854015
1	AA	231.411765
2	AB	27.500000
3	AC	57.176471
4	AD	1202.372881
5	AE	6817.783439
6	AF	2135.129032
7	AG	542.000000
8	AH	3119.945946
9	AI	856.910853

[42]:
```
sns.catplot(x="coa", y="n_comments", kind="bar",
            data=text_length_table_norm, height=4, aspect=2);
```



1.5 Exploring employee interactions and likes

We will now explore the third dataset, containing data about employees' interactions (by liking each other's comments) using the app.

1.5.1 Data Cleaning (`likes_df`)

[43]:
```
# Remove the company with alias 58a728a (useless data).
likes_df = likes_df[likes_df['companyAlias'] != '58a728a']
```

[44]:
```
# Replace companyAlias column with shorter labels.
unique_company_alias = likes_df['companyAlias'].unique()
mapAlias2Labels = dict(zip(unique_company_alias, labels))
likes_df['companyAlias'] = likes_df['companyAlias'].replace(mapAlias2Labels)
```

[45]:
```
# Make a unique ID for every entry by concatenating employee ID and company␣
↪alias.
likes_df['uid'] = (likes_df[['employee', 'companyAlias']]
                   .apply(lambda x: '{}{}'.format(x[0], x[1]).replace('\n',␣
↪''), axis=1))
```

[46]:
```
# Check that the dataframe now looks the way you expect.
likes_df.head(10)
```

[46]:

	employee	companyAlias	liked	disliked	commentId	uid
0	307	A	True	False	58d018d7e010990004e38070	307A
1	36	A	True	False	58d018d7e010990004e38070	36A
2	276	A	True	False	58d018d7e010990004e38070	276A
3	24	A	True	False	58d018d7e010990004e38070	24A
4	382	A	True	False	58d0179ae010990004e3806d	382A
5	24	A	False	True	58d0179ae010990004e3806d	24A
6	217	A	False	True	58d0179ae010990004e3806d	217A
7	164	A	True	False	58cff8cde010990004e37f6a	164A
8	34	A	True	False	58cff8cde010990004e37f6a	34A
9	152	A	True	False	58cff8cde010990004e37f6a	152A

Consider the following summary statistics on the variables in `likes_df` and then answer the question below.

[47]:
```
likes_df.describe(include='all')
```

[47]:

	employee	companyAlias	liked	disliked	commentId	\
count	336959.000000	336959	336959	336959	336959	
unique	NaN	34	2	2	37520	
top	NaN	K	True	False	58be80b80d9dcc0004d853ac	
freq	NaN	73691	284398	284398	86	
mean	162.880098	NaN	NaN	NaN	NaN	
std	182.069568	NaN	NaN	NaN	NaN	
min	-218.000000	NaN	NaN	NaN	NaN	
25%	38.000000	NaN	NaN	NaN	NaN	
50%	122.000000	NaN	NaN	NaN	NaN	
75%	212.000000	NaN	NaN	NaN	NaN	

```
max         999.000000          NaN      NaN       NaN                        NaN
```

```
           uid
count   336959
unique    3516
top       101K
freq      3774
mean       NaN
std        NaN
min        NaN
25%        NaN
50%        NaN
75%        NaN
max        NaN
```

1.5.2 Exercise 6:

6.1 Determine the average `liked` across the entire dataset.

Answer. One possible solution is shown below:

[48]:
```
print(likes_df['liked'].mean())
```

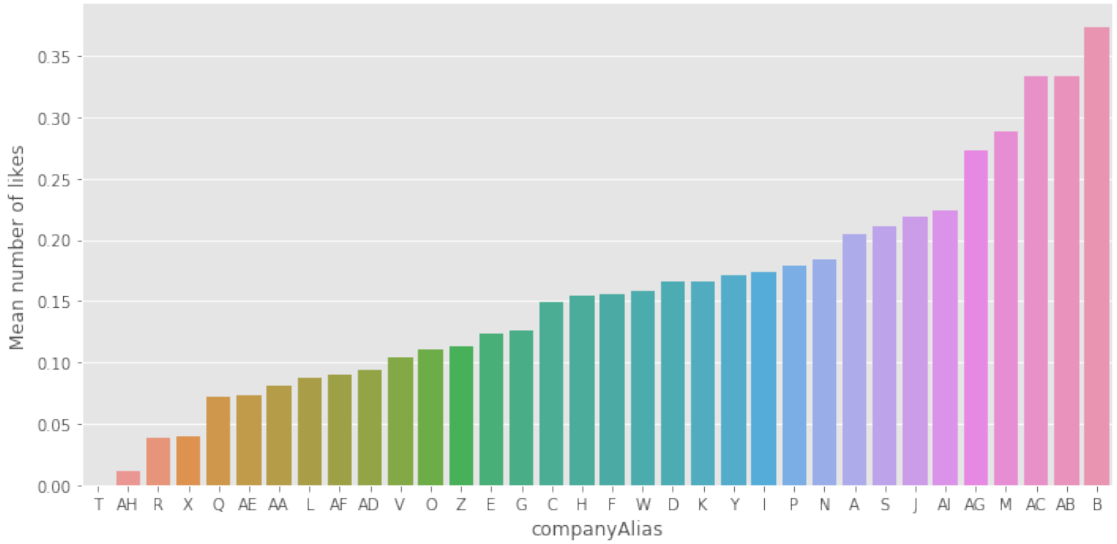
```
0.844013663383379
```

6.2 How useful is the above metric? Why or why not? If not, please indicate what alternative metric(s) you would look at and create a plot for it if helpful. In this exercise, we will determine the mean of `liked` by company and plot the resulting distribution.

Answer. This metric is not useful since we know there are multiple employee entries for each company, and each company has a different number of employees. Therefore, this metric is skewed towards companies with more employees. An alternative would be to compute the average `liked` metric for each company separately and then plot a bar graph of their levels:

[49]:
```
# Convert to numeric.
likes_df['liked'] = (likes_df['liked'] == 0)

# Find mean likes by company.
meanLikedByCompany = (likes_df.groupby('companyAlias')['liked']
                      .mean()
                      .sort_values(ascending=True)
                      .reset_index(name='mean_liked'))

# Plot the result.
sns.catplot(x='companyAlias', y='mean_liked', kind='bar',
            data=meanLikedByCompany, height=5, aspect=2);
plt.ylabel('Mean number of likes')
```

[49]: Text(0.42499999999999716, 0.5, 'Mean number of likes')



1.6 Exploring the relationship between happiness features and turnover

1.6.1 Data Cleaning (churn_df)

[50]:
```
# Remove the companies with useless data.
churn_df = churn_df[churn_df['companyAlias'] != '573a0671b5ec330003add34a']
churn_df = churn_df[churn_df['companyAlias'] != '58a728a0e75bda00042a3468']
churn_df = churn_df[churn_df['companyAlias'] != '573a0671b5ec330003add34a']
```

[51]:
```
# Replace companyAlias column with shorter labels.
unique_company_alias = churn_df['companyAlias'].unique()
mapAlias2Labels = dict(zip(unique_company_alias, labels))
churn_df['companyAlias'] = churn_df['companyAlias'].replace(mapAlias2Labels)
```

[52]:
```
# Make a unique ID for every entry by concatenating employee ID and company␣
↪alias.
churn_df['uid'] = (churn_df[['employee', 'companyAlias']]
                   .apply(lambda x: '{}{}'.format(x[0], x[1]).replace('\n',␣
↪''), axis=1))
```

[53]:
```
# Check that the dataframe now looks the way you expect.
churn_df.head(10)
```

[53]:

	employee	companyAlias	numVotes	lastParticipationDate	stillExists	\
0	512	A	4	Thu Feb 23 12:48:04 CET 2017	True	

```
1        -2            A          0  Wed Jan 18 14:00:55 CET 2017        False
2         2            A         72  Fri Mar 17 01:00:00 CET 2017         True
3       487            A         14  Sat Nov 19 15:02:14 CET 2016        False
4         3            A         22  Thu Feb 16 01:00:00 CET 2017         True
5        -4            A          0  Mon Nov 07 17:41:56 CET 2016        False
6         4            A        195  Mon Mar 20 01:00:00 CET 2017         True
7       516            A         29  Mon Mar 20 12:28:45 CET 2017         True
8       475            A         15  Sun Nov 06 19:38:30 CET 2016        False
9         5            A         42  Tue Mar 14 01:00:00 CET 2017         True

    uid
0  512A
1   -2A
2    2A
3  487A
4    3A
5   -4A
6    4A
7  516A
8  475A
9    5A
```

1.6.2 Exercise 7:

In this exercise, we will put several things together to answer the original question: *Is employee happiness associated with retention?*

7.1 Find and plot mean retention by company.

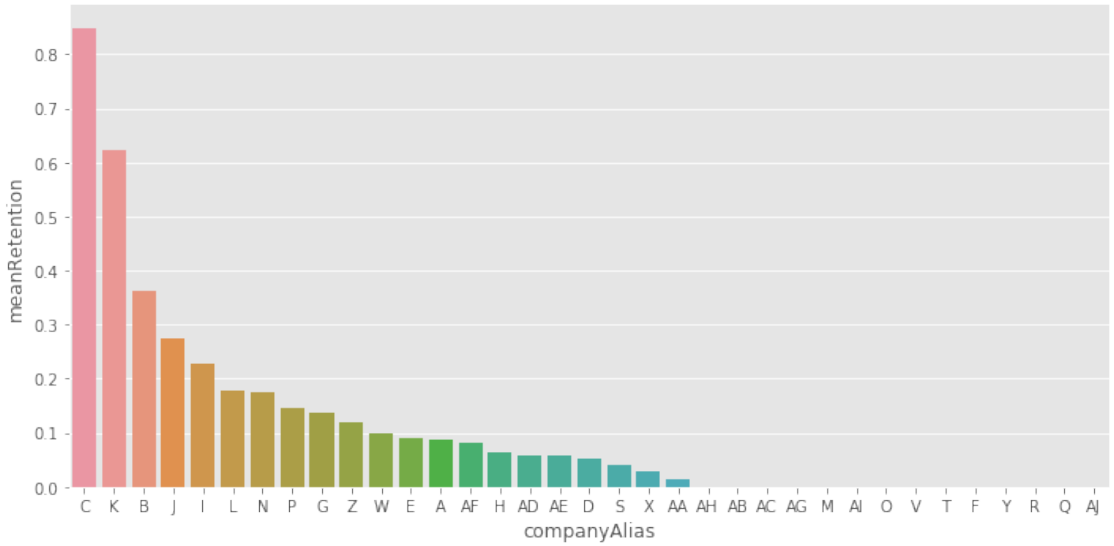
Answer. One possible solution is given below:

[54]:
```
# Convert to numeric.
churn_df['stillExists'] = (churn_df['stillExists'] == 0)

# Find mean likes by company.
meanRetentionByCompany = (churn_df.groupby('companyAlias')['stillExists']
                        .mean()
                        .sort_values(ascending=False)
                        .reset_index(name='meanRetention'))

meanRetentionByCompany
```

[54]:
```
  companyAlias  meanRetention
0            C       0.848214
1            K       0.623574
2            B       0.363636
```

3	J	0.274510
4	I	0.228758
5	L	0.176471
6	N	0.174107
7	P	0.146667
8	G	0.137931
9	Z	0.120000
10	W	0.100000
11	E	0.088757
12	A	0.087079
13	AF	0.081633
14	H	0.063380
15	AD	0.058824
16	AE	0.056604
17	D	0.050725
18	S	0.041667
19	X	0.027304
20	AA	0.014925
21	AH	0.000000
22	AB	0.000000
23	AC	0.000000
24	AG	0.000000
25	M	0.000000
26	AI	0.000000
27	O	0.000000
28	V	0.000000
29	T	0.000000
30	F	0.000000
31	Y	0.000000
32	R	0.000000
33	Q	0.000000
34	AJ	0.000000

[55]:
```
# plot retention vs company alias
sns.catplot(x="companyAlias", y="meanRetention", kind="bar",
            data=meanRetentionByCompany, height=5, aspect=2);
```

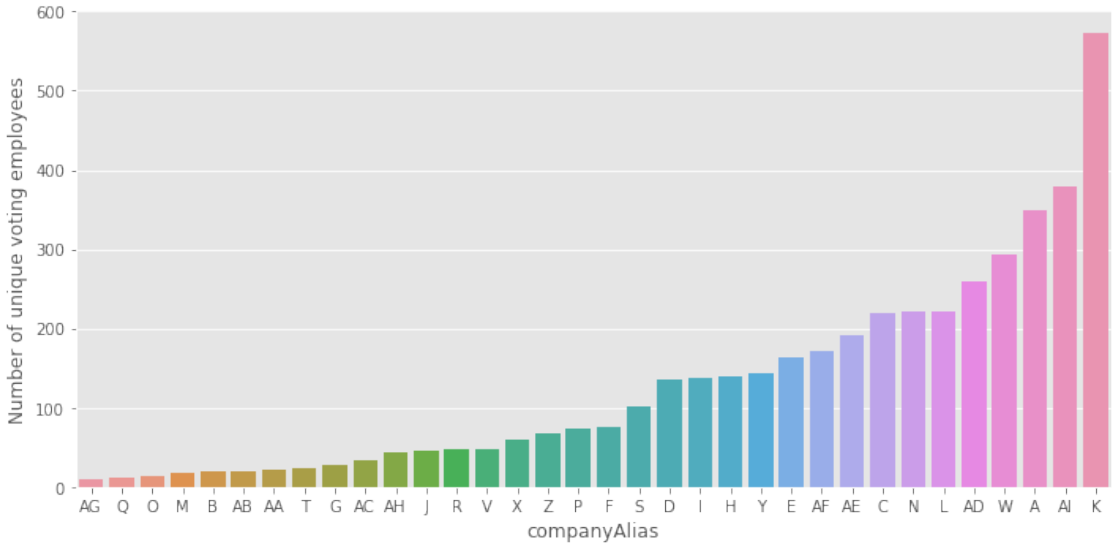


7.2 Plot the distribution of the number of unique employees contributing at least one vote for each company. (Hint: you will want to use the `votes_df` table.)

Answer. One possible solution is given below:

[56]:
```
uniq_employee_count_table = (votes_df.groupby('companyAlias')['employee']
                             .nunique()
                             .sort_values(ascending=True)
                             .reset_index(name='n_employees'))
sns.catplot(x='companyAlias', y='n_employees', kind='bar',
            data=uniq_employee_count_table, height=5, aspect=2);
plt.ylabel('Number of unique voting employees')
```

[56]:
```
Text(3.799999999999997, 0.5, 'Number of unique voting employees')
```

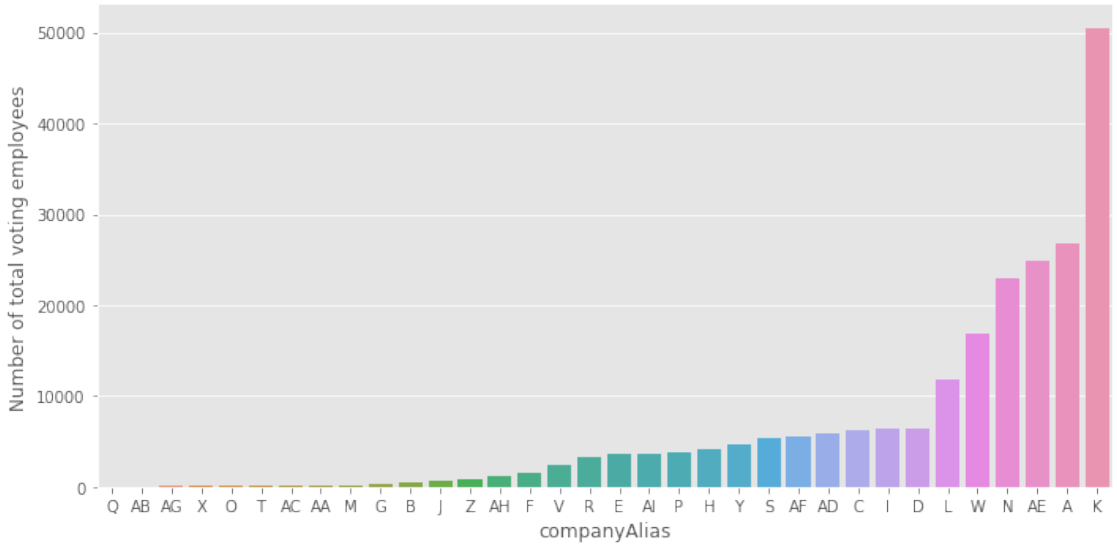


7.3 Plot the distribution of the number of unique employees contributing at least one vote for each company.

Answer. One possible solution is given below:

[57]:
```
employee_count_table = (votes_df.groupby('companyAlias')['employee']
                        .count()
                        .sort_values(ascending=True)
                        .reset_index(name='n_employees'))
sns.catplot(x='companyAlias', y='n_employees', kind='bar',
            data=employee_count_table, height=5, aspect=2);
plt.ylabel('Number of total voting employees')
```

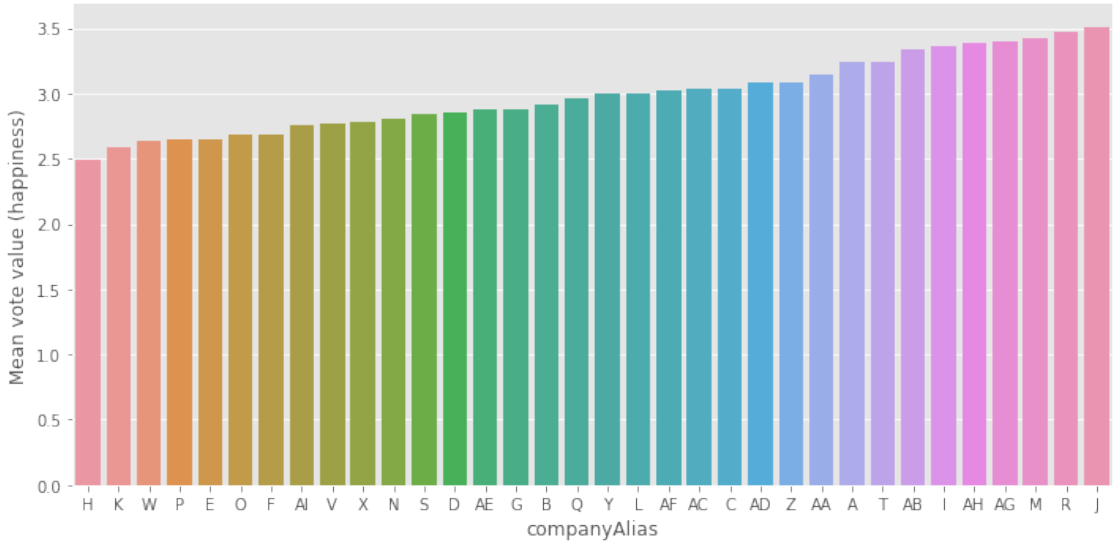
[57]:
```
Text(-8.700000000000003, 0.5, 'Number of total voting employees')
```



7.4 Plot the distribution of the average value of each vote per employee for each company.

Answer. One possible solution is given below:

[58]:
```
votes_count_table = (votes_df.groupby('companyAlias')['vote']
                     .mean()
                     .sort_values(ascending=True)
                     .reset_index(name='n_votes'))
sns.catplot(x='companyAlias', y='n_votes', kind='bar',
            data=votes_count_table, height=5, aspect=2);
plt.ylabel("Mean vote value (happiness)");
```



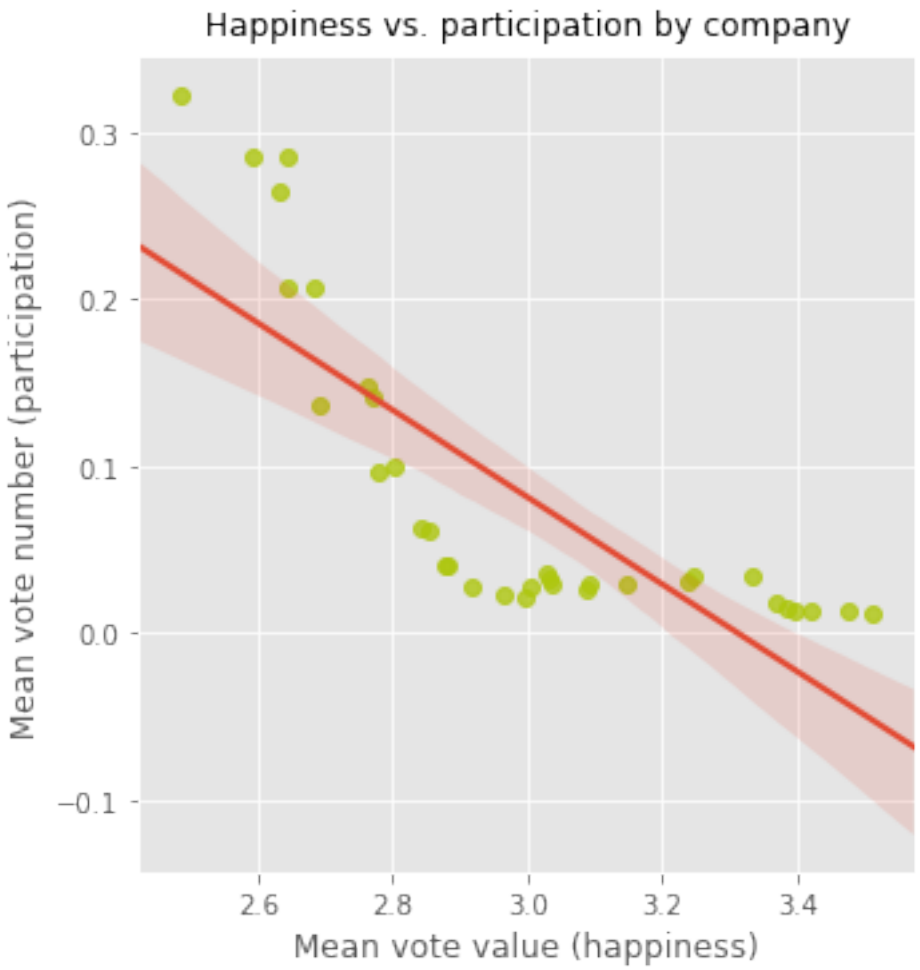
7.5 Now, put everything above together and determine how the mean vote per employee relates to the number of votes for each company.

Answer. One possible solution is shown below:

[59]:
```
# Determine mean votes per employee.
num   = uniq_employee_count_table.
 ↪sort_values(by=['companyAlias'])['n_employees']
denom = employee_count_table.sort_values(by=['companyAlias'])['n_employees']
meanVotesEmployee = num / denom
```

[60]:
```
# Put the two columns into a dataframe to make plotting easier.
data = pd.DataFrame()
data['meanHappiness'] = meanVotesEmployee
data['meanVotes'] = votes_count_table['n_votes']
```

[61]:
```
# Plot the data.
sns.lmplot(x = 'meanVotes', y = 'meanHappiness', data=data, scatter_kws =␣
 ↪{'color': (174/255,199/255,14/255)})
plt.title("Happiness vs. participation by company", fontsize=12,␣
 ↪verticalalignment='bottom');
plt.xlabel("Mean vote value (happiness)");
plt.ylabel("Mean vote number (participation)");
```



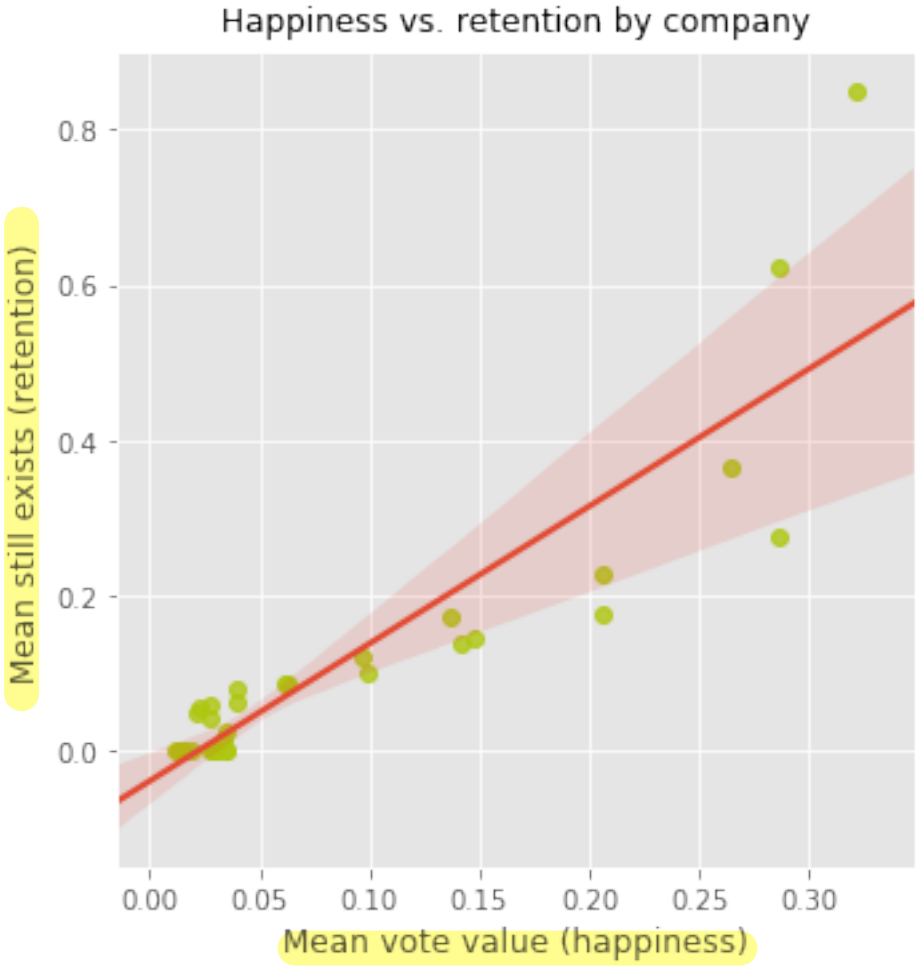
Note that the graph above suggests that there is a negative relationship between participation and happiness. To be specific, it means that the employees that participate the most in this feedback app tend to have the lowest average votes. This suggests that participation on the app might be indicative of unhappier employees.

7.6 Put everthing together to answer the original question:

Answer. One possible solution is shown below:

[62]:
```
# Put the two columns into a dataframe to make plotting easier.
data = pd.DataFrame()
data['companyAlias']  = meanRetentionByCompany.
 ↪sort_values(by=['companyAlias'])['companyAlias']
data['meanHappiness'] = meanVotesEmployee
data['meanRetention'] = meanRetentionByCompany.
 ↪sort_values(by=['companyAlias'])['meanRetention']
```

```
# Plot the data.
sns.lmplot(x = 'meanHappiness', y = 'meanRetention', data=data, scatter_kws =␣
 ↪{'color': (174/255,199/255,14/255)})
plt.title("Happiness vs. retention by company", fontsize=12,␣
 ↪verticalalignment='bottom');
plt.xlabel("Mean vote value (happiness)");
plt.ylabel("Mean still exists (retention)");
```



There is some evidence to suggest there is a positive associative relationship between employee happiness and retention. Furthermore, our analysis has shown that vote value, liking frequency, comment length, and app engagement are all different features we can explore to measure employee happiness. Our exploratory data analysis seems to suggest they are related in the ways that we would expect.