# Lab 11 - Jackson Nahom

November 18, 2021

[1]:
```python
import numpy as np
import pandas as pd
from nltk.sentiment import vader
```

[2]:
```python
call_data = pd.read_csv('data/earningscall_fraud.csv')

print(call_data.describe())
```

```
        Unnamed: 0         PRES  TURN_AT_TALK          CEO    WORDCOUNT  \
count  1114.000000  1114.000000   1114.000000  1114.000000  1114.000000
mean    556.500000     0.581688     15.447935     0.793537    18.205566
std     321.728405     0.493504     19.025057     0.404949     8.940149
min       0.000000     0.000000      2.000000     0.000000     1.000000
25%     278.250000     0.000000      3.000000     1.000000    12.000000
50%     556.500000     1.000000      4.000000     1.000000    17.000000
75%     834.750000     1.000000     24.000000     1.000000    23.000000
max    1113.000000     1.000000     93.000000     1.000000    62.000000

       Restatement Topic        FRAUD
count        1114.000000  1114.000000
mean            0.176840     0.352783
std             0.381705     0.636108
min             0.000000     0.000000
25%             0.000000     0.000000
50%             0.000000     0.000000
75%             0.000000     1.000000
max             1.000000     2.000000
```

[3]:
```python
# percent of fraud
print(call_data['Restatement Topic'].mean())
```

```
0.17684021543985637
```

[4]:
```python
from gensim.parsing.preprocessing import preprocess_string
from gensim import corpora

call_data['clean_text'] = call_data['Sentence'].apply(preprocess_string)
print(call_data.loc[1, ['Sentence', 'clean_text']])
```

```
Sentence      Welcome to Northwest Pipe's conference call an…
clean_text    [welcom, northwest, pipe, confer, announc, ear…
Name: 1, dtype: object
```

[5]:
```python
dictionary = corpora.Dictionary(call_data['clean_text'])
print(dictionary)
```

```
Dictionary(1116 unique tokens: ['diann', 'thank', 'announc', 'confer',
'earn']…)
```

[6]:
```python
bow_corpus = [dictionary.doc2bow(text) for text in call_data['clean_text']]
```

[7]:
```python
from gensim import models

lda_10 = models.LdaModel(bow_corpus, num_topics=10, id2word=dictionary)
```

[8]:
```python
for topic in lda_10.show_topics():
    print("Topic", topic[0], ":", topic[1])
```

```
Topic 0 : 0.031*"quarter" + 0.026*"think" + 0.020*"certainli" + 0.020*"bit" +
0.019*"littl" + 0.017*"price" + 0.015*"ye" + 0.010*"margin" + 0.009*"brian" +
0.009*"product"
Topic 1 : 0.029*"million" + 0.025*"cost" + 0.022*"quarter" + 0.021*"project" +
0.018*"water" + 0.017*"steel" + 0.016*"year" + 0.013*"fourth" +
0.011*"transmiss" + 0.011*"think"
Topic 2 : 0.035*"million" + 0.024*"project" + 0.023*"quarter" + 0.019*"expect" +
0.018*"revenu" + 0.017*"water" + 0.016*"steel" + 0.014*"group" + 0.012*"product"
+ 0.012*"year"
Topic 3 : 0.038*"price" + 0.028*"steel" + 0.027*"cost" + 0.022*"product" +
0.019*"sell" + 0.017*"expect" + 0.015*"differ" + 0.012*"averag" +
0.011*"quarter" + 0.009*"market"
Topic 4 : 0.020*"market" + 0.017*"go" + 0.014*"thank" + 0.013*"quarter" +
0.013*"strong" + 0.013*"product" + 0.012*"project" + 0.011*"busi" +
0.011*"believ" + 0.010*"good"
Topic 5 : 0.034*"product" + 0.022*"quarter" + 0.019*"expect" + 0.016*"market" +
0.016*"cost" + 0.015*"time" + 0.014*"steel" + 0.013*"energi" + 0.013*"tubular" +
0.012*"activ"
Topic 6 : 0.032*"million" + 0.022*"product" + 0.018*"year" + 0.018*"quarter" +
0.017*"revenu" + 0.016*"water" + 0.015*"time" + 0.013*"sale" + 0.013*"increas" +
0.012*"result"
Topic 7 : 0.021*"product" + 0.017*"tubular" + 0.017*"increas" + 0.017*"seen" +
0.014*"think" + 0.013*"result" + 0.012*"cost" + 0.012*"steel" + 0.012*"look" +
0.012*"signific"
Topic 8 : 0.035*"quarter" + 0.031*"year" + 0.028*"million" + 0.028*"expect" +
0.021*"look" + 0.020*"think" + 0.016*"cost" + 0.015*"project" + 0.013*"record" +
0.011*"backlog"
Topic 9 : 0.060*"quarter" + 0.044*"million" + 0.029*"year" + 0.015*"busi" +
0.014*"expect" + 0.014*"product" + 0.013*"market" + 0.013*"share" + 0.012*"sale"
+ 0.012*"result"
```

[9]:
```
print('Perplexity: ', lda_10.log_perplexity(bow_corpus))
```

```
Perplexity:  -6.879600814846877
```

[10]:
```
from gensim.models import CoherenceModel
coherence_model_lda = CoherenceModel(model=lda_10,␣
↪texts=call_data['clean_text'], dictionary=dictionary, coherence='u_mass')
coherence_lda = coherence_model_lda.get_coherence()
print('\nCoherence Score: ', coherence_lda)
```

```

Coherence Score:  -6.402824932960739
```

[11]:
```
stopwords = []
with open('data/stoplist.txt', 'r') as f:
    stopwords = f.read().splitlines()
```

[12]:
```
def remove_stopwords(text):
    """ preprocess string and remove words from custom stopword list. """
    result = []

    for word in preprocess_string(text):
        if word not in stopwords:
            result.append(word)
    return result

call_data['clean_newstop'] = call_data['Sentence'].apply(remove_stopwords)
```

[13]:
```
new_dictionary = corpora.Dictionary(call_data['clean_newstop'])
print(new_dictionary)

new_corpus = [new_dictionary.doc2bow(text) for text in␣
↪call_data['clean_newstop']]
```

```
Dictionary(1001 unique tokens: ['diann', 'announc', 'confer', 'earn',
'northwest']…)
```

[14]:
```
lda_new = models.LdaModel(new_corpus, num_topics=10, id2word=new_dictionary)

for topic in lda_new.show_topics():
    print("Topic", topic[0], ":", topic[1])
```

```
Topic 0 : 0.040*"steel" + 0.028*"compar" + 0.021*"tubular" + 0.021*"group" +
0.014*"financi" + 0.013*"signific" + 0.012*"certainli" + 0.011*"water" +
0.011*"pipe" + 0.011*"statement"
Topic 1 : 0.019*"steel" + 0.017*"signific" + 0.016*"water" + 0.015*"month" +
0.013*"believ" + 0.012*"opportun" + 0.010*"issu" + 0.009*"continu" + 0.009*"ago"
+ 0.009*"transmiss"
Topic 2 : 0.025*"share" + 0.021*"project" + 0.018*"earn" + 0.014*"result" +
```

```
0.012*"activ" + 0.010*"pipe" + 0.010*"probabl" + 0.010*"ahead" + 0.010*"strong"
+ 0.010*"compar"
Topic 3 : 0.021*"project" + 0.018*"gener" + 0.016*"result" + 0.015*"improv" +
0.013*"materi" + 0.013*"volum" + 0.012*"steel" + 0.012*"activ" + 0.011*"energi"
+ 0.011*"higher"
Topic 4 : 0.028*"backlog" + 0.019*"month" + 0.019*"project" + 0.015*"book" +
0.015*"order" + 0.013*"result" + 0.013*"ahead" + 0.011*"certainli" +
0.011*"forecast" + 0.011*"report"
Topic 5 : 0.023*"tubular" + 0.017*"strong" + 0.015*"steel" + 0.015*"result" +
0.015*"group" + 0.014*"bid" + 0.013*"water" + 0.012*"gener" + 0.012*"transmiss"
+ 0.011*"backlog"
Topic 6 : 0.021*"project" + 0.021*"brian" + 0.015*"stimulu" + 0.014*"bid" +
0.012*"steel" + 0.012*"order" + 0.011*"improv" + 0.011*"pipe" + 0.011*"gross" +
0.011*"profit"
Topic 7 : 0.023*"water" + 0.020*"backlog" + 0.017*"compar" + 0.016*"tubular" +
0.014*"strong" + 0.014*"transmiss" + 0.013*"net" + 0.013*"gross" +
0.011*"discuss" + 0.011*"continu"
Topic 8 : 0.029*"water" + 0.020*"steel" + 0.018*"differ" + 0.018*"transmiss" +
0.015*"tax" + 0.011*"project" + 0.011*"approxim" + 0.010*"effect" +
0.010*"capit" + 0.009*"averag"
Topic 9 : 0.042*"project" + 0.023*"certainli" + 0.019*"steel" + 0.018*"continu"
+ 0.013*"fund" + 0.012*"water" + 0.011*"agenc" + 0.010*"gener" + 0.010*"current"
+ 0.009*"opportun"
```

[15]:

```python
for doc in new_corpus[0:9]:
    print(lda_new.get_document_topics(doc))
```

```
[(0, 0.5499266), (1, 0.05000604), (2, 0.05000604), (3, 0.05000604), (4,
0.05000604), (5, 0.05000604), (6, 0.05000604), (7, 0.05000604), (8, 0.05000604),
(9, 0.050025117)]
[(0, 0.014287135), (1, 0.014286572), (2, 0.8714208), (3, 0.014286144), (4,
0.014286419), (5, 0.014286053), (6, 0.014287521), (7, 0.014286855), (8,
0.014286233), (9, 0.014286276)]
[(0, 0.93076396)]
[(0, 0.89998513), (1, 0.011112107), (2, 0.011112673), (3, 0.011113541), (4,
0.011113584), (5, 0.011112443), (6, 0.0111116385), (7, 0.011112282), (8,
0.011114783), (9, 0.011111833)]
[(3, 0.92498994)]
[(0, 0.016672745), (1, 0.016667735), (2, 0.84997666), (3, 0.016670074), (4,
0.016670566), (5, 0.016668493), (6, 0.016668448), (7, 0.016668517), (8,
0.016668335), (9, 0.01666842)]
[(0, 0.1), (1, 0.1), (2, 0.1), (3, 0.1), (4, 0.1), (5, 0.1), (6, 0.1), (7, 0.1),
(8, 0.1), (9, 0.1)]
[(0, 0.020002535), (1, 0.02000292), (2, 0.020005587), (3, 0.020004375), (4,
0.020005584), (5, 0.8199632), (6, 0.020002196), (7, 0.02000614), (8,
0.020002533), (9, 0.020004896)]
[(0, 0.020010464), (1, 0.020002734), (2, 0.020002205), (3, 0.02000322), (4,
0.81996214), (5, 0.020006903), (6, 0.02000185), (7, 0.020005394), (8,
```

0.02000424), (9, 0.020000843)]

[16]:
```
print('Perplexity: ', lda_new.log_perplexity(new_corpus))
```

Perplexity:  -7.250209451537484

[17]:
```
print('Perplexity: ', lda_new.log_perplexity(new_corpus))
```

Perplexity:  -7.250243200901119

[18]:
```
coherence_model_lda = CoherenceModel(model=lda_new,␣
↪texts=call_data['clean_text'], dictionary=dictionary, coherence='u_mass')
coherence_lda = coherence_model_lda.get_coherence()
print('\nCoherence Score: ', coherence_lda)
```

Coherence Score:  -18.18188088238054

[19]:
```
from gensim.matutils import corpus2csc
all_topics = lda_new.get_document_topics(new_corpus, minimum_probability=0.0)
all_topics_csr = corpus2csc(all_topics)
all_topics_numpy = all_topics_csr.T.toarray()
all_topics_df = pd.DataFrame(all_topics_numpy)

classification_df = pd.concat([call_data, all_topics_df], axis=1)
```

[20]:
```
classification_df.describe()
```

[20]:

| | Unnamed: 0 | PRES | TURN_AT_TALK | CEO | WORDCOUNT \ |
|---|---|---|---|---|---|
| count | 1114.000000 | 1114.000000 | 1114.000000 | 1114.000000 | 1114.000000 |
| mean | 556.500000 | 0.581688 | 15.447935 | 0.793537 | 18.205566 |
| std | 321.728405 | 0.493504 | 19.025057 | 0.404949 | 8.940149 |
| min | 0.000000 | 0.000000 | 2.000000 | 0.000000 | 1.000000 |
| 25% | 278.250000 | 0.000000 | 3.000000 | 1.000000 | 12.000000 |
| 50% | 556.500000 | 1.000000 | 4.000000 | 1.000000 | 17.000000 |
| 75% | 834.750000 | 1.000000 | 24.000000 | 1.000000 | 23.000000 |
| max | 1113.000000 | 1.000000 | 93.000000 | 1.000000 | 62.000000 |

| | Restatement Topic | FRAUD | 0 | 1 | 2 \ |
|---|---|---|---|---|---|
| count | 1114.000000 | 1114.000000 | 1114.000000 | 1114.000000 | 1114.000000 |
| mean | 0.176840 | 0.352783 | 0.106894 | 0.100772 | 0.100839 |
| std | 0.381705 | 0.636108 | 0.233505 | 0.223849 | 0.229745 |
| min | 0.000000 | 0.000000 | 0.005264 | 0.005557 | 0.005265 |
| 25% | 0.000000 | 0.000000 | 0.014288 | 0.014288 | 0.014288 |
| 50% | 0.000000 | 0.000000 | 0.020004 | 0.020006 | 0.020005 |
| 75% | 0.000000 | 1.000000 | 0.033347 | 0.033356 | 0.033345 |
| max | 1.000000 | 2.000000 | 0.939994 | 0.952623 | 0.943732 |

| | 3 | 4 | 5 | 6 | 7 \ |
|---|---|---|---|---|---|

```
count  1114.000000  1114.000000  1114.000000  1114.000000  1114.000000
mean      0.114237     0.090887     0.099371     0.086434     0.102081
std       0.244002     0.208439     0.227154     0.204559     0.231783
min       0.005264     0.005264     0.005264     0.005264     0.005264
25%       0.014288     0.014287     0.014288     0.014287     0.014288
50%       0.020005     0.020003     0.020005     0.020004     0.020005
75%       0.050001     0.033342     0.033344     0.033339     0.033349
max       0.939987     0.943736     0.939990     0.943740     0.935705

                 8            9
count  1114.000000  1114.000000
mean      0.095157     0.103329
std       0.215157     0.232446
min       0.005264     0.005264
25%       0.014287     0.014288
50%       0.020004     0.020005
75%       0.033343     0.033347
max       0.939984     0.949985
```

[21]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import StratifiedShuffleSplit
from sklearn.model_selection import cross_validate

n_splits = 5

pred_vars = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9,]


scoring = ['accuracy', 'neg_log_loss', 'f1', 'roc_auc']
rf_base = RandomForestClassifier()
cv_rf = cross_validate(rf_base, classification_df[pred_vars],
 ↪classification_df['Restatement Topic'], cv=StratifiedShuffleSplit(n_splits),
 ↪scoring=scoring)
print(cv_rf)
print("Mean Accuracy:", cv_rf['test_accuracy'].mean())
print("Mean F1:", cv_rf['test_f1'].mean())
print("Mean ROC:", cv_rf['test_roc_auc'].mean())
print("Mean Log Loss:", cv_rf['test_neg_log_loss'].mean())
```

```
{'fit_time': array([0.31362104, 0.32213783, 0.30219221, 0.31323647,
0.30397129]), 'score_time': array([0.04085565, 0.03294563, 0.03494191,
0.03191447, 0.03101945]), 'test_accuracy': array([0.82142857, 0.78571429,
0.79464286, 0.82142857, 0.82142857]), 'test_neg_log_loss': array([-0.71471155,
-0.44971985, -0.75287328, -0.73258985, -0.45952129]), 'test_f1':
array([0.33333333, 0.2       , 0.20689655, 0.33333333, 0.16666667]),
'test_roc_auc': array([0.68179348, 0.69809783, 0.66467391, 0.65380435,
0.63532609])}
Mean Accuracy: 0.8089285714285713
```

```
Mean F1: 0.24804597701149428
Mean ROC: 0.6667391304347826
Mean Log Loss: -0.621883163472536
```

# 1 Tasks

1. Try fitting LDA with just 5 topics instead of 10. How does this affect human interpretability, perplexity, coherence, and classification performance?
2. Try fitting LDA with 15 topics. How does this affect human interpretability, perplexity, coherence, and classification performance?
3. In addition to Random Forest, try another classifier of your choosing. How does this compare to the Random Forest?

# 2 Optional Tasks

1. Run sentiment analysis on this data. Does adding that to a classifier improve performance?
2. See the section below on weighted word counts. Does using tf-idf improve human interpretability?

# 3 Task 1

Try fitting LDA with just 5 topics instead of 10. How does this affect human interpretability, perplexity, coherence, and classification performance?

[22]:
```
lda_new_2 = models.LdaModel(new_corpus, num_topics=5, id2word=new_dictionary)

for topic in lda_new_2.show_topics():
    print("Topic", topic[0], ":", topic[1])
```

```
Topic 0 : 0.020*"group" + 0.019*"backlog" + 0.017*"water" + 0.016*"result" +
0.016*"lower" + 0.015*"transmiss" + 0.014*"pipe" + 0.013*"strong" +
0.012*"project" + 0.012*"steel"
Topic 1 : 0.026*"steel" + 0.014*"certainli" + 0.014*"signific" + 0.013*"tubular"
+ 0.013*"result" + 0.012*"continu" + 0.011*"effect" + 0.011*"project" +
0.010*"energi" + 0.010*"statement"
Topic 2 : 0.018*"continu" + 0.017*"compar" + 0.016*"believ" + 0.015*"gener" +
0.013*"half" + 0.013*"bid" + 0.013*"net" + 0.013*"incom" + 0.012*"activ" +
0.011*"share"
Topic 3 : 0.042*"project" + 0.015*"tubular" + 0.014*"water" + 0.014*"higher" +
0.012*"steel" + 0.012*"compar" + 0.011*"materi" + 0.011*"discuss" + 0.011*"plan"
+ 0.010*"stimulu"
Topic 4 : 0.026*"steel" + 0.019*"gener" + 0.018*"water" + 0.014*"record" +
0.010*"tubular" + 0.009*"inventori" + 0.009*"earn" + 0.009*"result" +
0.009*"sell" + 0.008*"posit"
```

[23]:
```
for doc in new_corpus[0:5]:
    print(lda_new_2.get_document_topics(doc))
```

```
[(0, 0.10002244), (1, 0.10002665), (2, 0.1000246), (3, 0.59989685), (4,
0.10002943)]
[(0, 0.88521415), (1, 0.028718451), (2, 0.028634999), (3, 0.028606309), (4,
0.028826077)]
[(0, 0.9380498), (1, 0.015489789), (2, 0.015464462), (3, 0.015550069), (4,
0.015445879)]
[(0, 0.02239238), (1, 0.9105051), (2, 0.022325426), (3, 0.022495663), (4,
0.022281416)]
[(0, 0.016834844), (1, 0.016838241), (2, 0.016740467), (3, 0.932851), (4,
0.016735407)]
```

[24]:
```
print('Perplexity: ', lda_new_2.log_perplexity(new_corpus))
print('Perplexity: ', lda_new_2.log_perplexity(new_corpus))
```

```
Perplexity:  -6.901986630853719
Perplexity:  -6.901965556211364
```

[25]:
```
coherence_model_lda = CoherenceModel(model=lda_new_2,␣
 ↪texts=call_data['clean_text'], dictionary=dictionary, coherence='u_mass')
coherence_lda = coherence_model_lda.get_coherence()
print('\nCoherence Score: ', coherence_lda)
```

```

Coherence Score:  -18.567963858613645
```

[26]:
```
all_topics = lda_new_2.get_document_topics(new_corpus, minimum_probability=0.0)
all_topics_csr = corpus2csc(all_topics)
all_topics_numpy = all_topics_csr.T.toarray()
all_topics_df = pd.DataFrame(all_topics_numpy)

classification_df = pd.concat([call_data, all_topics_df], axis=1)
```

[27]:
```
classification_df.describe()
```

[27]:
```
        Unnamed: 0         PRES  TURN_AT_TALK          CEO    WORDCOUNT  \
count  1114.000000  1114.000000   1114.000000  1114.000000  1114.000000
mean    556.500000     0.581688     15.447935     0.793537    18.205566
std     321.728405     0.493504     19.025057     0.404949     8.940149
min       0.000000     0.000000      2.000000     0.000000     1.000000
25%     278.250000     0.000000      3.000000     1.000000    12.000000
50%     556.500000     1.000000      4.000000     1.000000    17.000000
75%     834.750000     1.000000     24.000000     1.000000    23.000000
max    1113.000000     1.000000     93.000000     1.000000    62.000000

       Restatement Topic        FRAUD            0            1            2  \
count        1114.000000  1114.000000  1114.000000  1114.000000  1114.000000
```

```
mean        0.176840    0.352783    0.234218    0.183739    0.206125
std         0.381705    0.636108    0.319316    0.284007    0.301522
min         0.000000    0.000000    0.010623    0.010648    0.010635
25%         0.000000    0.000000    0.033413    0.029147    0.029119
50%         0.000000    0.000000    0.050957    0.050124    0.050479
75%         0.000000    1.000000    0.313860    0.102767    0.200000
max         1.000000    2.000000    0.946339    0.948852    0.949544

                 3            4
count  1114.000000  1114.000000
mean      0.212392     0.163526
std       0.301806     0.266406
min       0.010692     0.011438
25%       0.033370     0.028975
50%       0.050622     0.041776
75%       0.200000     0.100170
max       0.954486     0.957246
```

[28]:

```
n_splits = 5

pred_vars = [0, 1, 2, 3, 4,]


scoring = ['accuracy', 'neg_log_loss', 'f1', 'roc_auc']
rf_base = RandomForestClassifier()
cv_rf = cross_validate(rf_base, classification_df[pred_vars],␣
 ↪classification_df['Restatement Topic'], cv=StratifiedShuffleSplit(n_splits),␣
 ↪scoring=scoring)
print(cv_rf)
print("Mean Accuracy:", cv_rf['test_accuracy'].mean())
print("Mean F1:", cv_rf['test_f1'].mean())
print("Mean ROC:", cv_rf['test_roc_auc'].mean())
print("Mean Log Loss:", cv_rf['test_neg_log_loss'].mean())
```

```
{'fit_time': array([0.27995086, 0.27462554, 0.2818048 , 0.2818346 , 0.2579627
]), 'score_time': array([0.03387499, 0.03191972, 0.0388999 , 0.03247809,
0.0319469 ]), 'test_accuracy': array([0.82142857, 0.78571429, 0.77678571,
0.78571429, 0.8125    ]), 'test_neg_log_loss': array([-0.45937906, -0.81721368,
-1.09159659, -0.82516563, -0.83409488]), 'test_f1': array([0.375     , 0.2
, 0.32432432, 0.2       , 0.27586207]), 'test_roc_auc': array([0.6923913 ,
0.60543478, 0.5923913 , 0.55407609, 0.56548913])}
Mean Accuracy: 0.7964285714285715
Mean F1: 0.27503727865796834
Mean ROC: 0.6019565217391304
Mean Log Loss: -0.805489966212023
```

human interpretability: Topic 2 is the only cohearnt one that seems like they are talking about making a bid for a project and how much income they will make from it. The 5 topic human interpretability is worse than the original 10 topics in my opinon.

perplexity: 5 topics: -6.908 vs 10 topics: -7.203

coherence: 5 topics: -17.971 vs 10 topics: -18.127

classification performance: 5 Topics
Mean Accuracy: 0.8107142857142857
Mean F1: 0.2939923628466454
Mean ROC: 0.6096739130434783
Mean Log Loss: -0.6767382105982549
VS.
10 Topics
Mean Accuracy: 0.8071428571428572
Mean F1: 0.2180214166421063
Mean ROC: 0.6490217391304347
Mean Log Loss: -0.5865378903125963

Accuracy and F1 improved while ROC and Log Loss did not improve.

# 4 Task 2

Try fitting LDA with 15 topics. How does this affect human interpretability, perplexity, coherence, and classification performance?

[38]:
```
lda_new_3 = models.LdaModel(new_corpus, num_topics=15, id2word=new_dictionary)
#print(len(lda_new_3.show_topics()))
for topic in lda_new_3.show_topics(num_topics=15):
    #print()
    print("Topic", topic[0], ":", topic[1])
```

```
Topic 0 : 0.029*"project" + 0.023*"activ" + 0.014*"result" + 0.014*"group" +
0.011*"oper" + 0.011*"construct" + 0.011*"lower" + 0.011*"bid" + 0.011*"tubular"
+ 0.009*"fund"
Topic 1 : 0.019*"tubular" + 0.017*"order" + 0.014*"countri" + 0.014*"project" +
0.012*"recent" + 0.012*"tax" + 0.012*"signific" + 0.012*"activ" +
0.011*"postpon" + 0.011*"pipe"
Topic 2 : 0.027*"project" + 0.026*"steel" + 0.019*"believ" + 0.011*"fund" +
0.011*"plan" + 0.011*"ton" + 0.011*"effect" + 0.010*"gener" + 0.008*"acquir" +
0.008*"manufactur"
Topic 3 : 0.057*"water" + 0.047*"transmiss" + 0.039*"steel" + 0.028*"group" +
0.016*"certainli" + 0.011*"gener" + 0.010*"rang" + 0.010*"result" +
0.010*"compar" + 0.010*"strong"
Topic 4 : 0.033*"differ" + 0.023*"statement" + 0.022*"group" + 0.017*"certainli"
+ 0.014*"steel" + 0.014*"futur" + 0.014*"continu" + 0.014*"tubular" +
0.012*"profit" + 0.012*"plan"
Topic 5 : 0.019*"project" + 0.017*"signific" + 0.017*"certainli" +
0.015*"tubular" + 0.013*"probabl" + 0.013*"compar" + 0.012*"continu" +
0.012*"case" + 0.012*"improv" + 0.012*"pipe"
Topic 6 : 0.025*"report" + 0.019*"compar" + 0.016*"steel" + 0.016*"tubular" +
```

```
0.014*"profit" + 0.014*"lower" + 0.014*"gross" + 0.014*"tax" + 0.014*"stimulu" +
0.011*"rel"
Topic 7 : 0.025*"bid" + 0.024*"project" + 0.024*"water" + 0.020*"week" +
0.019*"continu" + 0.015*"sell" + 0.014*"steel" + 0.013*"half" + 0.013*"improv" +
0.013*"transmiss"
Topic 8 : 0.029*"project" + 0.020*"steel" + 0.018*"gener" + 0.018*"result" +
0.013*"higher" + 0.013*"specif" + 0.013*"issu" + 0.013*"brian" + 0.012*"sell" +
0.011*"volum"
Topic 9 : 0.016*"group" + 0.016*"strong" + 0.016*"differ" + 0.016*"compar" +
0.012*"reason" + 0.012*"structur" + 0.012*"share" + 0.012*"tubular" +
0.012*"slightli" + 0.008*"volum"
Topic 10 : 0.018*"steel" + 0.017*"energi" + 0.015*"compar" + 0.014*"tubular" +
0.013*"inventori" + 0.013*"water" + 0.012*"demand" + 0.010*"financi" +
0.010*"believ" + 0.010*"month"
Topic 11 : 0.021*"backlog" + 0.020*"energi" + 0.016*"fund" + 0.016*"order" +
0.016*"job" + 0.016*"continu" + 0.014*"activ" + 0.013*"probabl" + 0.013*"strong"
+ 0.013*"bond"
Topic 12 : 0.028*"project" + 0.026*"continu" + 0.026*"backlog" +
0.021*"certainli" + 0.016*"strong" + 0.015*"water" + 0.014*"share" +
0.012*"half" + 0.009*"infrastructur" + 0.009*"schedul"
Topic 13 : 0.025*"group" + 0.022*"result" + 0.019*"steel" + 0.016*"order" +
0.016*"adjust" + 0.016*"higher" + 0.016*"tubular" + 0.013*"ahead" +
0.013*"materi" + 0.010*"stephani"
Topic 14 : 0.030*"steel" + 0.022*"record" + 0.017*"share" + 0.017*"result" +
0.017*"tubular" + 0.015*"signific" + 0.014*"project" + 0.012*"pipe" +
0.012*"gross" + 0.011*"profit"
```

[30]:
```python
for doc in new_corpus[0:15]:
    print(lda_new_3.get_document_topics(doc))
```

```
[(0, 0.033336002), (1, 0.033336185), (2, 0.033336002), (3, 0.033336002), (4,
0.033336002), (5, 0.53329575), (6, 0.033336002), (7, 0.033336002), (8,
0.033336002), (9, 0.033336002), (10, 0.033336002), (11, 0.033336002), (12,
0.033336002), (13, 0.033336002), (14, 0.033336002)]
[(1, 0.8666633)]
[(9, 0.92820364)]
[(14, 0.896295)]
[(14, 0.92222023)]
[(0, 0.011111328), (1, 0.51886714), (2, 0.011111335), (3, 0.01111132), (4,
0.33668563), (5, 0.011111324), (6, 0.011111324), (7, 0.011111328), (8,
0.011111325), (9, 0.011111327), (10, 0.011111327), (11, 0.0111113265), (12,
0.011111331), (13, 0.011111325), (14, 0.011111332)]
[(0, 0.06666667), (1, 0.06666667), (2, 0.06666667), (3, 0.06666667), (4,
0.06666667), (5, 0.06666667), (6, 0.06666667), (7, 0.06666667), (8, 0.06666667),
(9, 0.06666667), (10, 0.06666667), (11, 0.06666667), (12, 0.06666667), (13,
0.06666667), (14, 0.06666667)]
[(0, 0.8133301), (1, 0.013333587), (2, 0.013333562), (3, 0.013333556), (4,
0.0133335665), (5, 0.013333556), (6, 0.013333547), (7, 0.013333577), (8,
```

```
0.013333538), (9, 0.0133335395), (10, 0.013333575), (11, 0.013333552), (12,
0.013333581), (13, 0.013333564), (14, 0.013333564)]
[(0, 0.013333632), (1, 0.013333624), (2, 0.013333648), (3, 0.013333647), (4,
0.01333363), (5, 0.013333637), (6, 0.01333363), (7, 0.013333634), (8,
0.013333648), (9, 0.013333636), (10, 0.8133291), (11, 0.013333641), (12,
0.013333642), (13, 0.013333626), (14, 0.013333654)]
[(0, 0.011111428), (1, 0.011111417), (2, 0.011111425), (3, 0.011111418), (4,
0.84444016), (5, 0.0111114355), (6, 0.011111407), (7, 0.011111409), (8,
0.01111141), (9, 0.011111412), (10, 0.011111446), (11, 0.011111416), (12,
0.0111114085), (13, 0.011111428), (14, 0.011111417)]
[(0, 0.013333912), (1, 0.013333905), (2, 0.013333938), (3, 0.81332505), (4,
0.013333916), (5, 0.013333915), (6, 0.013333892), (7, 0.0133339), (8,
0.0133339595), (9, 0.013333979), (10, 0.013333946), (11, 0.013333957), (12,
0.013333942), (13, 0.013333908), (14, 0.013333927)]
[(0, 0.016666776), (1, 0.016666772), (2, 0.016666893), (3, 0.016666774), (4,
0.01666678), (5, 0.01666677), (6, 0.016666794), (7, 0.01666679), (8,
0.016666764), (9, 0.016666776), (10, 0.766665), (11, 0.016666768), (12,
0.016666794), (13, 0.016666777), (14, 0.016666783)]
[(11, 0.88332677)]
[(9, 0.86666036)]
[(0, 0.013333538), (1, 0.01333353), (2, 0.013333555), (3, 0.013333533), (4,
0.013333537), (5, 0.01333353), (6, 0.013333545), (7, 0.013333569), (8,
0.01333353), (9, 0.013333533), (10, 0.5583055), (11, 0.01333353), (12,
0.01333355), (13, 0.26835847), (14, 0.013333535)]
```

[31]:
```python
print('Perplexity: ', lda_new_3.log_perplexity(new_corpus))
print('Perplexity: ', lda_new_3.log_perplexity(new_corpus))
```

```
Perplexity:  -7.487039290774965
Perplexity:  -7.488385576863903
```

[32]:
```python
coherence_model_lda = CoherenceModel(model=lda_new_3,␣
↪texts=call_data['clean_text'], dictionary=dictionary, coherence='u_mass')
coherence_lda = coherence_model_lda.get_coherence()
print('\nCoherence Score: ', coherence_lda)
```

```

Coherence Score:  -18.33191010177371
```

[33]:
```python
all_topics = lda_new_3.get_document_topics(new_corpus, minimum_probability=0.0)
all_topics_csr = corpus2csc(all_topics)
all_topics_numpy = all_topics_csr.T.toarray()
all_topics_df = pd.DataFrame(all_topics_numpy)

classification_df = pd.concat([call_data, all_topics_df], axis=1)
```

[34]:
```python
classification_df.describe()
```

[34]:

```
       Unnamed: 0         PRES  TURN_AT_TALK          CEO    WORDCOUNT  \
count  1114.000000  1114.000000   1114.000000  1114.000000  1114.000000
mean    556.500000     0.581688     15.447935     0.793537    18.205566
std     321.728405     0.493504     19.025057     0.404949     8.940149
min       0.000000     0.000000      2.000000     0.000000     1.000000
25%     278.250000     0.000000      3.000000     1.000000    12.000000
50%     556.500000     1.000000      4.000000     1.000000    17.000000
75%     834.750000     1.000000     24.000000     1.000000    23.000000
max    1113.000000     1.000000     93.000000     1.000000    62.000000

       Restatement Topic        FRAUD            0            1            2  \
count        1114.000000  1114.000000  1114.000000  1114.000000  1114.000000
mean            0.176840     0.352783     0.073247     0.067150     0.060742
std             0.381705     0.636108     0.197859     0.183461     0.182007
min             0.000000     0.000000     0.003509     0.003509     0.003509
25%             0.000000     0.000000     0.009524     0.009524     0.009524
50%             0.000000     0.000000     0.013334     0.013334     0.013334
75%             0.000000     1.000000     0.022223     0.022223     0.022223
max             1.000000     2.000000     0.941664     0.915150     0.937775

       …            5            6            7            8            9  \
count  …  1114.000000  1114.000000  1114.000000  1114.000000  1114.000000
mean   …     0.051696     0.052546     0.072565     0.077235     0.071115
std    …     0.158835     0.162186     0.197851     0.204955     0.191248
min    …     0.003509     0.003509     0.003509     0.003509     0.003509
25%    …     0.009524     0.009524     0.009524     0.009524     0.009524
50%    …     0.013334     0.013334     0.013334     0.013334     0.013334
75%    …     0.022223     0.022223     0.022223     0.022223     0.022223
max    …     0.941663     0.937775     0.933330     0.933330     0.950874

                10           11           12           13           14
count  1114.000000  1114.000000  1114.000000  1114.000000  1114.000000
mean      0.077117     0.074054     0.074296     0.065179     0.060940
std       0.199211     0.199990     0.200914     0.185033     0.182471
min       0.003509     0.003509     0.003509     0.003509     0.003509
25%       0.009524     0.009524     0.009524     0.009524     0.009524
50%       0.013334     0.013334     0.013334     0.013334     0.013334
75%       0.022224     0.022223     0.022223     0.022223     0.022223
max       0.941664     0.937776     0.941664     0.950874     0.948146

[8 rows x 22 columns]
```

[35]:

```
n_splits = 5

pred_vars = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,]
```

```
scoring = ['accuracy', 'neg_log_loss', 'f1', 'roc_auc']
rf_base = RandomForestClassifier()
cv_rf = cross_validate(rf_base, classification_df[pred_vars],␣
 ↪classification_df['Restatement Topic'], cv=StratifiedShuffleSplit(n_splits),␣
 ↪scoring=scoring)
print(cv_rf)
print("Mean Accuracy:", cv_rf['test_accuracy'].mean())
print("Mean F1:", cv_rf['test_f1'].mean())
print("Mean ROC:", cv_rf['test_roc_auc'].mean())
print("Mean Log Loss:", cv_rf['test_neg_log_loss'].mean())
```

```
{'fit_time': array([0.31615424, 0.31910419, 0.3211751 , 0.31615305, 0.3219769
]), 'score_time': array([0.03291202, 0.03324795, 0.03387499, 0.03291273,
0.03691149]), 'test_accuracy': array([0.8125    , 0.76785714, 0.77678571,
0.82142857, 0.75892857]), 'test_neg_log_loss': array([-0.5512609 , -0.63940361,
-0.62787623, -0.44254006, -0.65758884]), 'test_f1': array([0.4       ,
0.27777778, 0.3902439 , 0.41176471, 0.27027027]), 'test_roc_auc':
array([0.66413043, 0.56576087, 0.60842391, 0.80543478, 0.60326087])}
Mean Accuracy: 0.7875
Mean F1: 0.35001133127388506
Mean ROC: 0.6494021739130434
Mean Log Loss: -0.5837339276642147
```

human interpretability: The human interpretability is better thasn both the 10 and 5 topic models, it is much more readable than those models.

perplexity: 15 topics: -6.908 vs 10 topics: -7.404

coherence: 15 topics: -17.971 vs 10 topics: -18.987

classification performance:
15 Topics:
Mean Accuracy: 0.7839285714285713
Mean F1: 0.257784917486056
Mean ROC: 0.6614130434782608
Mean Log Loss: -0.5294505446969614

VS.

5 Topics:
Mean Accuracy: 0.8107142857142857
Mean F1: 0.2939923628466454
Mean ROC: 0.6096739130434783
Mean Log Loss: -0.6767382105982549

VS.

10 Topics:
Mean Accuracy: 0.8071428571428572
Mean F1: 0.2180214166421063
Mean ROC: 0.6490217391304347
Mean Log Loss: -0.5865378903125963

Accuracy and F1 got worse while ROC and Log Loss did improved from the 5 Topics and 15 Topics.

## 5 Task 3

In addition to Random Forest, try another classifier of your choosing. How does this compare to the Random Forest?

[46]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC #Does not work why?
from sklearn.neural_network import MLPClassifier #Does not work why? works now
from sklearn.ensemble import AdaBoostClassifier

n_splits = 5

pred_vars = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,]


scoring = ['accuracy', 'neg_log_loss', 'f1', 'roc_auc']
kn_base = AdaBoostClassifier()

cv_kn = cross_validate(kn_base, classification_df[pred_vars],
 ↪classification_df['Restatement Topic'], cv=StratifiedShuffleSplit(n_splits),
 ↪scoring=scoring)
print(cv_kn)
print("Mean Accuracy:", cv_kn['test_accuracy'].mean())
print("Mean F1:", cv_kn['test_f1'].mean())
print("Mean ROC:", cv_kn['test_roc_auc'].mean())
print("Mean Log Loss:", cv_kn['test_neg_log_loss'].mean())
```

```
{'fit_time': array([0.0947454 , 0.08436894, 0.08178115, 0.08676839,
0.08326721]), 'score_time': array([0.02194142, 0.02048635, 0.01994681,
0.01998401, 0.01894975]), 'test_accuracy': array([0.79464286, 0.82142857,
0.80357143, 0.76785714, 0.83928571]), 'test_neg_log_loss': array([-0.68552591,
-0.68551032, -0.67388198, -0.67240269, -0.65578137]), 'test_f1':
array([0.14814815, 0.23076923, 0.26666667, 0.13333333, 0.30769231]),
'test_roc_auc': array([0.60679348, 0.61005435, 0.73478261, 0.5375    ,
0.64891304])}
Mean Accuracy: 0.8053571428571427
Mean F1: 0.21732193732193733
Mean ROC: 0.6276086956521739
Mean Log Loss: -0.6746204558253005
```

15 Topics Random Forest:
Mean Accuracy: 0.7875
Mean F1: 0.35001133127388506
Mean ROC: 0.6494021739130434
Mean Log Loss: -0.5837339276642147

As an alternative model I used an AdaBoost Classifier Model as alternative. This model had an higher accuracy but lower F1 and ROC scores and a worse Mean Log Loss. This would suggest that the AdaBoost Classifier has a worse False Positive rate than the Random Forrest.

[ ]:
```

```