Assembling Data

```
#importing packages
import nltk
nltk.download('gutenberg')
nltk.download('punkt')
import re
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
nltk.download('omw-1.4')
nltk.download('wordnet')
import pandas as pd
import string
```

```
[nltk_data] Downloading package gutenberg to /root/nltk_data...
[nltk_data]   Unzipping corpora/gutenberg.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data] Downloading package wordnet to /root/nltk_data...
```

```
nltk.corpus.gutenberg.fileids()
```

```
['austen-emma.txt',
 'austen-persuasion.txt',
 'austen-sense.txt',
 'bible-kjv.txt',
 'blake-poems.txt',
 'bryant-stories.txt',
 'burgess-busterbrown.txt',
 'carroll-alice.txt',
 'chesterton-ball.txt',
 'chesterton-brown.txt',
 'chesterton-thursday.txt',
 'edgeworth-parents.txt',
 'melville-moby_dick.txt',
 'milton-paradise.txt',
 'shakespeare-caesar.txt',
 'shakespeare-hamlet.txt',
 'shakespeare-macbeth.txt',
 'whitman-leaves.txt']
```

```
#preprocessing
#grabbing the Shakespeare works

caesar = nltk.corpus.gutenberg.raw('shakespeare-caesar.txt')
hamlet = nltk.corpus.gutenberg.raw('shakespeare-hamlet.txt')
mcbeth = nltk.corpus.gutenberg.raw('shakespeare-macbeth.txt')

#lowercasing everything

c_lower = caesar.lower()
h_lower = hamlet.lower()
m_lower = mcbeth.lower()

#tokenizing the text blobs

from nltk.tokenize import sent_tokenize
caesar_tok = sent_tokenize(c_lower)
hamlet_tok = sent_tokenize(h_lower)
mcbeth_tok = sent_tokenize(m_lower)

#https://www.guru99.com/tokenize-words-sentences-nltk.html
#https://www.nltk.org/howto/corpus.html
```

```
#removing punctuation
#caesar_tok = re.sub('[%s]' % re.escape(string.punctuation), '', caesar_tok)

def remove_punct(str_list):
  no_punct = []

  for sent in str_list:
    sent = re.sub('[%s]' % re.escape(string.punctuation), '', sent)

```

```
    no_punct.append(sent)
  return no_punct
```

```
caesar_tok = remove_punct(caesar_tok)
hamlet_tok = remove_punct(hamlet_tok)
mcbeth_tok = remove_punct(mcbeth_tok)
```

```
print(caesar_tok)
```

```
['the tragedie of julius caesar by william shakespeare 1599\n\n\nactus primus', 'scoena prima', 'enter flauius murellus and certaine com
```

```
#checking length

len(caesar)
len(hamlet)
len(mcbeth)

#average length of a play is 125,000 lines (aka sentences)
```

```
100351
```

```
#Make Shakespeare plays into dataframes
import pandas as pd
ceasar_df = pd.DataFrame(caesar_tok)
author = 'Shakespeare'
ceasar_df['Author'] = author
ceasar_df = ceasar_df.rename(columns={0: 'Text'})

hamlet_df = pd.DataFrame(hamlet_tok)
hamlet_df['Author'] = author
hamlet_df = hamlet_df.rename(columns={0: 'Text'})

mcbeth_df = pd.DataFrame(mcbeth_tok)
mcbeth_df['Author'] = author
mcbeth_df = mcbeth_df.rename(columns={0: 'Text'})

print(ceasar_df)
```

```
                                                   Text       Author
0     the tragedie of julius caesar by william shake...  Shakespeare
1                                         scoena prima  Shakespeare
2     enter flauius murellus and certaine commoners ...  Shakespeare
3                                               flauius  Shakespeare
4     hence home you idle creatures get you home\nis...  Shakespeare
...                                                 ...          ...
1547  within my tent his bones to night shall ly\nmo...  Shakespeare
1548                                            exeunt  Shakespeare
1549                                             omnes  Shakespeare
1550                                             finis  Shakespeare
1551                      the tragedie of ivlivs caesar  Shakespeare

[1552 rows x 2 columns]
```

Non-Shakespeare Corpus Creation

```
#creating a non-Shakeaspeare corpus

bible = nltk.corpus.gutenberg.raw('bible-kjv.txt')
milton = nltk.corpus.gutenberg.raw('milton-paradise.txt')
blake = nltk.corpus.gutenberg.raw('blake-poems.txt')

b_lower = bible.lower()
mil_lower = milton.lower()
bl_lower = blake.lower()

#tokeninzing the text-blob

bible_tok = sent_tokenize(b_lower)
milton_tok = sent_tokenize(mil_lower)
blake_tok = sent_tokenize(bl_lower)

bible_tok = remove_punct(bible_tok)
```

```
milton_tok = remove_punct(milton_tok)
blake_tok = remove_punct(blake_tok)
```

```
#making the non-Shakespeare all one dataframe.

author_2 = 'Other'

bible_df = pd.DataFrame(bible_tok)
bible_df['Author'] = author_2
bible_df = bible_df.rename(columns={0: 'Text'})

milton_df = pd.DataFrame(milton_tok)
milton_df['Author'] = author_2
milton_df = milton_df.rename(columns={0: 'Text'})

blake_df = pd.DataFrame(blake_tok)
blake_df['Author'] = author_2
blake_df = blake_df.rename(columns={0: 'Text'})

noshakespear = bible_df.append([milton_df, blake_df], ignore_index=True)
print(noshakespear)
len(noshakespear)
```

```
                                                    Text Author
0      the king james bible\n\nthe old testament of t...  Other
1      12 and the earth was without form and void and...  Other
2      and the spirit of god moved upon the face of t...  Other
3      13 and god said let there be light and there w...  Other
4      14 and god saw the light that it was good and ...  Other
...                                                  ...    ...
31994   why a tongue impressd with honey from every wind  Other
31995  why an ear a whirlpool fierce to draw creation...  Other
31996  why a nostril wide inhaling terror trembling  ...  Other
31997  why a little curtain of flesh on the bed of ou...  Other
31998  the virgin started from her seat  with a shrie...  Other

[31999 rows x 2 columns]
<ipython-input-8-828362e671fb>:17: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future vers
  noshakespear = bible_df.append([milton_df, blake_df], ignore_index=True)
31999
```

```
#subsampling No Shakespeare

import random
random.seed(1)

noshakes_sub = noshakespear.sample(n = 3500, random_state=1)
len(noshakes_sub)
print(noshakes_sub)

#https://www.geeksforgeeks.org/how-to-randomly-select-rows-from-pandas-dataframe/
#https://www.w3schools.com/python/ref_random_seed.asp
#https://stackoverflow.com/questions/46638641/how-to-fix-valueerror-expected-2d-array-got-1d-array-instead-in-sklearn-pyth
```

```
                                                    Text Author
13667  3111 i was a reproach among all mine enemies b...  Other
1978   1922 and let the priests also which come near ...  Other
1428   4914 issachar is a strong ass couching down be...  Other
17621  3818 for the grave cannot praise thee death ca...  Other
3203   2637 and they shall fall one upon another as i...  Other
...                                                  ...    ...
26792  219 and the same man had four daughters virgin...  Other
22408       verily i say unto you they\nhave their reward  Other
7435   524 and let it be when thou hearest the sound ...  Other
13147  3918 what time she lifteth up herself on high ...  Other
26311  733 then said the lord to him put off thy shoe...  Other

[3500 rows x 2 columns]
```

```
#composite data frames

no_ceasar = hamlet_df.append([noshakes_sub, mcbeth_df], ignore_index=True)
no_hamlet = ceasar_df.append([noshakes_sub, mcbeth_df], ignore_index=True)
no_mcbeth = hamlet_df.append([noshakes_sub, ceasar_df], ignore_index=True)
```

```
<ipython-input-10-47ee7286166d>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future vers
  no_ceasar = hamlet_df.append([noshakes_sub, mcbeth_df], ignore_index=True)
```

```
<ipython-input-10-47ee7286166d>:4: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future vers
  no_hamlet = ceasar_df.append([noshakes_sub, mcbeth_df], ignore_index=True)
<ipython-input-10-47ee7286166d>:5: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future vers
  no_mcbeth = hamlet_df.append([noshakes_sub, ceasar_df], ignore_index=True)
```

```
display(no_ceasar.shape)
no_ceasar["Author"].value_counts()/no_ceasar.shape[0]

#move these down!
```

```
(7273, 2)
Shakespeare    0.518768
Other          0.481232
Name: Author, dtype: float64
```

```
display(no_hamlet.shape)
no_hamlet["Author"].value_counts()/no_hamlet.shape[0]
```

```
(6472, 2)
Other          0.540791
Shakespeare    0.459209
Name: Author, dtype: float64
```

```
display(no_mcbeth.shape)
no_mcbeth["Author"].value_counts()/no_mcbeth.shape[0]
```

```
(7405, 2)
Shakespeare    0.527346
Other          0.472654
Name: Author, dtype: float64
```

N-Grams

Preprocessing

```
from nltk.corpus import stopwords
nltk.download('stopwords')
import string
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
#cleaning text: remove stopwards, remove non-text characters
def clean_text(str_list, lemmatize = False):
  clean_list = []

  for text in str_list:
    text = re.sub('[^a-zA-Z]', ' ', text)
    text = re.sub(r'\s+', ' ', text)
    text = re.sub(r'[^\w\s]', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    words = word_tokenize(text)
    clean_words = []

    for word in words:
      if word not in stopwords.words('english'):
        if (len(word) > 1) and (re.match(r'^\w+$', word)):

         clean_words.append(word)
    clean_text = ' '.join(clean_words)
    clean_list.append(clean_text)

  return clean_list

  #https://datagy.io/python-remove-punctuation-from-string/
  #https://stackoverflow.com/questions/265960/best-way-to-strip-punctuation-from-a-string
```

```
c_ngrams = clean_text(ceasar_df['Text'])
h_ngrams = clean_text(hamlet_df['Text'])
m_ngrams = clean_text(mcbeth_df['Text'])
n_ngrams = clean_text(noshakes_sub['Text'])
```

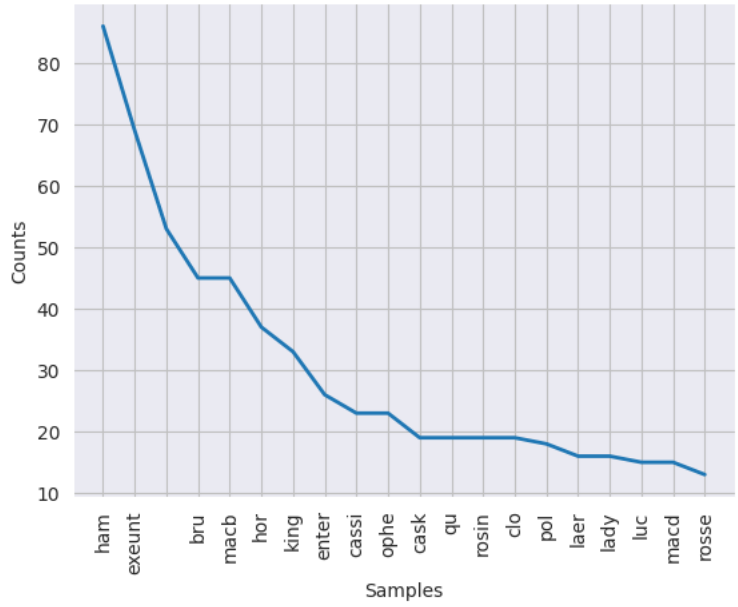
```
5
6 #there's a bunch of random blanks???
```

```
1 print(c_ngrams)
```

```
['tragedie julius caesar william shakespeare actus primus', 'scoena prima', 'enter flauius murellus certaine commoners ouer stage', 'fla
```

```
1 #most common words in all Shakespeare
2
3 comb_shakes = ceasar_df.append([hamlet_df, mcbeth_df], ignore_index=True)
4 comb_shakes = clean_text(comb_shakes['Text'])
5
6 from nltk.probability import FreqDist
7
8 fdist_comb_shakes = FreqDist(comb_shakes)
9 fdist_comb_shakes1 = fdist_comb_shakes.most_common(20)
10
11 print(fdist_comb_shakes)
12 print(fdist_comb_shakes1)
13
14 import seaborn as sns
15 sns.set_style('darkgrid')
16 nlp_words=nltk.FreqDist(fdist_comb_shakes)
17 nlp_words.plot(20);
```

```
<ipython-input-17-687fa6e03aba>:1: FutureWarning: The frame.append method is depre
  comb_shakes = ceasar_df.append([hamlet_df, mcbeth_df], ignore_index=True)
<FreqDist with 4317 samples and 5325 outcomes>
[('ham', 86), ('exeunt', 69), ('', 53), ('bru', 45), ('macb', 45), ('hor', 37), ('
```

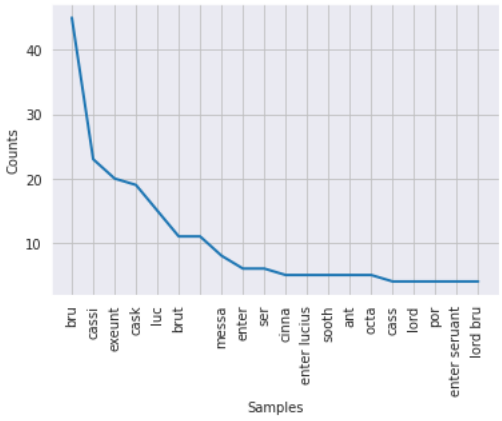



```
1 #counting the most common words in Julius Caesar
2
3 from nltk.probability import FreqDist
4
5 fdist_c = FreqDist(c_ngrams)
6 fdist_c1 = fdist_c.most_common(20)
7 fdist_c
8
9 #https://towardsai.net/p/data-mining/text-mining-in-python-steps-and-examples-78b3f8fd913b
```

```
FreqDist({'bru': 45, 'cassi': 23, 'exeunt': 20, 'cask': 19, 'luc': 15, 'brut': 11, '': 11, 'messa': 8, 'enter': 6, 'ser': 6, ...})
```

```
1 #graphing the frequency distribution
2
3 import seaborn as sns
4 sns.set_style('darkgrid')
5 nlp_words=nltk.FreqDist(c_ngrams)
6 nlp_words.plot(20);
```

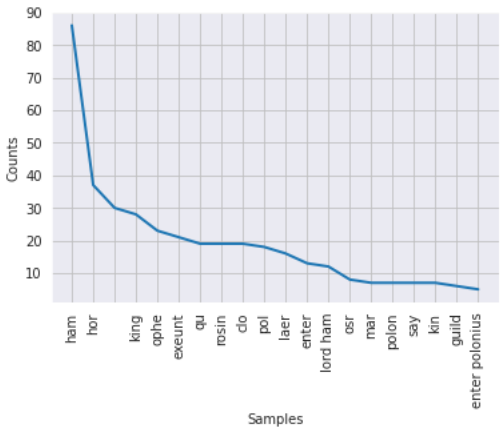
```
7
8 #https://www.milindsoorya.com/blog/introduction-to-word-frequencies-in-nlp#import-the-needed-libraries
```




```
1 #counting the most common words in The Tragedy of Hamlet
2
3 fdist_h = FreqDist(h_ngrams)
4 fdist_h1 = fdist_h.most_common(20)
5 fdist_h
```

```
FreqDist({'ham': 86, 'hor': 37, '': 30, 'king': 28, 'ophe': 23, 'exeunt': 21, 'qu': 19, 'rosin': 19, 'clo': 19, 'pol': 18, ...})
```

```
1 sns.set_style('darkgrid')
2 nlp_words=nltk.FreqDist(h_ngrams)
3 nlp_words.plot(20);
```




```
1 #counting the most common words in McBeth
2
3 fdist_m = FreqDist(m_ngrams)
4 fdist_m1 = fdist_m.most_common(20)
5 fdist_m
```

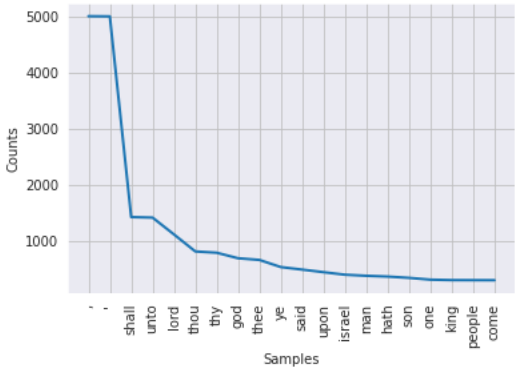
```
FreqDist({'macb': 45, 'exeunt': 28, 'lady': 15, 'macd': 15, 'rosse': 13, '': 12, 'mal': 10, 'lenox': 9, 'wife': 8, 'enter': 7, ...})
```

```
1 sns.set_style('darkgrid')
2 nlp_words=nltk.FreqDist(m_ngrams)
3 nlp_words.plot(20);
```

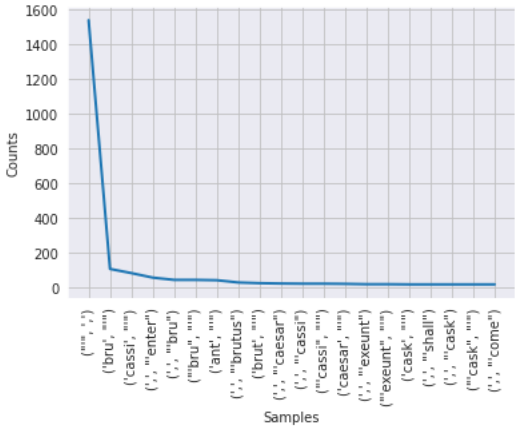

```
#counting the most common words in Non-Shakespeare

n_ngrams2 = word_tokenize(str(n_ngrams))
fdist_n = FreqDist(n_ngrams2)
fdist_n1 = fdist_n.most_common(20)
fdist_n
```

```
FreqDist({',': 4999, "'": 4996, 'shall': 1426, 'unto': 1417, 'lord': 1118, 'thou': 815, 'thy': 792, 'god': 694, 'thee': 664, 'ye': 537,
...})
```

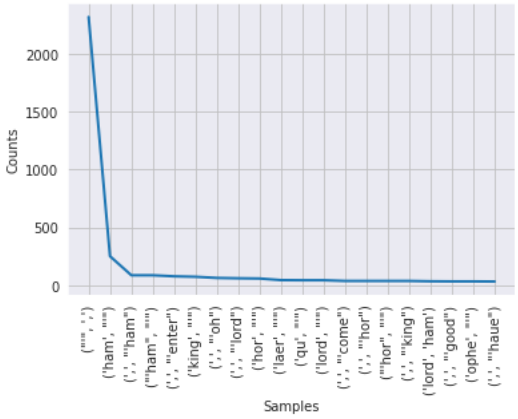
```
sns.set_style('darkgrid')
nlp_words=nltk.FreqDist(n_ngrams2)
nlp_words.plot(20);
```



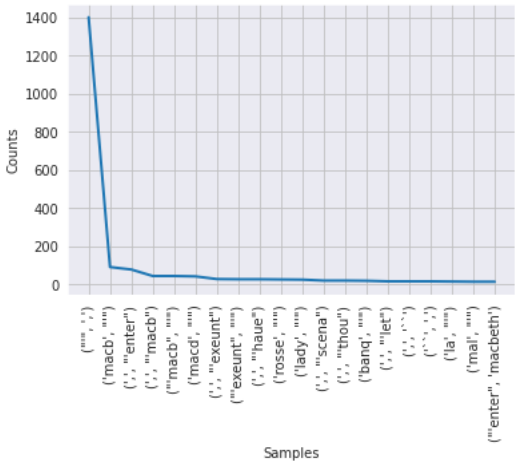

```
#redoing the frequency distributions with bigrams
#Ceasar

c_ngrams2 = word_tokenize(str(c_ngrams))
c_bigram = list(nltk.bigrams(c_ngrams2))

sns.set_style('darkgrid')
nlp_words=nltk.FreqDist(c_bigram)
nlp_words.plot(20);

#https://www.tutorialspoint.com/python_text_processing/python_bigrams.htm
```



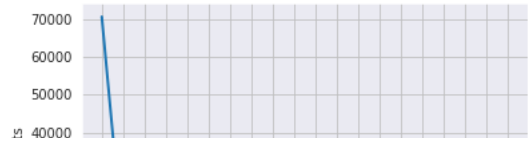
```
#Hamlet

h_ngrams2 = word_tokenize(str(h_ngrams))
h_bigram = list(nltk.bigrams(h_ngrams2))

sns.set_style('darkgrid')
nlp_words=nltk.FreqDist(h_bigram)
nlp_words.plot(20);
```




```
#McBeth

m_ngrams2 = word_tokenize(str(m_ngrams))
m_bigram = list(nltk.bigrams(m_ngrams2))

sns.set_style('darkgrid')
nlp_words=nltk.FreqDist(m_bigram)
nlp_words.plot(20);
```




```
#Non-Shakespeare

n_ngrams2 = word_tokenize(str(n_ngrams))
n_bigram = list(nltk.bigrams(n_ngrams2))

sns.set_style('darkgrid')
nlp_words=nltk.FreqDist(n_bigram)
nlp_words.plot(20);
```



Classification Models - Set up


```
#we may need to do either one-class training:

#https://machinelearningmastery.com/one-class-classification-algorithms/
#https://towardsdatascience.com/one-class-neural-network-in-keras-249ff56201c0
#https://stackoverflow.com/questions/57309958/one-class-classification-using-keras-and-python
#https://medium.com/geekculture/sklearn-expects-data-to-be-in-shape-64fbcaf80a8c
```



```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction import _stop_words
from sklearn.model_selection import cross_validate, KFold, GridSearchCV
```

```
# convert label to a numerical variable
no_ceasar['Author'] = no_ceasar['Author'].map({'Shakespeare':1, 'Other':0})
no_ceasar.shape
print(no_ceasar)

no_hamlet['Author'] = no_hamlet['Author'].map({'Shakespeare':1, 'Other':0})
no_mcbeth['Author'] = no_mcbeth['Author'].map({'Shakespeare':1, 'Other':0})
```

```
                                                   Text  Author
0     the tragedie of hamlet by william shakespeare ...       1
1                                          scoena prima       1
2            enter barnardo and francisco two centinels       1
3                                              barnardo       1
4                                            whos there       1
...                                                 ...     ...
7268  this and what need full else\nthat calls vpon ...       1
7269                                           flourish       1
7270                                         exeunt omnes       1
7271                                                finis       1
7272                             the tragedie of macbeth       1

[7273 rows x 2 columns]
```

```
#define predictor and response variables
c_X = no_ceasar[['Text']]
c_y = no_ceasar['Author']

h_X = no_hamlet[['Text']]
h_y = no_hamlet['Author']

m_X = no_mcbeth[['Text']]
m_y = no_mcbeth['Author']

#define cross-validation method to use
cv = KFold(n_splits=10, random_state=1, shuffle=True)

#https://www.statology.org/k-fold-cross-validation-in-python/
```

```
#check proportion of shakespeare and non-shakespeare
```

```
#Test Train - The old Fashioned Way

from sklearn.model_selection import train_test_split

c_X = no_ceasar.Text
c_y = no_ceasar.Author
print(c_X.shape, c_y.shape)

c_X_train, c_X_test, c_y_train, c_y_test = train_test_split(c_X, c_y, random_state=1)
print(c_X_train.shape, c_y_train.shape)
print(c_X_test.shape, c_y_test.shape)

#Train/Test and Vectorizing  for Hamlet

```

```
h_X = no_hamlet.Text
h_y = no_hamlet.Author
print(h_X.shape, h_y.shape)

h_X_train, h_X_test, h_y_train, h_y_test = train_test_split(h_X, h_y, random_state=1)
print(h_X_train.shape, h_y_train.shape)
print(h_X_test.shape, h_y_test.shape)

vect = CountVectorizer()
nohamlet_train_vect = vect.fit_transform(h_X_train)
nohamlet_test_vect = vect.transform(h_X_test)
print(nohamlet_train_vect.shape, nohamlet_test_vect.shape)

#Train/Test and Vectorizing for McBeth

m_X = no_mcbeth.Text
m_y = no_mcbeth.Author
print(m_X.shape, m_y.shape)

m_X_train, m_X_test, m_y_train, m_y_test = train_test_split(m_X, m_y, random_state=1)
print(m_X_train.shape, m_y_train.shape)
print(m_X_test.shape, m_y_test.shape)

vect = CountVectorizer()
nomcbeth_train_vect = vect.fit_transform(m_X_train)
nomcbeth_test_vect = vect.transform(m_X_test)
print(nomcbeth_train_vect.shape, nomcbeth_test_vect.shape)
```

```
(7273,) (7273,)
(5454,) (5454,)
(1819,) (1819,)
(6472,) (6472,)
(4854,) (4854,)
(1618,) (1618,)
(4854, 10460) (1618, 10460)
(7405,) (7405,)
(5553,) (5553,)
(1852,) (1852,)
(5553, 11144) (1852, 11144)
```

```
#Vectorize

vect = CountVectorizer(min_df=0.29)

#ceasar
noceasar_train_vect = vect.fit_transform(c_X_train)
noceasar_test_vect = vect.transform(c_X_test)
print(noceasar_train_vect.shape, noceasar_test_vect.shape)

#hamlet
nohamlet_train_vect = vect.fit_transform(h_X_train)
nohamlet_test_vect = vect.transform(h_X_test)
print(nohamlet_train_vect.shape, nohamlet_test_vect.shape)

#mcbeth
nomcbeth_train_vect = vect.fit_transform(m_X_train)
nomcbeth_test_vect = vect.transform(m_X_test)
print(nomcbeth_train_vect.shape, nomcbeth_test_vect.shape)

#consider taking down some of the features pca? but what specifically are we vectorizing here
#https://towardsdatascience.com/basics-of-countvectorizer-e26677900f9c
```

```
(5454, 3) (1819, 3)
(4854, 3) (1618, 3)
(5553, 3) (1852, 3)
```

Naive Bayes Classification Model

```
pip install --upgrade scikit-learn
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.9/dist-packages (1.2.2)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.9/dist-packages (from scikit-learn) (1.22.4)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.9/dist-packages (from scikit-learn) (3.1.0)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.9/dist-packages (from scikit-learn) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.9/dist-packages (from scikit-learn) (1.2.0)
```

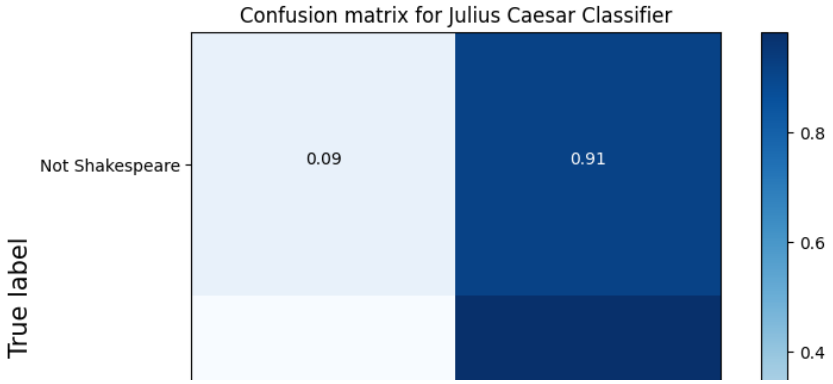
```
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import matplotlib as mpl
import matplotlib.cm as cm
import itertools
from sklearn.metrics import roc_auc_score
from sklearn import metrics
import numpy as np
```

```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label',fontsize=15)
    plt.xlabel('Predicted label',fontsize=15)
```

```
#experimental sets

caesar_experiment = vect.transform(caesar_tok)
hamlet_experiment = vect.transform(hamlet_tok)
mcbeth_experiment = vect.transform(mcbeth_tok)
```

```
# Step 3: Train the classifier and predict for test data
nb_c = MultinomialNB()
nb_c.fit(noceasar_train_vect, c_y_train)
noceasar_pred_class = nb_c.predict(noceasar_test_vect)
```

```
#graph

ceasar_matrix = confusion_matrix(c_y_test, noceasar_pred_class) #the order probably doesn't matter, but be consistent
plt.figure(figsize=(8,6))
plot_confusion_matrix(ceasar_matrix, classes=['Not Shakespeare','Shakespeare'],normalize=True,
                      title='Confusion matrix for Julius Caesar Classifier')

#check order of first row? should y be first or pred class?
```




```
from sklearn.metrics import (accuracy_score, f1_score)

accuray_c = accuracy_score(noceasar_pred_class, c_y_test)
f1_c = f1_score(noceasar_pred_class, c_y_test, average="weighted")

print("Accuracy:", accuray_c)
print("F1 Score:", f1_c)

#difference in accuracy is because you're calling "weighted", remove average = weighted and see what happens
#check why this is really good a finding not shakespeare and not finding anything else
#seriously check data balance
```

```
Accuracy: 0.5442550852116548
F1 Score: 0.6606056208769694
```

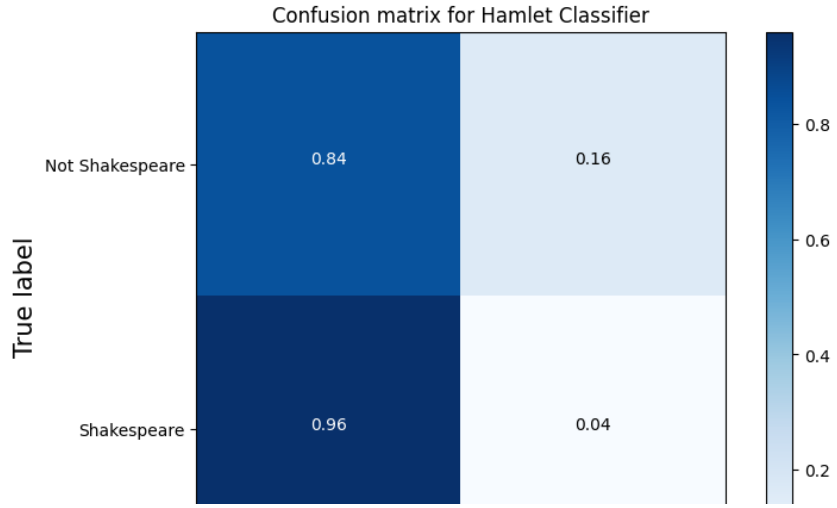
```
#predict

ceasar_prediction = nb_c.predict(caesar_experiment)
```

```
from numpy import count_nonzero

count_c_other = np.count_nonzero(ceasar_prediction == 0)
count_c_shakes = np.count_nonzero(ceasar_prediction == 1)

print(count_c_shakes)
print(count_c_other)

#https://www.tutsmake.com/python-count-the-occurrences-in-an-array/
#https://www.datacamp.com/tutorial/naive-bayes-scikit-learn
```

```
1541
11
```

```
#hamlet

nb_h = MultinomialNB()
nb_h.fit(nohamlet_train_vect, h_y_train)
nohamlet_pred_class = nb_h.predict(nohamlet_test_vect)
```

```
#graph

hamlet_matrix = confusion_matrix(h_y_test, nohamlet_pred_class)
plt.figure(figsize=(8,6))
plot_confusion_matrix(hamlet_matrix, classes=['Not Shakespeare','Shakespeare'],normalize=True,
                      title='Confusion matrix for Hamlet Classifier')
```




```
accuray_h = accuracy_score(nohamlet_pred_class, h_y_test)
f1_h = f1_score(nohamlet_pred_class, h_y_test, average="weighted")

print("Accuracy:", accuray_h)
print("F1 Score:", f1_h)
```

```
Accuracy: 0.48022249690976515
F1 Score: 0.578537079893798
```

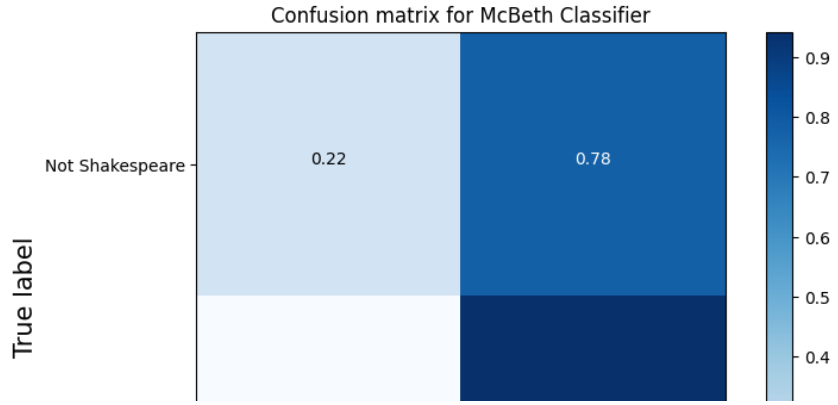
```
#predict

hamlet_prediction = nb_h.predict(hamlet_experiment)

#at min_def vect = 0.28 or lower, hamlet stops working???
```

```
count_h_other = np.count_nonzero(hamlet_prediction == 0)
count_h_shakes = np.count_nonzero(hamlet_prediction == 1)

print(count_h_shakes)
print(count_h_other)
```

```
75
2278
```

```
#mcbeth

nb_m = MultinomialNB()
nb_m.fit(nomcbeth_train_vect, m_y_train)
nomcbeth_pred_class = nb_m.predict(nomcbeth_test_vect)
```

```
#graph

mcbeth_matrix = confusion_matrix(m_y_test, nomcbeth_pred_class)
plt.figure(figsize=(8,6))
plot_confusion_matrix(mcbeth_matrix, classes=['Not Shakespeare','Shakespeare'],normalize=True,
                      title='Confusion matrix for McBeth Classifier')
```




```
accuray_m = accuracy_score(nomcbeth_pred_class, m_y_test)
f1_m = f1_score(nomcbeth_pred_class, m_y_test, average="weighted")

print("Accuracy:", accuray_m)
print("F1 Score:", f1_m)
```

```
Accuracy: 0.588012958963283
F1 Score: 0.6497792842010548
```

```
#predict

mcbeth_prediction = nb_m.predict(mcbeth_experiment)
```

```
count_m_other = np.count_nonzero(mcbeth_prediction == 0)
count_m_shakes = np.count_nonzero(mcbeth_prediction == 1)

print(count_m_shakes)
print(count_m_other)
```

```
1347
73
```

```
#metrics?

auc = nomcbeth_predclass['test_roc_auc']
accuracy = nomcbeth_predclass['test_accuracy']
f1 = nomcbeth_predclass['test_f1']
precision = nomcbeth_predclass['test_precision']
recall = nomcbeth_predclass['test_recall']
estimators = nomcbeth_predclass['estimator']

print(accuracy)

print(precision)

print(recall)

print(f1)
```

```
#graphs

#remake the confusion matrixes from below
#resource: https://colab.research.google.com/drive/1Hqj4E8SSrq5rMLvKamDPEKB_LikRKPLG
#https://stackoverflow.com/questions/49806790/iterable-over-raw-text-documents-expected-string-object-received
#https://stackoverflow.com/questions/9884132/what-are-iterator-iterable-and-iteration/9884501#9884501
#https://realpython.com/logistic-regression-python/
```

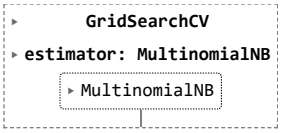
Cross Validation

```
m_X2 = no_mcbeth.Text
m_y2 = no_mcbeth.Author
mx_cv = vect.fit_transform(m_X2)
```

```
#mcbet
#nomcbeth_train_vect = vect.fit_transform(m_X_train)
```

```
#nomcbeth_test_vect = vect.transform(m_X_test)
#print(nomcbeth_train_vect.shape, nomcbeth_test_vect.shape)


#print(m_X.shape, m_y.shape)

from sklearn.model_selection import StratifiedKFold

params = {}

#gridsearch searches for the best hyperparameters and keeps the classifier with the highest recall score
skf = StratifiedKFold(n_splits=3)

nb2nb_m = GridSearchCV(MultinomialNB(), cv=skf, param_grid=params)
nb2nb_m.fit(mx_cv, m_y2)

#print(y_scores_nb2)

#https://stackoverflow.com/questions/51194627/python-naive-bayes-with-cross-validation-using-gaussiannb-classifier
```

```
CPU times: user 32 ms, sys: 0 ns, total: 32 ms
Wall time: 151 ms
```




```
scores = cross_validate(nb2nb_m,
                        X = mx_cv,
                        y = no_mcbeth.Author,
                        cv = skf,
                        scoring = ['accuracy', 'f1', 'precision'],
                        return_estimator = True)
```

```
accuracy = scores['test_accuracy']
f1 = scores['test_f1']
precision = scores['test_precision']
#recall = scores['recall']

print(accuracy)
print(f1)
print(precision)
```

```
[0.6200891 0.5952188 0.5684765]
[0.72460364 0.69458881 0.70797916]
[0.5865019  0.57694261 0.55029838]
```

```
cv_mb_pred = nb2nb_m.predict(mcbeth_experiment)
```

```
cv_mb_pred

count_m_other2 = np.count_nonzero(cv_mb_pred == 0)
count_m_shakes2 = np.count_nonzero(cv_mb_pred == 1)

print(count_m_shakes2)
print(count_m_other2)
```

```
1351
69
```

```
#Ceasar

c_X2 = no_ceasar.Text
c_y2 = no_ceasar.Author
cx_cv = vect.fit_transform(c_X2)

nb2nb_c = GridSearchCV(MultinomialNB(), cv=skf, param_grid=params)
nb2nb_c.fit(cx_cv, c_y2)

scores = cross_validate(nb2nb_c,
                        X = cx_cv,
                        y = no_ceasar.Author,
                        cv = skf,
```

```
                        scoring = ['accuracy', 'f1', 'precision'],
                        return_estimator = True)

accuracy = scores['test_accuracy']
f1 = scores['test_f1']
precision = scores['test_precision']
#recall = scores['recall']

print(accuracy)
print(f1)
print(precision)
```

```
[0.52618557 0.56683168 0.57343234]
[0.68649386 0.69862227 0.69924375]
[0.52264229 0.54672058 0.55112334]
```

```
cv_c_pred = nb2nb_c.predict(caesar_experiment)

cv_c_pred

count_c_other2 = np.count_nonzero(cv_c_pred == 0)
count_c_shakes2 = np.count_nonzero(cv_c_pred == 1)

print(count_c_shakes2)
print(count_c_other2)
```

```
1543
9
```

```
#Hamlet

h_X2 = no_hamlet.Text
h_y2 = no_hamlet.Author
hx_cv = vect.fit_transform(h_X2)

nb2nb_h = GridSearchCV(MultinomialNB(), cv=skf, param_grid=params)
nb2nb_h.fit(hx_cv, h_y2)

scores = cross_validate(nb2nb_h,
                        X = hx_cv,
                        y = no_hamlet.Author,
                        cv = skf,
                        scoring = ['accuracy', 'f1', 'precision'],
                        return_estimator = True)

accuracy = scores['test_accuracy']
f1 = scores['test_f1']
precision = scores['test_precision']
#recall = scores['recall']

print(accuracy)
print(f1)
print(precision)

```

```
[0.53429101 0.472879   0.45155308]
[0.00593472 0.04694049 0.0497992 ]
[0.15       0.13861386 0.12156863]
```

```
cv_h_pred = nb2nb_m.predict(hamlet_experiment)

cv_h_pred

count_h_other2 = np.count_nonzero(cv_h_pred == 0)
count_h_shakes2 = np.count_nonzero(cv_h_pred == 1)

print(count_h_shakes2)
print(count_h_other2)
```

```
2245
108
```

Resouces Used

https://www.datasciencebytes.com/bytes/2014/12/30/topic-modeling-of-shakespeare-characters/ https://www.geeksforgeeks.org/adding-new-column-to-existing-dataframe-in-pandas/ https://www.geeksforgeeks.org/different-ways-to-create-pandas-dataframe/
https://stackoverflow.com/questions/11346283/renaming-column-names-in-pandas https://freelancedatascientist.net/fast-stylometry-tutorial/
https://towardsdatascience.com/natural-language-processing-count-vectorization-with-scikit-learn-e7804269bb5e
https://stackoverflow.com/questions/65651544/nameerror-name-plot-confusion-matrix-is-not-defined
https://stackoverflow.com/questions/67149541/cannot-import-plot-confusion-matrix