Statistical Sentiment-Analysis for Survey Data using Python

Survey's play the main part when receiving client feedback on a particular product or service one offers for the public.

Dilan Jayasekara Follow
Aug 13, 2019 · 9 min read ★

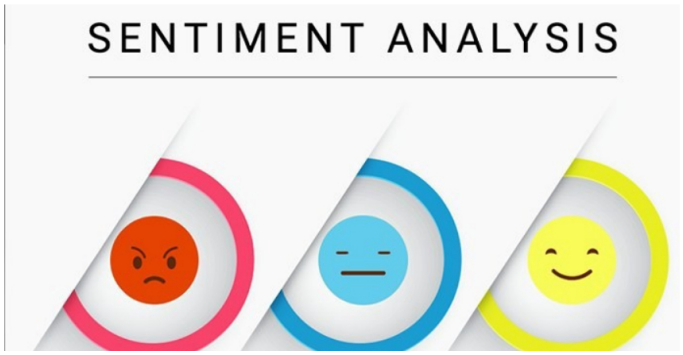
Photo by Helloquence on Unsplash

Are we getting too many negative feedbacks? Why? How can we fix issues? What are we doing well and what have we improved after some time? What are the most key issues to comprehend?

Assessing responses from customer surveys and creating a report that will give us the answers to these questions is easier said than done. It may take us hours, or even days to go through all responses and find the root of a problem.

That's when **Sentiment Analysis** comes up and it can help us make sense of survey responses to automatically answer questions such as:

- How many negative responses did we receive?
- What aspects of our product/service do customers love?
- What aspects of our product/service do customers hate?
- Has a particular product feature improved?



Sentiment can be Negative, Neutral or Positive

Let's see how we can use a simple Sentiment Analysis for Survey Data and get answers for the above questions I've mentioned.

. . .

Using a python, Our analysis was completed on the **qualitative feedback provided by clients**, I have a CSV file of responses to the question "What did we do well" to a service I provide at my business.

Sentiment Analysis consists of two parts; **Subjectivity and Polarity**. Simply, Polarity and Subjectivity can be explained as follows:

- **Polarity —** It simply means emotions expressed in a sentence, across a range of negative, to positive.
- **Subjectivity —** Subjective sentence expresses some personal feelings, views, or beliefs.

. . .

So here's a summary of what I'm going to do in the next 10–15 Minutes.

1. I'm going to create a new CSV file by reading my existing client response file then generating Subjectivity and Polarity for each client feedback.
2. Then I'll plot some graphs to get an idea of how my clients are feeling. For this section, I'll be creating

- A **Box Plot**
- **Scatter Plot**
- **Covariance and Correlation** using Scatter Graph
- **Polarity Distribution**
- Polarity **Density Distribution**

3. After that, I'm eager to find out the **Frequent Words** they used while giving feedback.

4. Then put them into a **word cloud**

Alright ladies and gentlemen, Let's go!

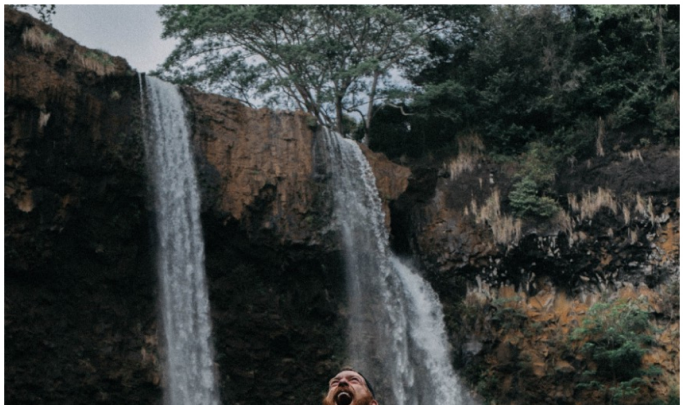


Photo by Jakob Owens on Unsplash

Jump into a new Jupyter notebook project and, first things first.

```
import pandas as pd            //Pandas
from textblob import TextBlob //For Sentiment Anlysis
from itertools import islice //Will Explain Later
```

Read CSV File (Q7_Text.CSV) in to a DataFrame object.

```
df_survey_data = pd.read_csv("Q7_Text.csv")
```

As I said before, I'm generating a new CSV file so I'll create a list object and store my column names for my new file.

```
COLS = ['date','text', 'sentiment','subjectivity','polarity']
```

Then I'm using these columns for a brand new dataframe called df.

```
df = pd.DataFrame(columns=COLS)
```

Now comes a tricky part, I want to read each line of my Q7_Text.Csv and calculate sentiment then store the result in a new CSV file.

I want to **change the starting index of *iterrows()*** method that's why I've imported ***itertools.islice*** at the beginning of my code. Here's where I grabbed this trick.

```
for index, row in islice(df_survey_data.iterrows(), 0, None):

    new_entry = []
    text_lower=to_lower(row['Responses'])
    blob = TextBlob(text_lower)
    sentiment = blob.sentiment

    polarity = sentiment.polarity
    subjectivity = sentiment.subjectivity

    new_entry += [row['Response Date'],text_lower,sentiment,subjectivity,polarity]

    single_survey_sentimet_df = pd.DataFrame([new_entry], columns=COLS)
    df = df.append(single_survey_sentimet_df, ignore_index=True)

```

```
16  df.to_csv('Q7_Text_Sentiment_Values.csv', mode='w', columns=COLS, index=False, encoding="utf-8")
```

forLoop.py hosted with ❤ by GitHub view raw

That piece of code has created me a nice little file with all the sentiments.

Let's see what I have with a ***df.head()***

```
In [67]: df.head()
```

Out[67]:

	date	text	sentiment	subjectivity	polarity
0	Jun 27 2019 04:28 PM	self empowering.	(0.0, 0.0)	0.0000	0.00
1	Jun 27 2019 03:31 PM	i can express my feeling and get advice.	(0.0, 0.0)	0.0000	0.00
2	Jun 27 2019 03:28 PM	being able to talk and discuss my issues with ...	(0.25, 0.5125)	0.5125	0.25
3	Jun 27 2019 03:10 PM	talking about any interest, goals etc. excelle...	(1.0, 1.0)	1.0000	1.00
4	Jun 27 2019 02:44 PM	that i was able to implement changes that i wo...	(0.5, 0.625)	0.6250	0.50

Figure 1.0

Let's run a ***df.describe()*** for a summary.

```
In [68]: df.describe()
```

Out[68]:

	subjectivity	polarity
count	375.000000	375.000000
mean	0.335306	0.178592
std	0.310195	0.253179
min	0.000000	-0.500000
25%	0.000000	0.000000
50%	0.346058	0.000000
75%	0.600000	0.375000
max	1.000000	1.000000

Figure 2.0

> *So my program has confirmed to me that all the 375 records are there and gave me a* ***mean polarity of 0.178****, which is good that means as an average, most people are in between* ***neutral to positive*** *with our services.*

And as you can see the 50% Value which means the **median is zero (0)**. Out of 375 records, I'm getting the median value as zero can be coincidence or **maybe I have too many neutral feedbacks which will affect the accuracy of our analysis**. I need to clear my head with that doubt so I'm going to **Find rows with non zero values in a subset of columns in pandas dataframe.**

To accomplish that I'm using an alternative solution which uses select_dtypes() method:

```
dffilter = df.loc[(df.loc[:, df.dtypes != object] != 0).any(1)]
```

Take a sneak peek at what **dffilter** is made of.

Summary after Preprocessing : dffilter

```
In [37]: dffilter = df.loc[(df.loc[:, df.dtypes != object] != 0).any(1)]
         dffilter.describe()
```

Out[37]:

	subjectivity	polarity
count	236.000000	236.000000
mean	0.532795	0.283780
std	0.217866	0.268397
min	0.000000	-0.500000
25%	0.375000	0.077313
50%	0.527083	0.254167
75%	0.650000	0.500000
max	1.000000	1.000000

Figure 2.1

> *HA! Now I can clearly see that my doubt is a reality! New count is equal to* ***236****. So the* ***delta value of df and dffilter is 139.*** **delta = count(df) — count(dffilter),**

Which means **I've got 139 neutral feedbacks mate**, no wonder I got '0' as my median value. Not just that, Mean polarity has risen up to 0.28 and the median is no more 0. (Which clearly explains that those 139 Neutrals were directly affected to my analysis)

Let me take you back to my College days.

Photo by Nathan Dumlao on Unsplash

My friend **Farzad** has asked this question "How can I transfer a data including lots of zero (precipitation) to a **normal distribution**?"

Mr. Stephen Politzer-Ahles has answered that "I don't think you can…"

. . .

So as Mr. Stephen said and I agree that the best option would be to separate the analyses.

As a result, Out of 375 client feedbacks, our analysis has detected 139 Neutral polarities (polarity is 0) so those records were cut out for more accurate results. Hence, these results are based on the remaining 236 feedbacks for the question "**what did we do well".**

PLOTS

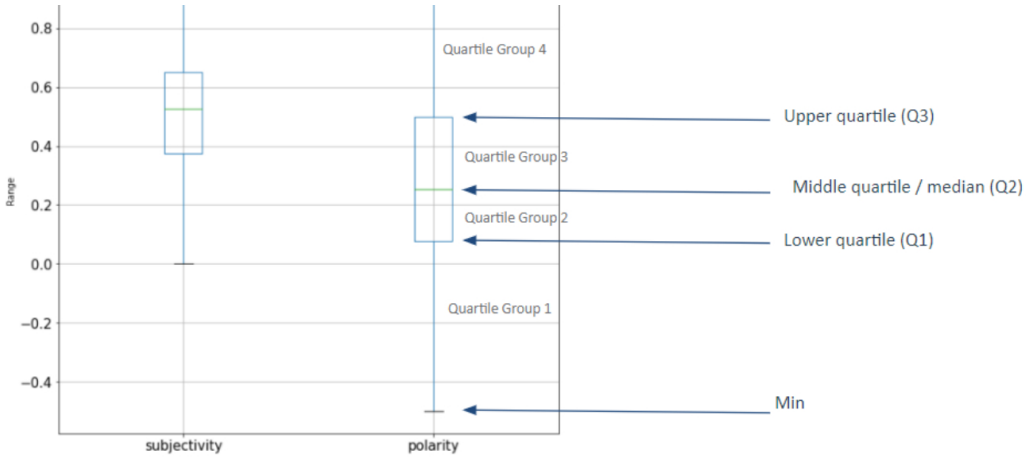
1. BOX PLOT

Let's create a box plot as the first step of our plot analysis. (Remember, We are using **dffilter** dataframe from now onwards)

```
#boxplot for dffilter

boxplot = dffilter.boxplot(column=['subjectivity','polarity'],
                           fontsize = 15,grid = True, vert=True,figsize=(10,10,))
plt.ylabel('Range')
```

BoxPlot.py hosted with ❤ by GitHub — view raw

Python code for the box plotting



Basically, **The Boxplot is a graphical representation of the describe() method**. I'll make a little table here with the values I've got for dffilter.describe() and you'll make a comparison against the two. Except for count, mean and standard deviation all the outliers are graphically represented by the boxPlot.

	Subjectivity	Polarity
Count	236.000000	236.00000
Mean	0.532795	0.283780
Std Deviation	0.217866	0.268397
Min	0.000000	-0.500000
Q1	0.375000	0.077313
Q2 (Median)	0.527083	0.254167
Q3	0.650000	0.500000
Max	1.000000	1.000000

result of describe() for dffilter

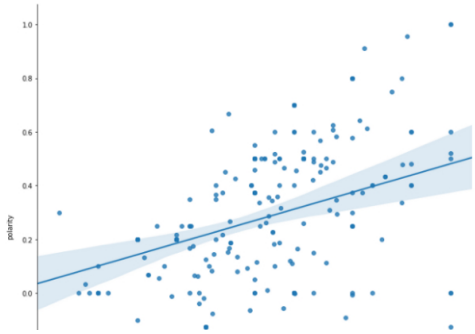
. . .

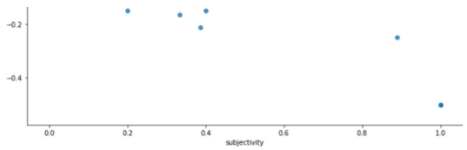
2. SCATTER PLOT

```
#scatter for dffilter

sns.lmplot(x='subjectivity',y='polarity',data=dffilter,fit_reg=True,scatter=True, height=10,palet
```

scatterPlot.py hosted with ❤ by GitHub view raw

Python Code for the Scatter Diagram (Scatter Plot)





Scatter Plot for **dffilter**

The scatter diagram is known by many names, such as scatter plot, scatter graph, and correlation chart.

The scatter diagram is used to find the **covariance and correlation between two variables**. This diagram helps you determine how closely the two variables are related. After determining the correlation between the variables, you can then predict the behavior of the dependent variable based on the measure of the independent variable.

Covariance/Correlation

$$\text{Cov}(X, Y) = \frac{\sum(X_i - \overline{X})(Y_j - \overline{Y})}{n - 1} \qquad r = \frac{\sum_i (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_i (x_i - \overline{x})^2}\sqrt{\sum_i (y_i - \overline{y})^2}}$$

Covariance/Correlation Equations is Statistics

I'll briefly explain what Covariance and Correlation means:

Covariance

Covariance indicates how two variables are related (In our case, It's Subjectivity and Polarity). A positive covariance means the variables are positively related and they move in the same direction, while a negative covariance means the variables are inversely related and they move in opposite directions.

Correlation

Correlation is another way to determine how two variables are related. In addition to telling you whether variables are positively or inversely related, correlation also tells you the degree to which the variables tend to move together.

Let's calculate Covariance and Correlation with Python!

```python
#covariance and correlation for dffilter
# calculate the covariance between two variables

from numpy.random import randn
from numpy.random import seed
from numpy import cov
from scipy.stats import pearsonr

# prepare data
data1 = dffilter['subjectivity']
data2 = data1 + dffilter['polarity']
# calculate covariance matrix
covariance = cov(data1, data2)
print(covariance)

corr, _ = pearsonr(data1, data2)
print('Pearsons correlation: %.5f' % corr)
```

covarianceCorrelation.py hosted with ♥ by GitHub view raw

Python code for Cov/Correlation

```
#Result of Covariance

[[0.04746545 0.06750151]
 [0.06750151 0.15957442]]
```

- **The covariance between the two variables is 0.06750151. We can**

see that it is positive, suggesting the variables change in the same direction as we expect.

```
Pearsons correlation: 0.77561
```

- **We can see that the two variables are positively correlated and that the correlation is 0.775. This suggests a high level of correlation, e.g. a value above 0.5 and close to 1.0.**

So, By looking at our Scatter Diagram, What can we say about these two variables?

- The plot shows a positive correlation between Subjectivity and Polarity.

> Meaning, as subjectivity increase, the polarity in the response increase too, Or in other words, the more strong feelings are expressed, the more the overall comment is subjective.

. . .

3. POLARITY DISTRIBUTION & DENSITY CURVE

The polarity distribution describes all the values of the variable Polarity within a given range. This range will be bounded between the minimum and maximum possible values (-1 to +1 in our case). By looking at the distribution we can identify how the polarity is distributed among clients and key factors as mean, median and standard deviation.

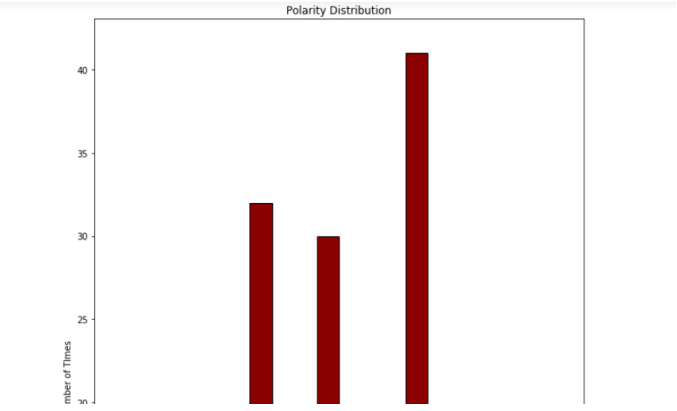
Most commonly distribution is Normal Distribution or "Bell Curve".The important thing to note about a normal distribution is the curve is concentrated in the center and decreases on either side.

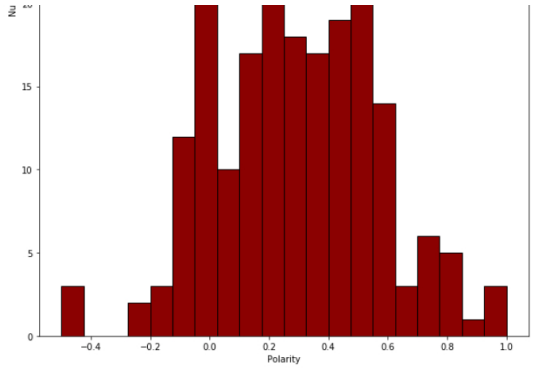
Let's jump up to our python code to see what's behind the code.

```python
#Polarity Distribution for dffilter

plt.hist(dffilter['polarity'], color = 'darkred', edgecolor = 'black', density=False,
         bins = int(30))
plt.title('Polarity Distribution')
plt.xlabel("Polarity")
plt.ylabel("Number of TImes")

from pylab import rcParams
rcParams['figure.figsize'] = 10,15
```

Distribution.py hosted with ❤ by GitHub view raw

Python Code for Polarity Distribution





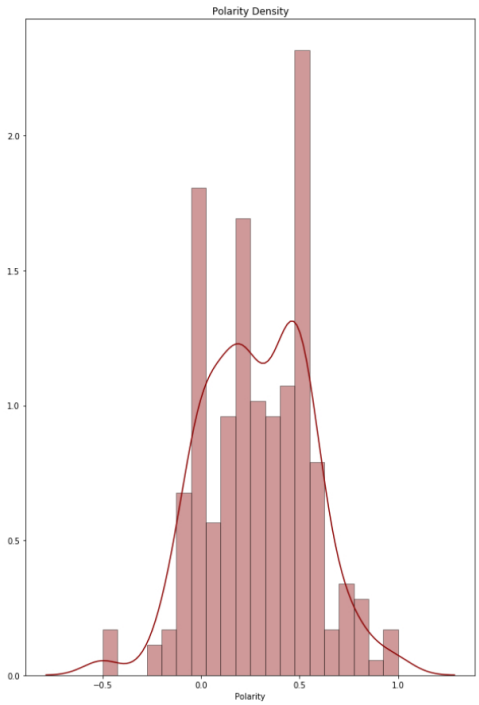
Polarity Distribution

Here's how to enable the density curve on it.

```
sns.distplot(dffilter['polarity'], hist=True, kde=True,
             bins=int(30), color = 'darkred',
             hist_kws={'edgecolor':'black'},axlabel ='Polarity')
plt.title('Polarity Density')

from pylab import rcParams
rcParams['figure.figsize'] = 10,15
```

density.py hosted with ❤ by GitHub view raw

Python Code for Density Curve



Polarity Density Curve

. . .

4. FREQUENT WORDS

Frequently we want to know which words are the most common from survey s since we are looking for some patterns. Given the data set, we can find k number of most frequent words with Natural Language Processing (NLP) using Python.

In natural language processing, useless words (data), are referred to as **Stop Words**: A **stop word** is a commonly used **word** (such as "the", "a", "an", "in") that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query. So, We've generated 10 most commonly used words in our

search query. So, we've generated 10 most commonly used words in our survey.

If you are new to Python **NLTK**, Click here to read more about it.

Let's download the nltk package.

```
import nltk
nltk.download()
```

store all the stop words in 'English' into a list called **stopwords**.

```
stopwords = nltk.corpus.stopwords.words('english')
```

With the help of regular expression, I'm adding more symbols that might appear as frequent because I want them to be ignored.

```
RE_stopwords = r'\b(?:{})\b'.format('|'.join(stopwords))

words = (df.text
           .str.lower()
           .replace([r'\|',r'\&',r'\-',r'\.',r'\,',r'\'',
RE_stopwords], [' ', '','','','','',''], regex=True)
           .str.cat(sep=' ')
           .split()
)
```

Ok Cool, It's almost over man, Thanks for hanging out this further :) Let's kick off this final stage:

```
from collections import Counter

# generate DF out of Counter
rslt = pd.DataFrame(Counter(words).most_common(10),
                    columns=['Word', 'Frequency']).set_index('Word')
rslt
```

wordcloud1.py hosted with ♥ by GitHub view raw

Python Code for Frequent Words

This will give us the below result set which are the top 10 commonly used words across our clients.

Out[223]:

Word	Frequency
information	50
understanding	39
help	37
helpful	34
children	30
able	26
child	26
support	25
family	24
relationship	23

Frequent Words TOP 10

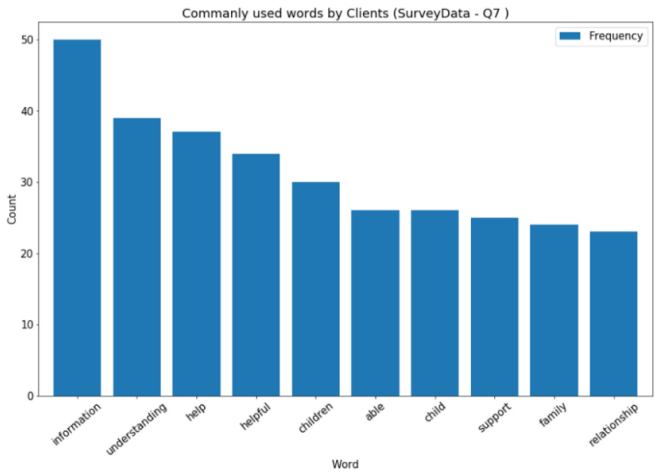
Let's plot them in a bar chart and a pie chart.

```
rslt_wordcloud = pd.DataFrame(Counter(words).most_common(100),
                    columns=['Word', 'Frequency'])
#BAR CHART
rslt.plot.bar(rot=40, figsize=(16,10), width=0.8,colormap='tab10')
plt.title("Commanly used words by Clients (SurveyData - Q7 )")
plt.ylabel("Count")

from pylab import rcParams
rcParams['figure.figsize'] = 10,15
```

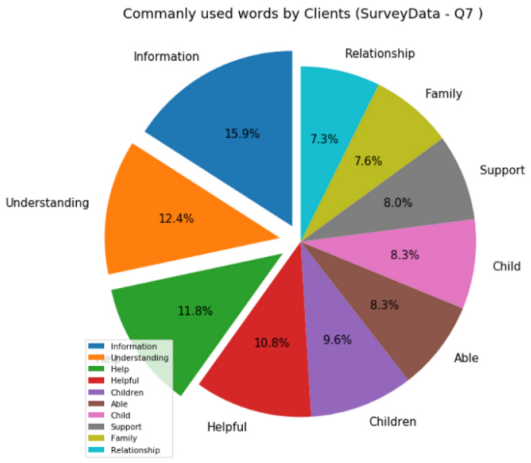
```
#PIE CHART

explode = (0.1, 0.12, 0.122, 0,0,0,0,0,0,0)  # explode 1st slice
labels=['Information',
        'Understanding',
        'Help',
        'Helpful',
        'Children',
        'Able',
        'Child',
        'Support',
        'Family',
        'Relationship',]

plt.pie(rslt['Frequency'], explode=explode,labels =labels , autopct='%1.1f%%',
        shadow=False, startangle=90)
plt.legend( labels, loc='lower left',fontsize='x-small',markerfirst = True)
plt.tight_layout()
plt.title(' Commanly used words by Clients (SurveyData - Q7 )')
plt.show()

import matplotlib as mpl
mpl.rcParams['font.size'] = 15.0
```

wordcloud2charts.py hosted with ❤ by GitHub view raw

Python Code for Visualization of Most Commonly used words



Bar graph of Frequent Words



Pie Chart of Frequent Words

5. WORDCLOUD

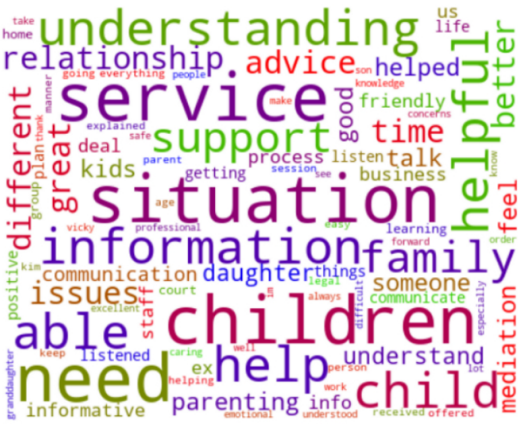
Many times you might have seen a cloud filled with lots of words in different sizes, which represent the frequency or the importance of each word. This is called Tag Cloud or **WordCloud**.

```
import numpy as np
import pandas as pd
from os import path
```

```
from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
import random

wordcloud = WordCloud(max_font_size=60, max_words=100, width=480, height=380,colormap="brg",
                      background_color="white").generate(' '.join(rslt_wordcloud['Word']))

plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.figure(figsize=[10,10])
plt.show()
```

wordclpudfinal.py hosted with ❤ by GitHub view raw

Python Code for WordCloud



WordCloud generated using Frequent Words

. . .

After considering the

- Box plot values
- Relation of scatter diagram
- Covariance and Correlation values
- Polarity distribution and density shape of our data
- Frequently used words and WordCloud

We could say that

- overall polarity demonstrates a positive value.
- Overall subjectivity is a positive value.
- Subjectivity and Polarity are moving towards the same direction as one increases, the other is likely to increase as well.
- Or in other words, the more strong feelings are expressed, the more the overall comment is subjective and vice versa.

To conclude the article, Our analysis was completed on the qualitative feedback provided by clients. Responses to the question **"what did we do well"** demonstrated a positive polarity (mean 0.283780 and standard deviation 0.268397 where polarity can range from negative -1.0 to positive 1.0) with a strong level of subjectivity in responses (mean 0.532795 and standard deviation 0.217866 where 0.0 is very objective and 1.0 is very subjective). This information proves us that "**most responses to this question were of a positive, subjective nature**."

. . .

Keen on getting to know me and my work? Click here for more!

Data Science | Sentiment Analysis | Statistics | Corrections | Programming

WRITTEN BY

Dilan Jayasekara

Follow

3x Featured Writer @TDataScience with 600K+ Views 🔥 | | Developer | Poetry & Content Writer | Software & Data Consultant: www.djayasekara.com LK AU

Towards Data Science

Follow

A Medium publication sharing concepts, ideas, and codes.