In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:
```
from scipy.sparse import csr_matrix
```

In [3]:
```
excel = "C:\\Users\\brian\\Documents\\Data Sabic\\2019-10-24 SAP Work Order &
```

In [4]:
```
df = pd.read_excel(excel,sheet_name='INFO')
```

In [5]:
```
df['Notification'] = pd.to_numeric(df['Notification'], errors='coerce').fillr
```

In [6]:
```
df.head()
```

Out[6]:

	Maintenance Plan	Equipment	Manual/ Auto	Order	Notification	Header text:	Created on	
0	BL1000001170	BL000000005-10-035	AUTO	913195093	713335025	* 09/19/2016 19:04:48 rick kessler (FNP_050408...	2016-09-19	0
1	Not Planned	BL000000101-12-206	MANUAL	913195171	713335093	* 09/19/2016 19:13:49 rick kessler (FNP_050408...	2016-09-19	1
2	Not Planned	BL000000104-09-851	MANUAL	913195174	713335096	* 09/19/2016 19:14:44 rick kessler (FNP_050408...	2016-09-19	1 F
3	Not Planned	BL000000154-12-730	MANUAL	913195175	713335097	* 09/19/2016 19:14:57 rick kessler (FNP_050408...	2016-09-19	1 EX
4	BL1000001170	BL000000101-12-206	AUTO	913195180	713335112	* 09/19/2016 19:16:05 rick kessler (FNP_050408...	2016-09-19	1

5 rows × 22 columns

In [7]:
```
df['Tech ID'] = df['Sort field'].apply(lambda x: x.split()[0])
```

In [8]:
```
df.head()
```

2	Not Planned	BL000000104-09-851	MANUAL	913195174	713335096	* 09/19/2016 19:14:44 rick kessler (FNP_050408...	2016-09-19
3	Not Planned	BL000000154-12-730	MANUAL	913195175	713335097	* 09/19/2016 19:14:57 rick kessler (FNP_050408...	2016-09-19
4	BL1000001170	BL000000101-12-206	AUTO	913195180	713335112	* 09/19/2016 19:16:05 rick kessler (FNP_050408...	2016-09-19

In [9]:
```
df2 = pd.read_excel(excel,sheet_name='CYCLE')
```

In [10]:
```
df2
```

Out[10]:

	Maint plan	cycle
0	500000770376	52
1	500000770393	52
2	500000770396	52
3	500000770397	52
4	500000770404	52
5	500000842187	1
6	500000843147	52
7	500000916370	52
8	BL1000001159	4
9	BL1000001170	1
10	BL1000001205	1
11	BL1000001271	4
12	BL1000001608	4
13	BL1000001611	4
14	BL1000001612	52
15	BL1000001614	52

In [11]:
```
d = dict(zip(df2['Maint plan'], df2['cycle']))
```

In [12]:
```
df['Cycle'] = df['Maintenance Plan'].map(d)
```

In [13]:
```
df.head()
```

2	Not Planned	BL000000104-09-851	MANUAL	913195174	713335096	* 09/19/2016 19:14:44 rick kessler (FNP_050408...	2016-09-19
3	Not Planned	BL000000154-12-730	MANUAL	913195175	713335097	* 09/19/2016 19:14:57 rick kessler (FNP_050408...	2016-09-19
4	BL1000001170	BL000000101-12-206	AUTO	913195180	713335112	* 09/19/2016 19:16:05 rick kessler (FNP_050408...	2016-09-19

In [14]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1713 entries, 0 to 1712
Data columns (total 24 columns):
Maintenance Plan                  1713 non-null object
Equipment                         1713 non-null object
Manual/ Auto                      1713 non-null object
Order                             1713 non-null int64
Notification                      1713 non-null int64
Header text:                      1556 non-null object
Created on                        1713 non-null datetime64[ns]
Sort field                        1713 non-null object
Description of Technical Object   1713 non-null object
Description                       1713 non-null object
Functional Location               1713 non-null object
Main work center                  1713 non-null object
ABC indicator                     1713 non-null object
User Status                       1713 non-null object
System status                     1713 non-null object
Priority                          1713 non-null int64
Estimated costs                   238 non-null float64
Total planned costs               238 non-null float64
Total actual costs                238 non-null float64
Basic start date                  238 non-null datetime64[ns]
Basic finish date                 238 non-null datetime64[ns]
MaintActivityType                 238 non-null object
Tech ID                           1713 non-null object
Cycle                             238 non-null float64
dtypes: datetime64[ns](3), float64(4), int64(3), object(14)
memory usage: 321.3+ KB
```

In [15]:
```
df['Cycle'].fillna(value='Not Planned',inplace=True)
```

In [16]:
```
df.head()
```

Out[16]:

	Maintenance Plan	Equipment	Manual/ Auto	Order	Notification	Header text:	Created on
0	BL1000001170	BL000000005-10-035	AUTO	913195093	713335025	* 09/19/2016 19:04:48 rick kessler (FNP_050408...	2016-09-19
1	Not Planned	BL000000101-12-206	MANUAL	913195171	713335093	* 09/19/2016 19:13:49 rick kessler (FNP_050408...	2016-09-19
2	Not Planned	BL000000104-09-851	MANUAL	913195174	713335096	* 09/19/2016 19:14:44 rick kessler	2016-09-19

In [17]:
```
df['Header text:'] = df['Header text:'].astype('str')
```

In [18]:
```
import string
```

In [19]:
```
import nltk
```

In [20]:
```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\brian\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[20]:
```
True
```

In [21]:
```
from nltk.corpus import stopwords
stopwords.words('english')[0:10]
```

Out[21]:
```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're"]
```

In [22]:
```
def text_clean(text):

    nopunc = [char for char in text if char not in string.punctuation]

    nopunc = ''.join(nopunc)

    return [word for word in nopunc.split() if word.lower() not in stopwords.
```

In [23]:
```
df['Header text:'].head(5)
```

Out[23]:
```
0    * 09/19/2016 19:04:48 rick kessler (FNP_050408...
1    * 09/19/2016 19:13:49 rick kessler (FNP_050408...
2    * 09/19/2016 19:14:44 rick kessler (FNP_050408...
3    * 09/19/2016 19:14:57 rick kessler (FNP_050408...
4    * 09/19/2016 19:16:05 rick kessler (FNP_050408...
Name: Header text:, dtype: object
```

In [24]:
```
df['Text Length'] = df['Header text:'].apply(text_clean).apply(len)
```

In [25]:
```
from sklearn.feature_extraction.text import CountVectorizer
```

In [26]:
```
bowxform = CountVectorizer(analyzer=text_clean).fit(df['Header text:'])
```

In [27]:
```
print(len(bowxform.vocabulary_))
```

```
10126
```

In [28]:
```
mess3 = df['Header text:'][2]
print(mess3)
```

```
* 09/19/2016 19:14:44 rick kessler (FNP_050408_F) * 104-09-851 FC1 POLY, In
spect Piping for CUI: OIL LOOP WHERE FLASH FIRE   STARTED, Insulation was r
emoved and replaced. When insulation was   removed, the pipe was inspected
and spots of corrosion were noted.Remove   insulation during turnaround, in
spect, repair if necessary, coat and   replace insulation. * 11-10-2016 18:
31:53 Jim Bowers (US_501692235) *  GLOBAL COMPLETED THIS JOB 11-2-2016.
```

In [29]:
```
bow3 = bowxform.transform([mess3])
print(bow3)
```

```
  (0, 1521)     1
  (0, 1709)     1
  (0, 1804)     1
  (0, 1840)     1
  (0, 3001)     1
  (0, 3125)     1
  (0, 5198)     1
  (0, 5295)     1
  (0, 5337)     1
  (0, 5625)     1
  (0, 5645)     1
  (0, 5651)     1
  (0, 5667)     1
  (0, 5742)     1
  (0, 5898)     1
  (0, 5908)     1
  (0, 5924)     1
  (0, 5941)     1
  (0, 6026)     1
```

In [30]:
```
print(bowxform.get_feature_names()[9909])
```

```
turnaround
```

In [31]:
```
bow = bowxform.transform(df['Header text:'])
```

In [32]:
```
print(bow.shape)
print(bow.nnz) # amount of non-zeros
```

```
(1713, 10126)
42494
```

In [33]:
```
sparsity = (100.0 * bow.nnz / (bow.shape[0] * bow.shape[1]))
print('sparsity: {}'.format(round(sparsity)))
```

```
sparsity: 0
```

In [34]:
```
from sklearn.feature_extraction.text import TfidfTransformer
```

In [35]:
```
tfidfx = TfidfTransformer().fit(bow)
```

In [36]:
```
tfidf3 = tfidfx.transform(bow3)
print(tfidf3)
```

```
  (0, 9909)     0.1499075033480169
  (0, 9622)     0.15766665348626746
  (0, 9345)     0.11559200259137001
  (0, 9278)     0.08812751920942793
  (0, 9277)     0.09851815623905447
  (0, 9263)     0.09960782185813376
  (0, 9257)     0.19775137481021704
  (0, 8977)     0.12688113068795687
  (0, 8806)     0.17879659107128973
  (0, 8777)     0.15766665348626746
  (0, 8488)     0.11559200259137001
  (0, 8425)     0.41243424981337334
  (0, 8403)     0.11346184627635698
  (0, 8402)     0.12457317628108784
  (0, 7754)     0.16944644473759693
  (0, 7638)     0.16944644473759693
  (0, 7063)     0.10154680362102783
  (0, 6762)     0.14682822960481726
  (0, 6434)     0.1534622640043607
```

In [37]:
```
print(bowxform.get_feature_names()[9909])
```

```
turnaround
```

In [38]:
```
print(tfidfx.idf_[bowxform.vocabulary_['fix']])
```

```
7.060290738037835
```

In [39]:
```
tfidf = tfidfx.transform(bow)
print(tfidf.shape)
```

```
(1713, 10126)
```

In [40]:
```
tfidf3.sum()
```

Out[40]:
```
5.9782394048429985
```

In [41]:
```
print(tfidf)
```

```
  (0, 10086)    0.16093646403504225
  (0, 9965)     0.6192517192130789
  (0, 9715)     0.21263168468446075
  (0, 9492)     0.17232928705074455
  (0, 9345)     0.16395791870399865
  (0, 9277)     0.13974004679729188
  (0, 9126)     0.2309362528528563
  (0, 8784)     0.11417979962069731
  (0, 8488)     0.16395791870399865
  (0, 8173)     0.22363741240834734
  (0, 8172)     0.2082639836088106
  (0, 7959)     0.24034609478288166
  (0, 7528)     0.253608522096902
  (0, 6615)     0.2044113982248156
  (0, 5667)     0.16395791870399865
  (0, 3098)     0.253608522096902
  (0, 1521)     0.16613933823695828
  (1, 9345)     0.24428015866046454
  (1, 8488)     0.24428015866046454
```

In [42]:
```
test = pd.DataFrame(tfidf)
```

In [43]:
```
test.head()
```

Out[43]:

	0
0	(0, 10086)\t0.16093646403504225\n (0, 9965)...
1	(0, 9345)\t0.24428015866046454\n (0, 8488)\...
2	(0, 9909)\t0.1499075033480169\n (0, 9622)\t...
3	(0, 9845)\t0.17831278967496197\n (0, 9811)\...
4	(0, 10100)\t0.11713431970231188\n (0, 9345)...

In [44]:
```
test.shape
```

Out[44]:
```
(1713, 1)
```

In [45]:
```
print(test.iloc[0,0])
```

```
  (0, 10086)    0.16093646403504225
  (0, 9965)     0.6192517192130789
  (0, 9715)     0.21263168468446075
  (0, 9492)     0.17232928705074455
  (0, 9345)     0.16395791870399865
  (0, 9277)     0.13974004679729188
  (0, 9126)     0.2309362528528563
  (0, 8784)     0.11417979962069731
  (0, 8488)     0.16395791870399865
  (0, 8173)     0.22363741240834734
  (0, 8172)     0.2082639836088106
  (0, 7959)     0.24034609478288166
  (0, 7528)     0.253608522096902
  (0, 6615)     0.2044113982248156
  (0, 5667)     0.16395791870399865
  (0, 3098)     0.253608522096902
  (0, 1521)     0.16613933823695828
```

In [46]:
```
arg = np.argwhere(tfidf)
```

In [47]:
```
arg
```

Out[47]:
```
array([[    0, 10086],
       [    0,  9965],
       [    0,  9715],
       ...,
       [ 1712,  1417],
       [ 1712,  1404],
       [ 1712,    60]], dtype=int32)
```

In [48]:
```
argdf = pd.DataFrame(arg)
```

In [49]:
```
argdf
```

17	1	9545
18	1	8488
19	1	6534
20	1	6302
21	1	6002
22	1	5814
23	1	5667
24	1	5454
25	1	3223
26	1	3124
27	1	1615
28	1	1521
29	2	9909

In [50]:
```
# Series.to_dense
```

In [51]:
```
tfidf_array = tfidf.toarray()
```

In [52]:
```
print(tfidf_array)
```

```
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```

In [53]:
```
newdf = pd.DataFrame(data=tfidf_array)
```

In [54]:
```
newdf.head()
```

Out[54]:

	0	1	2	3	4	5	6	7	8	9	...	10116	10117	10118	10119	10120	1012
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.

5 rows × 10126 columns

In [55]:
```
tfidf_array.shape
```

Out[55]:
```
(1713, 10126)
```

In [56]:
```
newdf.shape
```

Out[56]:
```
(1713, 10126)
```

In [57]:
```
newdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1713 entries, 0 to 1712
Columns: 10126 entries, 0 to 10125
dtypes: float64(10126)
memory usage: 132.3 MB
```

In [58]:
```
newdf[newdf[0] >0][0]
```

Out[58]:
```
527     0.125524
1257    0.079642
Name: 0, dtype: float64
```

In [59]:
```
new_array= enumerate(tfidf_array)
```

In [60]:
```
print(tfidf_array.shape)
```

```
(1713, 10126)
```

In [61]:
```
for counter, doc in enumerate(tfidf_array):
    word_num = list(zip(bowxform.get_feature_names(), doc))
    one_doc_as_df = pd.DataFrame.from_records(word_num, columns=['term', 'sco
```

In [62]:
```
one_doc_as_df
```

Out[62]:

	term	score
0	drawings	0.498723
1	foreman	0.326147
2	US500977992	0.326147
3	jerry	0.317714
4	loop	0.200994
5	electrical	0.160884
6	Randy	0.133996
7	engineering	0.133996
8	Went	0.121356
9	4	0.083436
10	work	0.078769