Text Analysis Project

Problem Statement

The CMO Offboarding team has been in existence since late 2016. I want to determine how the rest of the business feels about the offboarding process and how the team is performing. I am going achieve this by running a sentiment analysis model based on the e-mails taken from the Offboarding shared mailbox. This mailbox is where all requests and queries relating to the offboarding process is sent to. This is important as the offboarding process has gone undergone huge change and refinement over past three years. New risk checks and additional communication to the business are two areas we have been focusing on in order to make the process as seamless as possible. I hope the results will be able to show that the team, and our standing with the business has improved.

Hypothesis

- The offboarding process and team is viewed in a positive way by colleagues in UBS.

Overview

- I am going to take a cut of the emails sent to the Offboarding mailbox and try to analyse the content to see if we can derive the current sentiment for the team's performance and the process.
- Alongside the sentiment analysis I am going to produce a 'bag of worlds' to gain insight into the common words being sent to the inbox and see if we can determine any meaningful conclusions from this. Common phrases and words being sent should be able to give insight into the most common subjects of queries received.

Sourcing the data

- This data has been taken from the Offboarding shared mailbox.
- It was exported to Microsoft Access before being saved as a .csv file for analysis.
- There will be Client Identifying Data (CID) in this project therefore I will be taking all required actions to mask this in order to comply with the UBS data guidelines.

Tasks

In order to complete this analysis successfully I will be carrying out the following tasks:

1. Import the libraries and data.

1. Clean the data
   I need to make sure the data I am using is in a suitable and useable format for analysis.

1. Bag of Words model
   This should provide insight into the most common phrases used the emails that the team's shared mailbox has received. We can make initial assumptions on the business sentiment based on this.

1. Sentiment analysis
   I am going to use textblob to assign a sentiment score for the body of text in the emails.

1. Text Classification
   I am going to assign positive or negative categorisation to the emails based on the sentiment analysis score derived from textblob. This will give me the basis to report to senior management the business' opinion on the offboarding process.

1. Evaluation and conclusion
   What was the outcome? What actions can I take from this? What would the future steps be to improve this analysis?

Importing the required libraries

In [1]:

```
# data manipulation libaries
import pandas as pd
import numpy as np

# visualiation libaries
import seaborn as sns
import matplotlib.pyplot as plt
from wordcloud import WordCloud
import cufflinks as cf
import plotly.graph_objs as go
from plotly.plotly import iplot

# string manipulation libraries
import re
import string

# text pre-processing libaries
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer

# sentiment analysis
from textblob import TextBlob

# vectorizer
from sklearn.feature_extraction.text import CountVectorizer

# splitting the model for train test
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

# naive bayes - machine learning algo of of choice for text analysis
from sklearn.naive_bayes import MultinomialNB

# machine learning model evaluation
from sklearn.metrics import accuracy_score, classification_report

# dependices for NLTLK
#nltk.download('punkt')
#nltk.download('stopwords')
#nltk.download('wordnet')
```

Importing the data and preprosessing

In [2]:

```
# importing the data
data = pd.read_csv('offboarding_inbox.csv',encoding='ANSI')
```

In [67]:

```
# checking the data has imported correctly - removed the body of text due to CID issues
data.head(0)
```

Out[67]:

	importance	icon	priority	subject	from	message_to_me	message_cc_to_me	sender_nam

In [4]:

```
# changing the column headers to make them easier to work with
data.columns = data.columns.str.strip().str.lower().str.replace(' ','_').str.replace(
'(', '').str.replace(')', '')
```

In [10]:

```
# checking the column headings have been amended correctly - removed the body of text d
ue to CID issues
data.head(0)
```

Out[10]:

	importance	icon	priority	subject	from	message_to_me	message_cc_to_me	sender_nam

In [6]:

```
# checking the shape of the data
data.shape
```

Out[6]:

```
(1851, 20)
```

- The shape tells me that all the data appears to have been imported correctly.
- I am going to create a new varible with just the body of the e-mails so we analysis the body of the text properly

In [11]:

```
inbox = data['contents']

inbox.head()
```

Out[11]:

```
0    Hi Markus,\r\n\r\nClient you are requesting to...
1    Hi Anna,\r\n\r\nThanks, are you sure this is t...
2    Hi Markus,\r\n\r\nThis is the same client:\r\n...
3    Hi Anna,\r\n\r\nCan I check if this reactivati...
4    Approved\r\nLinda\r\n\r\n\r\n\r\n\r\nFrom: Chi...
Name: contents, dtype: object
```

- I can see from the above that there are a number of formatting issues I will need to clean up.
- When e-mail text is moved to new line in the body of the text the notebook is interpreting his as "/r" or "/n".

In [13]:

```
# taking the first e-mail as a sample to look into what potential preprosessing is requ
ired
sample = inbox[0]

#sample
```

In [68]:

```
# tokenzine the sample e-mail to break it down into a list of works

# word_tokenize(sample)  - commented out for CID reasons
```

In [69]:

```
# sent_tokenise - transforms the string into a list of sentance, where each sentance is
a token

# sent_tokenize(sample) -  - commented out for CID reasons
```

- I am going to try and clean up the sample e-mail before applying this to the entire population of data.
- I am going to use regular expression to remove unnecessary punctuation.
- I am also going to use stopwords to remove words such as "the", "a", and "an" as these tell us nothing about sentiment and are used to define and qualify in the English language.

In [16]:

```
# define the clean function to apply to the text
def clean(sample_email):

    sample_email = re.sub(r'[#|@|-|,|?|!|<|>|\|/|=|;|(|)|+|]',r'',sample_email)

    sample_email = re.sub('\s',' ', sample_email)

    sample_email = sample_email.lower()

    sample_email = WordNetLemmatizer().lemmatize(sample_email)

    words = word_tokenize(sample_email)

    words = [w for w in words if w not in stopwords.words('english')]

    text = ' '.join(words)

    return text
```

- With the function defined I can now apply this to my set of emails.

In [17]:

```
emails = inbox.apply(lambda x: clean(x))
```

In [20]:

```
# emails.head() - commented out for CID reasons
```

- The cleaning has worked successfully.
- I am going to move onto creating a bag of words to see if I can generate any insight from the contents.

Modeling

Bag of Words Model

- I am going to take 2,000 words as this should cover the useable repeatable words, alongside any noise that consistently repeats.

In [19]:

```
# we are going to specify that min_df parameter to be 0.01. This means the terms we are
checking need to be used in 1% of our tweets.
# this prevents analysis on terms that only appear rarely, thus unecessarly taking up r
esources

# we specify an ngram_range of 1. This means that we're only looking for words -- an ng
ram_range of (1,2)
#would include both words (lenth = 1) and phrases or combinations of words of lengh = 2
vector = CountVectorizer(max_features = 2000, min_df = 0.01, ngram_range = (1,1))

#use the fit_transform() function to apply the above to our emails
bag_of_words = vector.fit_transform(emails)

bag_of_words
```

Out[19]:

```
<1851x2000 sparse matrix of type '<class 'numpy.int64'>'
	with 310119 stored elements in Compressed Sparse Row format>
```

- I can see the sparse matrix is 1,851 by 2,000 so the vectoring has worked.
- I am going to list the all the individual terms in this sparse matrix.

In [22]:

```
#list the unique terms
#vector.get_feature_names() -
```

Visualisation

- Visualisation of the word bag should enable further insight.

In [24]:

```
# sum of occurences of each terms

sum_of_words = bag_of_words.sum(axis = 0)


#create list of tuples where each element reqresents the term and times it occurs

words_freq = [(word, sum_of_words[0, idx]) for word, idx in vector.vocabulary_.items()]

#sort by frequency

words_freq = sorted(words_freq, key = lambda x: x[1], reverse = True)

# words_freq - commented out for CID reasons
```

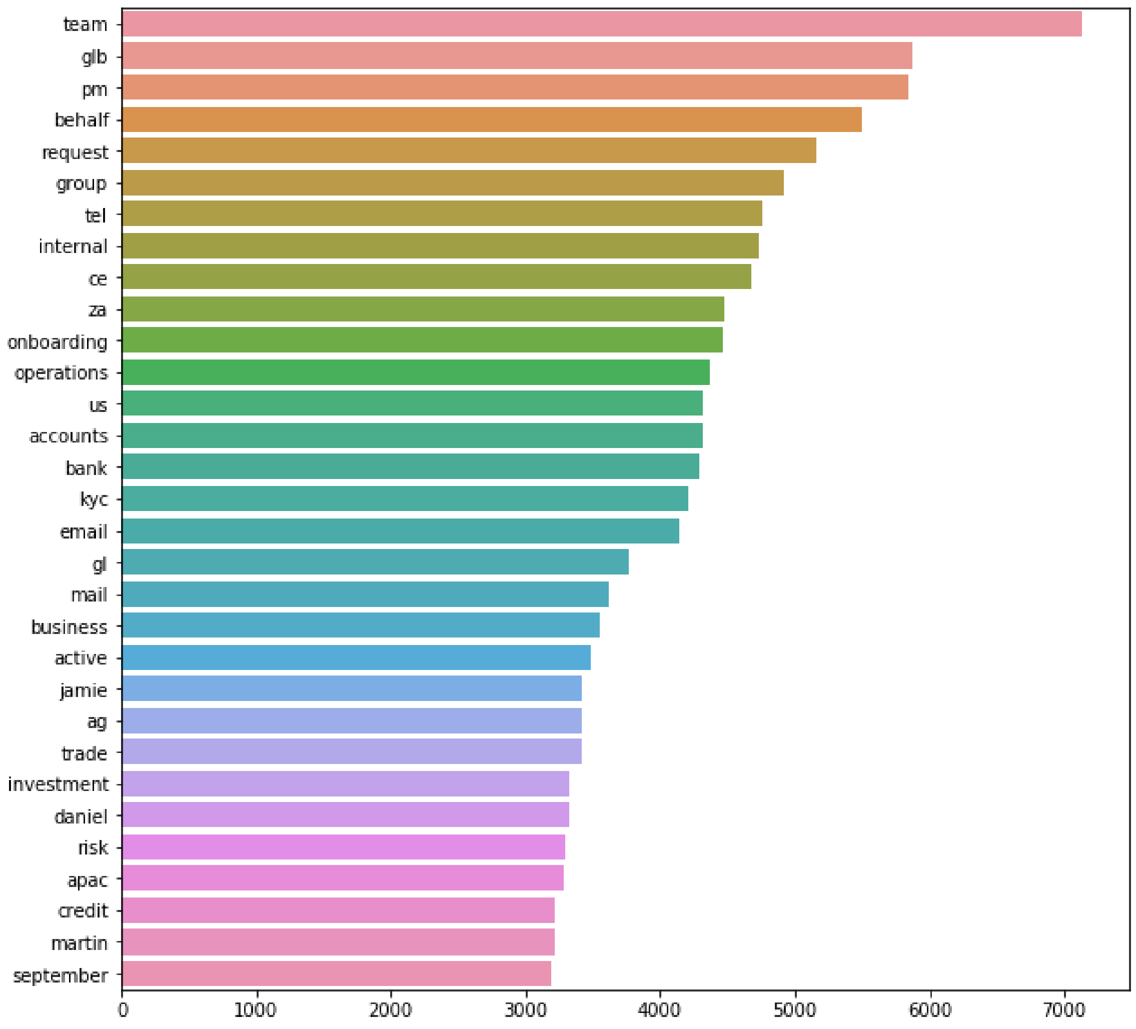
- I'm going to ignore the first 20 words.
- They are all standard words expected in e-mails and will make the visualisation useless.

In [25]:

```
top_words = words_freq[19:50]

word = []
count = []

for i, j in top_words:
    word.append(i)
    count.append(j)

plt.figure(figsize = (10, 10))

sns.barplot(x = count, y = word)
```

Out[25]:

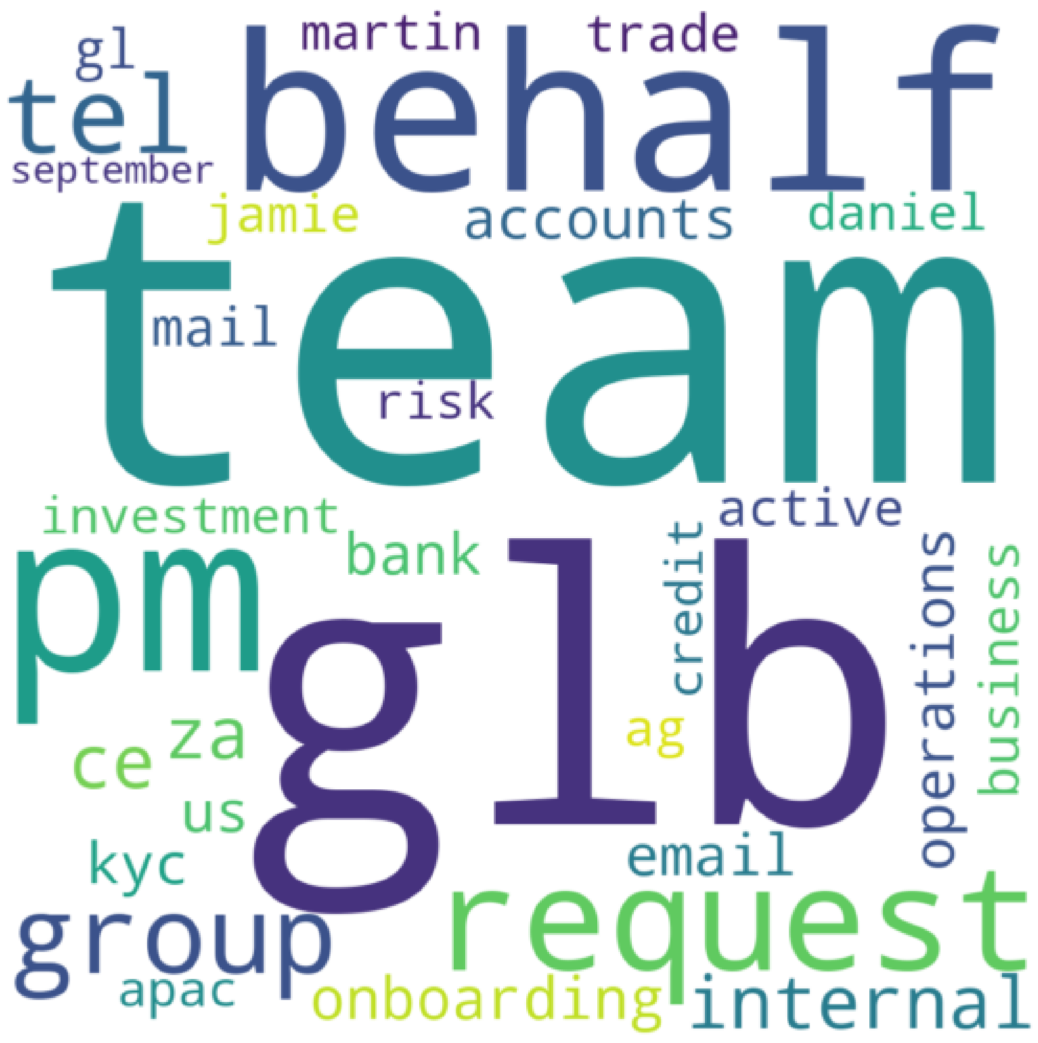
```
<matplotlib.axes._subplots.AxesSubplot at 0x1722d860>
```



- The words that are appearing in the above plot are quite generic as well as containing a number of names.

In [26]:

```
# using wordclouds to plot the most frequent words
words_dict = {}
for k, v in top_words:
    words_dict[k] = int(v)

#wordclouds library
wordcloud = WordCloud(width=1000, height=1000, background_color='white').generate_from_
frequencies(words_dict)

plt.figure(figsize=(20,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```



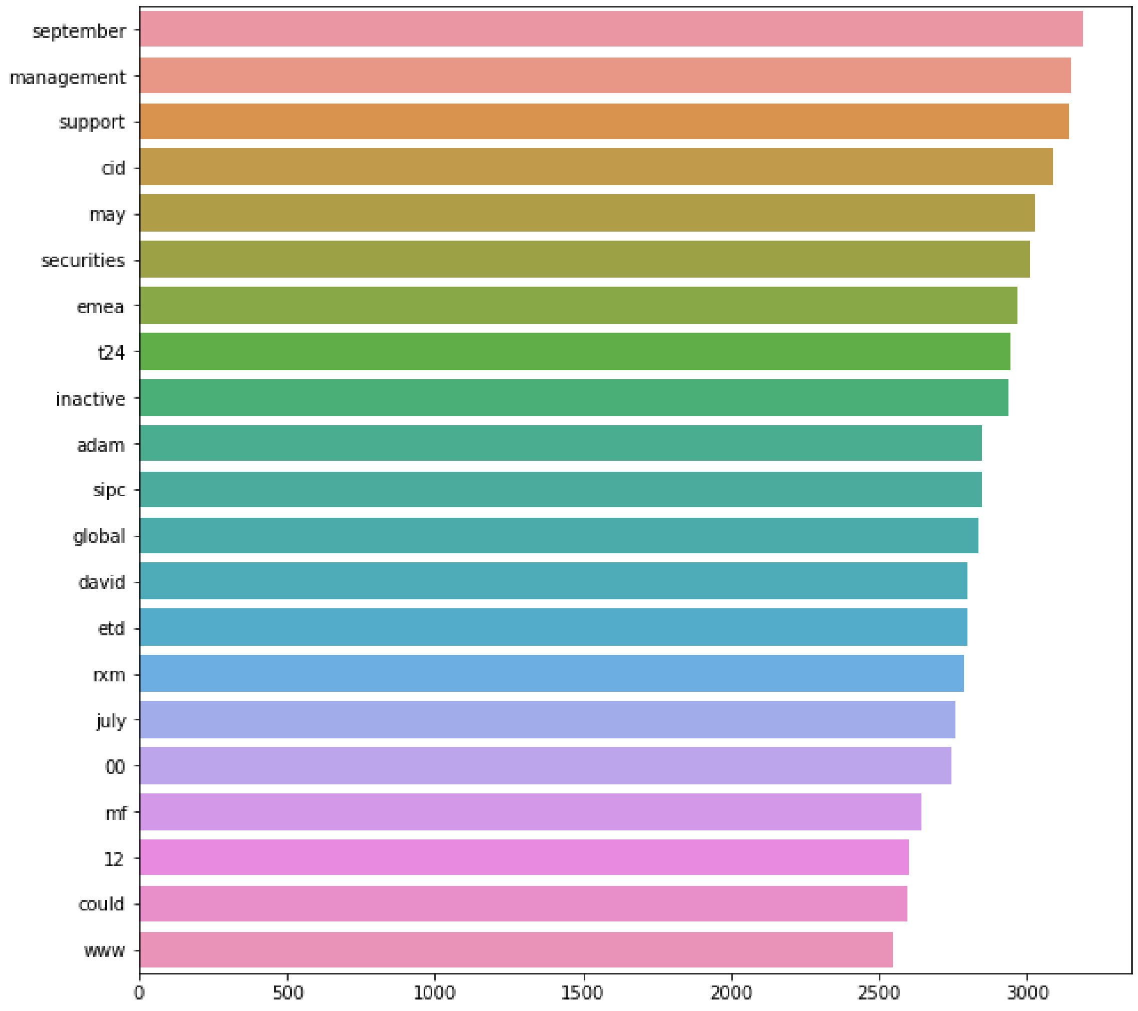
- I can see there are some words that I will need to be removed. This list includes:
  - The names of team members.
  - The word 'team'
- These are used in the majority of emails.
- I'm going to look at the list of top words and make a decision to proceed with the bag of words model if it does not product analysis of of value.
- Trying to identify every name would take too long and would likely not reap good enough rewards.

In [27]:

```
top_words = words_freq[49:70]

word = []
count = []

for i, j in top_words:
    word.append(i)
    count.append(j)

plt.figure(figsize = (10, 10))

sns.barplot(x = count, y = word)
```

Out[27]:

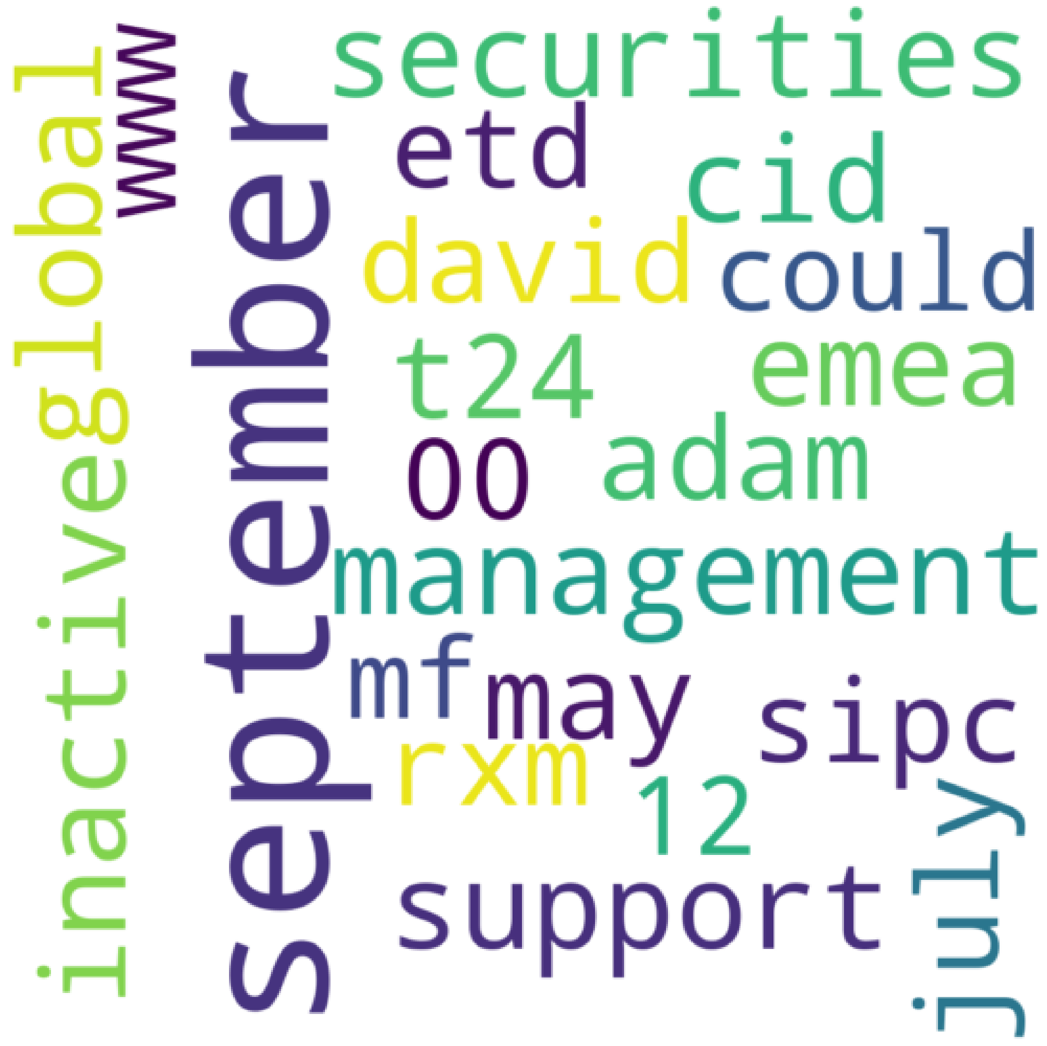
```
<matplotlib.axes._subplots.AxesSubplot at 0x173c1f60>
```



- I am getting more generic words such as "www" and "July".
- A word of note is "T24". This is a system used in Cash Equities. The team are currently helping clean-up client data in this system, hence why it's appearing regularly.
- More names are appearing in the plot.

In [28]:

```
# using wordclouds to plot the most frequent words
words_dict = {}
for k, v in top_words:
    words_dict[k] = int(v)

#wordclouds library
wordcloud = WordCloud(width=1000, height=1000, background_color='white').generate_from_
frequencies(words_dict)

plt.figure(figsize=(20,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```



- I can see from the above that the bag of words model has not been as useful as I originally thought. There is too much noise within the e-mails to completely tidy up. The one good result I can get from the bag of words is making senior management aware of how must effort has gone into helping clean-up the T24 system.
- Moving onto the sentiment analysis should be able to provide greater sight.
- I believe the problem using this technique stems from the data itself. There is so much noise in all the emails it's very difficult to remove. Team members, dates, hyperlinks and other such inputs which are important to the e-mail in context but out of contact make the word bag almost nonsensical.

Sentiment analysis

- **Textblob** can be used to assign sentiment popularity score to the emails I have extracted.
- This is ranged from -1 (for the most negative) to 1 (the most positive).
- Once it has done this it will calculate the sum total sentiment polarity for each email by averaging the score for the terms in the email.
- It uses a prepopulated library based of human input so we have to remain sceptical on this taking this score as fact.

In [29]:

```
# create a new variable for analysis
emails2 = emails.astype(str)
```

In [30]:

```
# create the sentiment score in text blob
sentiments = []

for emails2 in emails2:
    analysis = TextBlob(emails2)
    sentiments.append(analysis.sentiment.polarity)
```

In [70]:

```
# checking the new variable

# emails2  - commented out for CID reasons
```

In [33]:

```
# creating a new data frame to add the sentiment analysis to once completed
emails_df = pd.DataFrame(columns = ['emails'])
```

In [34]:

```
# checking the variables
emails_df
```

Out[34]:

emails

- I need to add the body of the emails to this variable now.

In [35]:

```
# add the body of the email text to the dataframe for analysis
emails_df['emails'] = emails
```

In [36]:

```
# check
emails_df.head(1)
```

Out[36]:

	emails
0	hi markus client requesting set neo offboarded...

- I am showing the first email only in order to make sure there is no CID shared.

In [37]:

```
# add this score to a dataframe of emails pandas
emails_df['sentiments'] = sentiments

emails_df.head(1)
```

Out[37]:

	emails	sentiments
0	hi markus client requesting set neo offboarded...	0.082517

- The sentiment score has been added to the variable successfully.

In [38]:

```
# find the most positive
positive = emails_df.sort_values(by = 'sentiments', ascending = False)
positive = positive.reset_index(drop = True)


# find the most negitive
negitive = emails_df.sort_values(by = 'sentiments', ascending = True)
negitive = negitive.reset_index(drop = True)
```

In [39]:

```
negitive.head(1)
```

Out[39]:

	emails	sentiments
0	hi zhengyang form beijing ops cs team could pl...	0.672222

In [40]:

```
positive.head(1)
```

Out[40]:

	emails	sentiments
0	hi zhengyang form beijing ops cs team could pl...	0.672222

In [41]:

```
# create a new column to categorise sentiment -- 'Negitive' if <0, 'Postiive' if >0 and 'Neutral' if == 0
emails_df['sentiment_category'] = np.where(emails_df['sentiments'] > 0, 'Positive',
                                           np.where(emails_df['sentiments'] == 0, 'Neutral', 'Negitive'))
```

In [42]:

```
# check the spread of sentimentality
emails_df['sentiment_category'].value_counts()
```

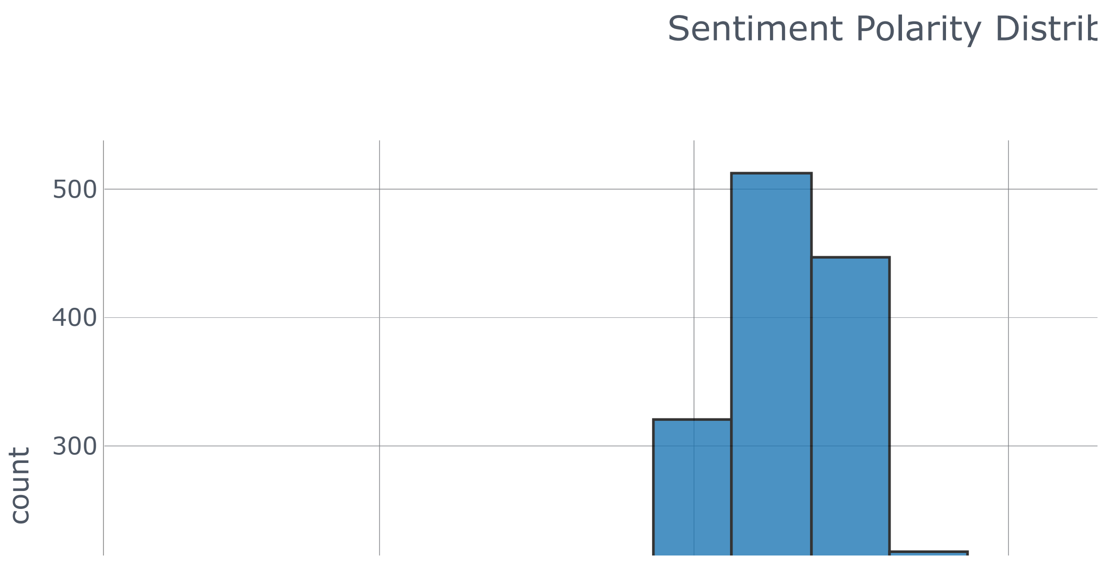
Out[42]:

```
Positive    1519
Negitive     310
Neutral       22
Name: sentiment_category, dtype: int64
```

These are very encouraging results. Visualising these results will make it clear.

In [43]:

```
#cufflinks offline library
cf.go_offline()

# plotting an interactive distribution plot of sentiment scores in plotly
emails_df['sentiments'].iplot(
    kind = 'hist',
    bins = 50,
    xTitle = 'polarity',
    theme = 'pearl',
    colorscale = 'plotly',
    linecolor = 'black',
    yTitle = 'count',
    title = 'Sentiment Polarity Distribution')
```



- Based on the score from the TextBlob library I can assume that the sentimentally of the process is slightly positive.
- It is possible to argue that the sentiment score leans more the neutral as the positive score are not heavily weighted.
- There is a drawback using this method as the textblob library is based on human evaluation of the word. No context for qualifiers are assessed when working out the score.
- Semantics within the English language can greatly alter the meaning of a sentence and this is not taken into account. You cannot comminute tone in text so affectations such as sarcasm are missed.

Text Classification

- Building the model to predict the sentiment of emails.

In [44]:

```
# checking the variable
emails_df.head(1)
```

Out[44]:

	emails	sentiments	sentiment_category
0	hi markus client requesting set neo offboarded...	0.082517	Positive

In [45]:

```
# Dropping the sentiments column
emails_df.drop('sentiments', axis = 1, inplace = True)
```

In [46]:

```
emails_df.head(1)
```

Out[46]:

	emails	sentiment_category
0	hi markus client requesting set neo offboarded...	Positive

- The sentiment score has been successfully removed, leaving me with just the categorisation of the email.

In [47]:

```
# back to the bag of words model

vector = CountVectorizer(max_features=10000, min_df=0.01, ngram_range=(1,1))
```

- Split the data into two in order to train and test model
  - train size - 0.75 (75%)
  - test size - 0.25 (25%)
  - random_state = 123 (for results to be reproducted)

In [48]:

```
#train/ test function
x_train, x_test, y_train, y_test = train_test_split(emails_df['emails'], emails_df['sen
timent_category'],
                                                        test_size = 0.35, random_state = 123
)
```

In [49]:

```
# check x_train data
x_train.head(1)
```

Out[49]:

```
696    hi team could please check rxm ch9692 moved by...
Name: emails, dtype: object
```

In [50]:

```
# check the size of the splits
print(x_train.shape, x_test.shape)
print(y_train.shape, y_test.shape)
```

```
(1203,) (648,)
(1203,) (648,)
```

In [51]:

```
# fit the bag of words model into full text first
vector.fit(emails_df['emails'])
```

Out[51]:

```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
        lowercase=True, max_df=1.0, max_features=10000, min_df=0.01,
        ngram_range=(1, 1), preprocessor=None, stop_words=None,
        strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
        tokenizer=None, vocabulary=None)
```

In [52]:

```
# apply feature transformation to both x_train and x_test
x_train_bow = vector.transform(x_train)

x_test_bow = vector.transform(x_test)
```

In [53]:

```
print(x_train_bow)
```

```
  (0, 313)	1
  (0, 369)	1
  (0, 628)	1
  (0, 697)	2
  (0, 863)	2
  (0, 1358)	3
  (0, 1403)	1
  (0, 1805)	1
  (0, 1904)	2
  (0, 2167)	2
  (0, 2341)	1
  (0, 2354)	1
  (0, 2513)	1
  (0, 2739)	1
  (0, 2819)	1
  (0, 2837)	1
  (0, 2858)	1
  (0, 2889)	1
  (1, 8)	1
  (1, 23)	3
  (1, 30)	3
  (1, 33)	1
  (1, 41)	1
  (1, 52)	1
  (1, 74)	1
  :	:
  (1202, 2438)	4
  (1202, 2486)	1
  (1202, 2510)	1
  (1202, 2546)	1
  (1202, 2566)	1
  (1202, 2582)	1
  (1202, 2587)	4
  (1202, 2591)	4
  (1202, 2605)	2
  (1202, 2751)	4
  (1202, 2795)	2
  (1202, 2796)	2
  (1202, 2819)	4
  (1202, 2837)	3
  (1202, 2840)	1
  (1202, 2923)	6
  (1202, 2952)	1
  (1202, 2953)	1
  (1202, 2959)	2
  (1202, 2961)	2
  (1202, 3021)	3
  (1202, 3032)	1
  (1202, 3036)	1
  (1202, 3065)	1
  (1202, 3089)	1
```

In [54]:

```
# same number of features
print(x_train_bow.shape, x_test_bow.shape)
```

```
(1203, 3123) (648, 3123)
```

In [66]:

```
# vector.get_feature_names()  - commented out for CID reasons
```

- Run sklearn's ML algorithim
- Naive Bayes is the basic one

In [56]:

```
# use the MultinomialNB() function from sklearn
nb = MultinomialNB()
nb
```

Out[56]:

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

In [57]:

```
# first fit and train it on x_train_bow on input and y_train as output
nb.fit(x_train_bow, y_train)
```

Out[57]:

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

- I need to label all e-mails in the data set with a categorical sentiment value.
- I have chosen the binary positive/negative values.
- The label each e-mail will be given will be based on the sentiment score.
- Positive will be assigned to the e-mails with higher sentiment language score and vice-versa.
- Future iterations of this analysis could have more complex categories assigned to provide more nuanced results.

In [58]:

```
# predicitions bsaed on new inpiut
predictions = nb.predict(x_test_bow)

predictions
```

Out[58]:

```
array(['Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Negitive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Negitive', 'Negitive', 'Positive', 'Negitive',
       'Negitive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Negitive', 'Positive', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Negitive', 'Positive', 'Positive',
       'Negitive', 'Negitive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Positive', 'Negitive', 'Positive', 'Positive', 'Positive',
       'Negitive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Negitive', 'Positive', 'Negitive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Negitive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Negitive', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Negitive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Negitive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Negitive', 'Positive', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Negitive', 'Positive', 'Neutral', 'Positive',
       'Positive', 'Positive', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Positive', 'Negitive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Negitive', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Negitive', 'Positive', 'Negitive', 'Positive',
       'Negitive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Negitive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Negitive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Negitive', 'Positive', 'Negitive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Neutral',
       'Negitive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Negitive', 'Negitive', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Negitive', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Negitive', 'Negitive', 'Negitive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Negitive', 'Negitive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Negitive', 'Positive', 'Positive',
       'Negitive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Positive', 'Positive', 'Positive', 'Negitive', 'Positive',
       'Negitive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Negitive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Negitive', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Negitive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Negitive', 'Negitive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Negitive', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Negitive', 'Negitive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Positive', 'Negitive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Negitive', 'Positive',
       'Negitive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Negitive', 'Positive', 'Positive',
       'Positive', 'Neutral', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Negitive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Positive', 'Positive', 'Negitive', 'Negitive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Negitive', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Positive', 'Positive', 'Negitive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Negitive', 'Negitive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Negitive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Negitive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Negitive', 'Positive', 'Negitive',
       'Positive', 'Negitive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Negitive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Negitive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Negitive', 'Positive', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Positive', 'Positive', 'Negitive', 'Positive', 'Positive',
       'Positive', 'Negitive', 'Positive', 'Positive', 'Positive',
       'Negitive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Positive', 'Positive', 'Negitive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive'], dtype='<U8')
```

In [59]:

```
# print accuracy test
accuracy_score(y_test, predictions)
```

Out[59]:

```
0.8333333333333334
```

- The accuracy score is high.
- This means that the model is making the correct prediction 83% of the time. This is very promising.

In [60]:

```
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

    Negitive       0.56      0.55      0.55       108
     Neutral       0.00      0.00      0.00        11
    Positive       0.89      0.91      0.90       529

   micro avg       0.83      0.83      0.83       648
   macro avg       0.48      0.49      0.48       648
weighted avg       0.82      0.83      0.83       648
```

- There are other Machine Learning Algorithms (MLA) we I could use.
- I am going to use a Decision Tree Classifier to see if it produces better results.

In [61]:

```
tree = DecisionTreeClassifier()
```

In [62]:

```
tree.fit(x_train_bow, y_train)
```

Out[62]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=Non
e,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=Non
e,
            splitter='best')
```

In [63]:

```
predictions_tree = tree.predict(x_test_bow)
```

In [64]:

```
predictions_tree
```

Out[64]:

```
array(['Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Negitive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Negitive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Neutral', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Positive', 'Negitive', 'Positive', 'Positive',
       'Neutral', 'Positive', 'Positive', 'Positive', 'Positive',
       'Negitive', 'Negitive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Negitive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Neutral', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Negitive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Negitive', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Positive', 'Negitive', 'Positive', 'Negitive', 'Negitive',
       'Positive', 'Positive', 'Negitive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Negitive', 'Positive', 'Positive', 'Negitive', 'Positive',
       'Negitive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Negitive', 'Neutral', 'Neutral', 'Positive', 'Negitive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Negitive', 'Positive', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Negitive', 'Positive', 'Positive',
       'Positive', 'Negitive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Negitive', 'Positive',
       'Neutral', 'Positive', 'Positive', 'Negitive', 'Positive',
       'Negitive', 'Positive', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Negitive', 'Positive', 'Negitive', 'Positive',
       'Negitive', 'Positive', 'Positive', 'Negitive', 'Negitive',
       'Negitive', 'Negitive', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Negitive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Negitive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Negitive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Negitive', 'Positive', 'Negitive', 'Positive', 'Negitive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Negitive', 'Neutral', 'Negitive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Negitive', 'Negitive', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Negitive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Negitive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Negitive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Negitive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Negitive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Negitive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Negitive', 'Positive', 'Negitive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Neutral', 'Positive', 'Positive',
       'Positive', 'Negitive', 'Negitive', 'Positive', 'Positive',
       'Negitive', 'Negitive', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Positive', 'Positive', 'Negitive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Neutral',
       'Negitive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Negitive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Negitive', 'Positive', 'Positive', 'Negitive', 'Positive',
       'Negitive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Negitive', 'Negitive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Negitive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Negitive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Positive', 'Positive', 'Negitive', 'Negitive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Negitive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Negitive', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Neutral', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Negitive', 'Positive', 'Negitive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Negitive', 'Positive', 'Negitive', 'Positive', 'Positive',
       'Negitive', 'Neutral', 'Negitive', 'Positive', 'Positive',
       'Positive', 'Negitive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Negitive', 'Positive', 'Positive', 'Positive',
       'Negitive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Negitive', 'Positive', 'Positive',
       'Negitive', 'Positive', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Neutral', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Negitive', 'Negitive', 'Positive', 'Negitive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Negitive',
       'Positive', 'Positive', 'Positive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Negitive', 'Positive', 'Positive',
       'Positive', 'Positive', 'Positive'], dtype=object)
```

In [65]:

```
accuracy_score(y_test, predictions_tree)
```

Out[65]:

```
0.9027777777777778
```

Evaluate & Communicate

- Both MLA models I have used have produced accurate models which is positive.
- My original hypothesis "The offboarding process and team is viewed in a positive way by colleagues in UBS." can be said to be true based on my analysis however this simplifies the problem.
- There is no binary way of assessing sentimentality. Textblob has assigned scores which can come back as overall as slightly positive.
- I would argue that the hypothesis is true but the results produced are more nuanced than the conclusion of my hypothesis would suggest.

Future Steps

- My next steps would be including more data from the shared mailbox.
- There are over three years' worth of data to explore however this would require more processing power and time. Just exporting the thousands of emails we receive would take a very long time.
- I would like to try and split the data by the year the e-mails were received as it could create an interesting narrative on how the offboarding team has improved it reputation in the bank.