In [8]:
```
import pandas as pd
import numpy as np
from sklearn import model_selection, preprocessing, linear_model, naive_bay
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorize
from sklearn import decomposition, ensemble
import textblob, string
```

In [2]:
```
# read the dataset as pandas dataframe

abst = pd.read_excel('abstract.xlsx')
```

In [3]:
```
# check the column types

abst.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4218 entries, 0 to 4217
Data columns (total 5 columns):
foodname    4218 non-null object
fu          2380 non-null object
ti          4218 non-null object
ab          4218 non-null object
rating1     4218 non-null int64
dtypes: int64(1), object(4)
memory usage: 164.9+ KB
```

In [4]:
```
# check the first 5 lines of dataframe

abst.head(5)
```

Out[4]:

	foodname	fu	ti	ab	rating1
0	TI = (sorghum OR milo OR durra OR jowari OR gr...	NaN	Pretreatment of Sweet Sorghum Bagasse for Etha...	(1) Background: Commercial production of fuel ...	99
1	TI = (wheat OR Triticum)	NaN	Metabolites of 4-n-nonylphenol in wheat cell s...	4-Nonylphenol, a metabolite of nonionic surfac...	99
2	TI = (sorghum OR milo OR durra OR jowari OR gr...	NaN	A sorghum xylanase inhibitor-like protein with...	A 25-kDa protein, with an N-terminal amino aci...	1
3	TI = (rice)	NaN	Molecular identification of yeast species asso...	A B S T R A C T In Manipur state of North-East...	99
4	TI = (corn OR maize OR Zea mays)	Monsanto Argentina S.A.	Fungal and mycotoxin contamination in Bt maize...	A Bt maize hybrid and its non-transgenic count...	99

In [5]:
```
# change the funding source's datatype to string

abst['fu'] = abst.fu.astype(str)
```

1. Write a script to identify whether the funding source is from industry or not (e.g. if pepsi is the

funder, it should be coded as industry). Explain your steps briefly in a readme document.

In [6]:
```
# add column 'fu_source', 1 indicates the funding source is from industry a

ind = [row for row in abst['fu'] if 'Inc.' in row]
abst_ind = abst[abst['fu'].isin(ind)]
abst['fu_source'] = np.where(abst['fu'].isin(ind), 1, 0)
```

In [7]:
```
# check the dataframe

abst.head(5)
```

Out[7]:

	foodname	fu	ti	ab	rating1	fu_source
0	TI = (sorghum OR milo OR durra OR jowari OR gr...	nan	Pretreatment of Sweet Sorghum Bagasse for Etha...	(1) Background: Commercial production of fuel ...	99	0
1	TI = (wheat OR Triticum)	nan	Metabolites of 4-n-nonylphenol in wheat cell s...	4-Nonylphenol, a metabolite of nonionic surfac...	99	0
2	TI = (sorghum OR milo OR durra OR jowari OR gr...	nan	A sorghum xylanase inhibitor-like protein with...	A 25-kDa protein, with an N-terminal amino aci...	1	0
3	TI = (rice)	nan	Molecular identification of yeast species asso...	A B S T R A C T In Manipur state of North-East...	99	0
4	TI = (corn OR maize OR Zea mays)	Monsanto Argentina S.A.	Fungal and mycotoxin contamination in Bt maize...	A Bt maize hybrid and its non-transgenic count...	99	0

In [9]:
```
abst.sum()
```

Out[9]:
```
foodname     TI = (sorghum OR milo OR durra OR jowari OR gr...
fu           nannannannanMonsanto Argentina S.A.CRTI [04-00...
ti           Pretreatment of Sweet Sorghum Bagasse for Etha...
ab           (1) Background: Commercial production of fuel ...
rating1                                                 315287
fu_source                                                   95
dtype: object
```

2. Write a machine learning script to train and classify abstracts. You can assume a binary coding for the rating (positive/not positive) for the ML script.

In [10]:
```
# check the values included in rating1 column

abst.rating1.value_counts()
```

Out[10]:
```
 99    3179
 1      672
 0      261
-1      106
Name: rating1, dtype: int64
```

In [11]:
```
# change the abstracts' datatype to string

abst['ab'] = abst.ab.astype(str)
```

In [12]:
```
# prepare the binary coding data for machine learning

data = abst[abst['rating1'].isin([-1, 1])]
```

In [13]:
```
# check the data

data
```

Out[13]:

	foodname	fu	ti	ab	rating1	fu_source
2	TI = (sorghum OR milo OR durra OR jowari OR gr...	nan	A sorghum xylanase inhibitor-like protein with...	A 25-kDa protein, with an N-terminal amino aci...	1	0
9	TI = (rice)	Genomics for Agricultural Innovation [PMI0004]...	Involvement of ethylene signaling in Azospiril...	A bacterial endophyte Azospirillum sp. B510 in...	1	0
21	TI = (wheat OR Triticum)	Advanced Food and Materials Network through op...	Diets Enriched in Oat Bran or Wheat Bran Tempo...	A clear understanding of how diet alters gastr...	-1	1
37	TI = (corn OR maize OR Zea mays)	Dina Food Industrial Group; BehAra Food Indus...	Determination of acrylamide level in popular I...	Acrylamide is a chemical found in starchy food...	-1	0
55	TI = (wheat OR Triticum)	Kuwaiti Flour Mills and Bakeries Company (Kuwa...	Efficacy of wheat-based biscuits fortified wit...	Adverse sensory changes prevent the addition o...	1	0
...	...	...	...	...	...	...
4202	TI = (wheat OR Triticum)	HarvestPlus Program; German Research Foundati...	Biofortification and Localization of Zinc in W...	Zinc (Zn) deficiency associated with low dieta...	1	0
4208	TI = (wheat OR Triticum)	Primary Industries Innovation Centre; NANO Ma...	Effect of beta-Glucan on Technological, Sensor...	beta-Glucan is known to have valuable properti...	1	0
4209	TI = (barley)	UNIK (Food, Fitness & Pharma for Health and Di...	Extracted Oat and Barley beta-Glucans Do Not A...	beta-Glucans are known to exhibit hypocholeste...	1	0
4212	TI = (rice)	Hansells Food Group, Auckland, New Zealand	Consumption of a plant sterol-based spread der...	fTo establish the effectiveness of a new phyto...	1	0
4217	TI = (corn OR maize OR Zea mays)	Indian Council of Medical ResearchIndian Counc...	Chemopreventive Effect of Different Ratios of ...	n-3 Polyunsaturated fatty acids (PUFA) have a ...	1	0

778 rows × 6 columns

In [14]:
```
# split the dataset into training and validation datasets

train_x, valid_x, train_y, valid_y = model_selection.train_test_split(data[
```

In [16]:
```
# create a count vectorizer object

count_vect = CountVectorizer(analyzer='word', token_pattern=r'\w{1,}')
count_vect.fit(data['ab'])

# transform the training and validation data using count vectorizer object

xtrain_count =  count_vect.transform(train_x)
xvalid_count =  count_vect.transform(valid_x)
```

In [17]:
```
# word level tf-idf

tfidf_vect = TfidfVectorizer(analyzer='word', token_pattern=r'\w{1,}', max_
tfidf_vect.fit(data['ab'])
xtrain_tfidf =  tfidf_vect.transform(train_x)
xvalid_tfidf =  tfidf_vect.transform(valid_x)

# ngram level tf-idf

tfidf_vect_ngram = TfidfVectorizer(analyzer='word', token_pattern=r'\w{1,}'
tfidf_vect_ngram.fit(data['ab'])
xtrain_tfidf_ngram =  tfidf_vect_ngram.transform(train_x)
xvalid_tfidf_ngram =  tfidf_vect_ngram.transform(valid_x)

# characters level tf-idf

tfidf_vect_ngram_chars = TfidfVectorizer(analyzer='char', token_pattern=r'\
tfidf_vect_ngram_chars.fit(data['ab'])
xtrain_tfidf_ngram_chars =  tfidf_vect_ngram_chars.transform(train_x)
xvalid_tfidf_ngram_chars =  tfidf_vect_ngram_chars.transform(valid_x)
```

3. Given your script, provide validation statistics, i.e. provide classifications of each abstract as positive or not from the script and compare against the actual data.

In [18]:
```
def train_model(classifier, feature_vector_train, label, feature_vector_val
    # fit the training dataset on the classifier
    classifier.fit(feature_vector_train, label)

    # predict the labels on validation dataset
    predictions = classifier.predict(feature_vector_valid)

    return metrics.accuracy_score(predictions, valid_y)
```

In [28]:
```
# Naive Bayes on Word Level TF IDF Vectors
accuracy = train_model(naive_bayes.MultinomialNB(), xtrain_tfidf, train_y,
print('The accuracy rate of NB, WordLevel TF-IDF is', accuracy)

# Naive Bayes on Ngram Level TF IDF Vectors
accuracy = train_model(naive_bayes.MultinomialNB(), xtrain_tfidf_ngram, tra
print('The accuracy rate of NB, N-Gram Vectors is', accuracy)

# Naive Bayes on Character Level TF IDF Vectors
accuracy = train_model(naive_bayes.MultinomialNB(), xtrain_tfidf_ngram_char
print('The accuracy rate of NB, CharLevel TF-IDF is', accuracy)
```

```
The accuracy rate of NB, WordLevel TF-IDF is 0.8871794871794871
The accuracy rate of NB, N-Gram Vectors is 0.9025641025641026
The accuracy rate of NB, CharLevel TF-IDF is 0.8871794871794871
```

In [27]:
```
# Linear Classifier on Count Vectors
accuracy = train_model(linear_model.LogisticRegression(), xtrain_count, tra
print('The accuracy rate of LR, WordLevel TF-IDF is', accuracy)

# Linear Classifier on Ngram Level TF IDF Vectors
accuracy = train_model(linear_model.LogisticRegression(), xtrain_tfidf_ngra
print('The accuracy rate of LR, N-Gram Vectors is', accuracy)

# Linear Classifier on Word Level TF IDF Vectors
accuracy = train_model(linear_model.LogisticRegression(), xtrain_tfidf, tra
print('The accuracy rate of LR, CharLevel TF-IDF is', accuracy)
```

```
The accuracy rate of LR, WordLevel TF-IDF is 0.9333333333333333
The accuracy rate of LR, N-Gram Vectors is 0.8923076923076924
The accuracy rate of LR, CharLevel TF-IDF is 0.8923076923076924

/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.
py:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22.
Specify a solver to silence this warning.
  FutureWarning)
```

In [29]:
```
# SVM on Ngram Level TF IDF Vectors
accuracy = train_model(svm.SVC(), xtrain_count, train_y, xvalid_count)
print('The accuracy rate of SVM, WordLevel TF-IDF is', accuracy)

# SVM on Ngram Level TF IDF Vectors
accuracy = train_model(svm.SVC(), xtrain_tfidf_ngram, train_y, xvalid_tfidf
print('The accuracy rate of SVM, WordLevel TF-IDF is', accuracy)

# SVM on Ngram Level TF IDF Vectors
accuracy = train_model(svm.SVC(), xtrain_tfidf, train_y, xvalid_tfidf)
print('The accuracy rate of SVM, WordLevel TF-IDF is', accuracy)
```

```
/opt/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:193: Futur
eWarning: The default value of gamma will change from 'auto' to 'scale' i
n version 0.22 to account better for unscaled features. Set gamma explici
tly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)

The accuracy rate of SVM, WordLevel TF-IDF is 0.8871794871794871
The accuracy rate of SVM, WordLevel TF-IDF is 0.8871794871794871
The accuracy rate of SVM, WordLevel TF-IDF is 0.8871794871794871

/opt/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:193: Futur
eWarning: The default value of gamma will change from 'auto' to 'scale' i
n version 0.22 to account better for unscaled features. Set gamma explici
tly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
/opt/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:193: Futur
eWarning: The default value of gamma will change from 'auto' to 'scale' i
n version 0.22 to account better for unscaled features. Set gamma explici
tly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

4. Comment on how you would improve the ML script given more time and computational resources.

As we can see from data preprocessing part, I use 3 different tf-idf methods (word level, ngram, and character level) after converting the collection of abstraction text to a matrix of token counts. In terms of improving the ML performance, I think we could do it in a few ways. Firstly, here we only have 'abstract' as explantory variable, if we could include more related features, we will presumably increase the ML model performace. Besides, here I include Naive Bayes, Linear, and SVM as ML models, there are other boosting models like XGBoost or LightGBM could potenially help to improve the overall performance as well. But we should definitely add more features as possible in order to achieve better results.

In [ ]: