```
import pandas as pd
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
import nltk
import matplotlib.pyplot as plt
from pandas.plotting import radviz
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')

df = pd.read_csv('PoemTrain.csv', header=0)
df2 = pd.read_csv('PoemTest.csv', header=0)
frames = [df, df2]
df = pd.concat(frames)
#)

for i, p in enumerate(df.get("Poem")):  #lemmatize text but it didnt improve accuracy
  if pd.isna(p):
    df.drop(index=i, axis=0, inplace=True)

for i, p in enumerate(df.get("Poem")):  #lemmatize text but it didnt improve accuracy
  poem = word_tokenize(p)
  poem = [t.lower() for t in poem]
  wnl = WordNetLemmatizer()
  lemmas = [wnl.lemmatize(t) for t in poem]
  df.iloc[i,1] = ' '.join(lemmas)
```

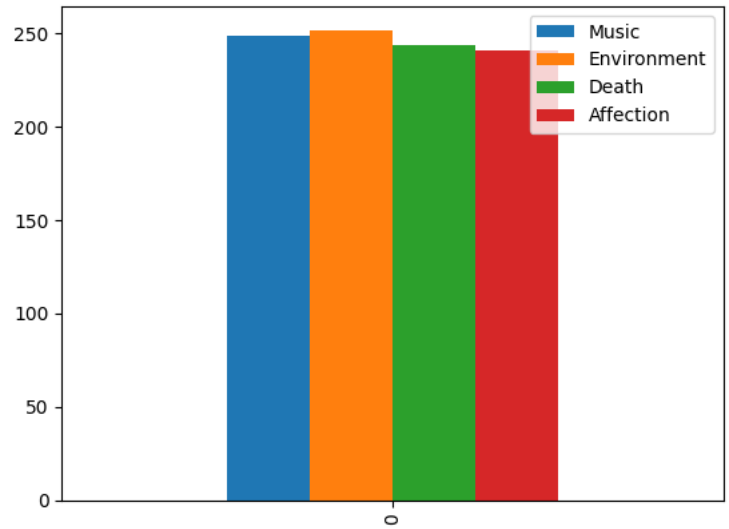
```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
Genre        Poem
Death        translated from french by marilyn hackerfor eliane , mireille , and
             translated from french by marilyn hacker pour ali la pointehere whe
             listen , child : your father is dead . from his old coat i 'll make
Environment  the fern gather where the water seldom go unless the storm swell th
             the holly ! the holly ! oh , twine it with bay—come give the holly

Death        i strayed about the deck , an hour , to-nightunder a cloudy moonles
             i that in heill wa and gladness am trublit now with great sickness
             i wa sympathetic to language , but often it shrugged me and kept ot
             i wa trying to wave to you but you wouldn ' t wave back
Music        “ it ' s all empty , empty , ” he said to himself . “ the sex and d
Length: 983, dtype: int64
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.9/dist-packages/pandas/core/generic.py in
_get_axis_number(cls, axis)
    549         try:
--> 550             return cls._AXIS_TO_AXIS_NUMBER[axis]
    551         except KeyError:

KeyError: 'Music'

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
                                  2 frames
/usr/local/lib/python3.9/dist-packages/pandas/core/generic.py in
_get_axis_number(cls, axis)
    550             return cls._AXIS_TO_AXIS_NUMBER[axis]
    551         except KeyError:
--> 552             raise ValueError(f"No axis named {axis} for object type
{cls.__name__}")
    553
    554     @final

ValueError: No axis named Music for object type DataFrame
```

SEARCH STACK OVERFLOW

```
from tkinter.constants import X
nltk.download('stopwords')
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
stopwords = stopwords.words('english')
vectorizer = TfidfVectorizer( ngram_range=(1,3))

# set up X and y
x = df.Poem   #first row is NaN dont use that
y = df.Genre

print(y.head())

yMusic = [m for m in y if m == 'Music']
print(len(yMusic), 'M')

yEnvironment = [e for e in y if e == 'Environment']
print(len(yEnvironment), 'E')

yDeath = [d for d in y if d == 'Death']
print(len(yDeath), 'D')

yAffection = [d for d in y if d == 'Affection']
print(len(yAffection), 'A')

d = {'Music': [len(yMusic)], 'Environment': [len(yEnvironment)], 'Death': [len(yDeath)], 'Affection': [len(yAffection)]}
dfDisplay = pd.DataFrame(data=d)
plt.figure()
dfDisplay.plot.bar()

x.head()
```

```
1    Music
2    Music
3    Music
4    Music
5    Music
Name: Genre, dtype: object
249 M
252 E
244 D
241 A
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
1    in the thick brushthey spend the hottest part ...
2    storm are generous . something so easy to surr...
3    —after ana mendieta did you carry around the m...
4    for aja sherrard at 20the portent may itself b...
5    for bob marley , bavaria , november 1980 here ...
Name: Poem, dtype: object
<Figure size 640x480 with 0 Axes>
```




```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, train_size=0.8, random_state=1226)

x_train.shape

# apply tfidf vectorizer
x_train = vectorizer.fit_transform(x_train.astype('U'))  # fit and transform the train data
```

```
x_test = vectorizer.transform(x_test.astype('U'))        # transform only the test data

# take a peek at the data
# this is a very sparse matrix because most of the 8613 words don't occur in each sms message

print('train size:', x_train.shape)
print(x_train.toarray()[:5])

print('\ntest size:', x_test.shape)
print(x_test.toarray()[:5])
```

```
train size: (788, 65308)
[[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.]]

test size: (198, 65308)
[[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.]]
```

```
from sklearn.naive_bayes import MultinomialNB

naive_bayes = MultinomialNB()
naive_bayes.fit(x_train, y_train)
```

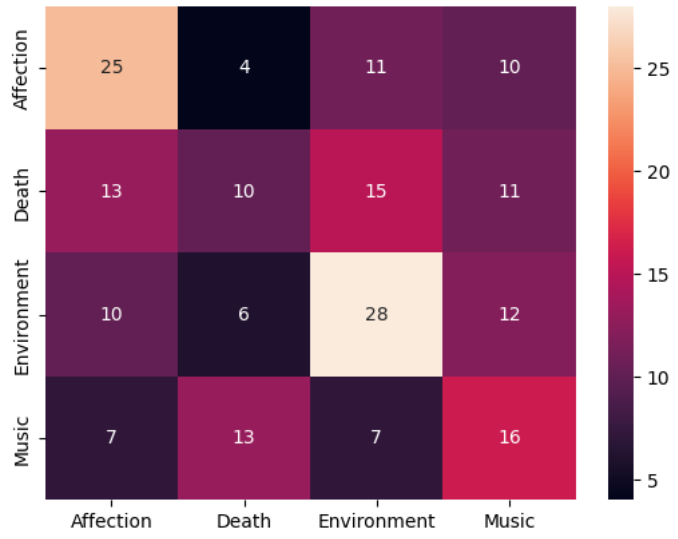
```
▾ MultinomialNB
MultinomialNB()
```

```
# priors
import math
counter = 0
for c in y_train:
  if c == 'Music':
    counter += 1
print(counter)

prior_m = counter/len(y_train)
print('prior Music:', prior_m, 'log of prior:', math.log(prior_m))


counter = 0
for c in y_train:
  if c == 'Environment':
    counter += 1
print(counter)

prior_e = counter/len(y_train)
print('prior Environment:', prior_e, 'log of prior:', math.log(prior_e))


counter = 0
for c in y_train:
  if c == 'Death':
    counter += 1
print(counter)

prior_d = counter/len(y_train)
print('prior Death:', prior_d, 'log of prior:', math.log(prior_d))


counter = 0
for c in y_train:
  if c == 'Affection':
    counter += 1
print(counter)

prior_a = counter/len(y_train)
print('prior Affection:', prior_a, 'log of prior:', math.log(prior_a))


# the model prior matches the prior calculated above
```

```
for classification in naive_bayes.class_log_prior_:
  print(classification)


from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix

# make predictions on the test data
pred = naive_bayes.predict(x_test)

# print confusion matrix
print('\n\n', confusion_matrix(y_test, pred))
```

```
206
prior Music: 0.2614213197969543 log of prior: -1.341621921068298
196
prior Environment: 0.24873096446700507 log of prior: -1.3913834306273618
195
prior Death: 0.24746192893401014 log of prior: -1.3964985312941323
191
prior Affection: 0.24238578680203046 log of prior: -1.4172246618112492
-1.4172246618112494
-1.3964985312941325
-1.3913834306273625
-1.341621921068298


 [[22  7  6 15]
 [ 9 14 11 15]
 [ 8  9 22 17]
 [ 5 11  5 22]]
```

```
#From online to print a heat map of multiple variables
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

#training the classifier using X_Train and y_train
clf = SVC(kernel = 'linear').fit(x_train,y_train)
clf.predict(x_train)

#Testing the model using X_test and storing the output in y_pred
y_pred = clf.predict(x_test)
# Creating  a confusion matrix,which compares the y_test and y_pred
cm = confusion_matrix(y_test, y_pred)

cm_df = pd.DataFrame(cm,index = ['Affection','Death','Environment', 'Music'],
                     columns = ['Affection','Death','Environment', 'Music'])
sns.heatmap(cm_df, annot=True)
plt.show()
#random 1232
```




```
print('accuracy score: ', accuracy_score(y_test, pred))
print(len(y_test))
```

```
print(y_test[y_test != pred])
#with stopword and 0 ngrams. accuracy = 0.33838 with 131 incorrect      RANDOMSATATE = 1222
#with stopwords and 1-3 ngrams. accuracy = 0.343434 with 130 incorrect
#with stopwords and 1-4 ngrams. accuracy = 0.0.353535 with 128 incorrect
#with stopwords and 1-5 ngrams. accuracy = 0.3585858 with 127 incorrect
#without stopwords and 1-4 ngrams accuracy = 0.3686868 with 125 incorrect
#without stopwords and 1-3 ngrams accuracy = 0.373737 with 124 incorrect
#without stopwords and 1-3 ngrams accuracy = 0.41919191919191917 with 115 incorrect RANDOMSATATE = 1224
#without stopwords and 1-3 ngrams accuracy = 0.424242424242 with 114 incorrect RANDOMSATATE = 1226
#without stopwords and 1-3 ngrams accuracy = 0.43434343434343436 with 112 incorrect RANDOMSATATE = 9654
#without stopwords and 1-3 ngrams accuracy = 0.4444444444444444 with 110 incorrect RANDOMSATATE = 1229
```

```
accuracy score:  0.40404040404040403
198
17           Death
334          Death
704    Environment
262          Death
520      Affection
          ...
795    Environment
58       Affection
113          Music
100      Affection
360          Death
Name: Genre, Length: 118, dtype: object
```

```
for i in [198,389,633,70,359,147,396,765,298,279,822]:
  print(df.loc[i])
```

```
Genre                                                 Music
Poem     this is not how it begin but how you understan...
Name: 198, dtype: object
Genre                                                 Death
Poem     one morning the spirit of my lover ' s uncle r...
Name: 389, dtype: object
Genre                                           Environment
Poem     ( thee will i praise between those river whose...
Name: 633, dtype: object
        Genre                                                  Poem
70      Music  for natalieso much like sequin the sunlight on...
70  Affection  elizabeth it is in vain you say '' love not ''...
Genre                                                 Death
Poem     in the steamer is the troutseasoned with slive...
Name: 359, dtype: object
           Genre                                                  Poem
147        Music  say your body ' slife-size trip clock start in...
147  Environment  when pulled , the spider web took another form...
Genre                                                 Death
Poem     shame on you for dating a museum : everything ...
Name: 396, dtype: object
Genre                                     Environment
Poem     stand on the highest pavement of the stair—
Name: 765, dtype: object
Genre                                                 Death
Poem     everybody is doing trigger warning now , so to...
Name: 298, dtype: object
Genre                                                 Death
Poem     ambition—died on august 3 , 2015 , asudden dea...
Name: 279, dtype: object
Genre                                           Environment
Poem     what i meant is that when the child shook the ...
Name: 822, dtype: object
```

```
# binary=True gives binary data instead of counts
vectorizer_b = TfidfVectorizer( binary=True, ngram_range=(1,3))

# set up X and y
x = vectorizer_b.fit_transform(df.Poem)
y = df.Genre

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, train_size=0.8, random_state=1234)

from sklearn.naive_bayes import BernoulliNB

naive_bayes2 = BernoulliNB()
naive_bayes2.fit(x_train, y_train)
```

```
▾ BernoulliNB
BernoulliNB()
```

```
# make predictions on the test data
pred = naive_bayes2.predict(x_test)

# print confusion matrix
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, pred))
print('accuracy score: ', accuracy_score(y_test, pred))
```

```
[[19  2 13  5]
 [12  1 27 17]
 [ 6  2 35  8]
 [11  3 14 23]]
accuracy score:  0.3939393939393939
```

This is a difficult topic to classify because you can make death seem beautiful or use a majority of negative words for affectio nand twist it at the end with "despite it all I still care". The entire second column (death) seems to be greatly misclassified as 27 values are environment when they should be death. Were that fixed the accuracy might be a lot better.But I can see where the predicitons are getting confusing because poems are very interpretation heavy.

The accuracy also is a lot better when I left the stop words in. I knew the bag of words approach is probably pulling down my results but leaving stop words in helps it a lot. I did go back for a secnod time a preprocess the poems into lemmas but that seems to actually have brought the accuracy down too.

Logistic Regression

```
# all the imports for the next code block

import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, log_loss


x = df.Poem   #first row is NaN dont use that
y = df.Genre

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, train_size=0.8, random_state=2486)

x_train.shape

# apply tfidf vectorizer
vectorizer = TfidfVectorizer(binary=True, ngram_range=(1,3))
x_train = vectorizer.fit_transform(x_train.astype('U'))  # fit and transform the train data
x_test = vectorizer.transform(x_test.astype('U'))        # transform only the test data


#train
classifier = LogisticRegression(solver='lbfgs', class_weight='balanced')
classifier.fit(x_train, y_train)


#evaluate
pred = classifier.predict(x_test)
print('accuracy score: ', accuracy_score(y_test, pred))
#print('precision score: ', precision_score(y_test, pred))
#print('recall score: ', recall_score(y_test, pred))
#print('f1 score: ', f1_score(y_test, pred))
probs = classifier.predict_proba(x_test)
print('log loss: ', log_loss(y_test, probs))
```

```
accuracy score:  0.4090909090909091
log loss:  1.3424520037882386
```

```
from sklearn.pipeline import Pipeline

# read in data, split raw data into train and test, then use pipeline to transform
```

```
df = pd.read_csv('PoemTrain.csv', header=0)
df2 = pd.read_csv('PoemTest.csv', header=0)
frames = [df, df2]
df = pd.concat(frames)
#)

for i, p in enumerate(df.get("Poem")):  #lemmatize text but it didnt improve accuracy
  if pd.isna(p):
    df.drop(index=i, axis=0, inplace=True)

x = vectorizer.fit_transform(df.Poem)
y = df.Genre


x_train, x_test, y_train, y_test = train_test_split(df['Poem'], df['Genre'], test_size=0.2, train_size=0.8, random_state=5313)



pipe1 = Pipeline([
        ('tfidf', TfidfVectorizer(binary=True)),
        ('logreg', LogisticRegression(solver='lbfgs', class_weight='balanced')),
])

print(pipe1.fit(x_train, y_train), '\n')

pipe1.fit(x_train, y_train)
pred = pipe1.predict(x_test)
print("accuracy: ", accuracy_score(y_test, pred))
probs = pipe1.predict_proba(x_test)
print("log loss: ", log_loss(y_test, probs))
print('Loss goes down! (Changing parameters only increased loss and lowered accuraccy)')
```

```
Pipeline(steps=[('tfidf', TfidfVectorizer(binary=True)),
                ('logreg', LogisticRegression(class_weight='balanced'))]) 

accuracy:  0.41919191919191917
log loss:  1.3041495050987029
Loss goes down! (Changing parameters only increased loss and lowered accuraccy)
```

```
print('\n\n', confusion_matrix(y_test, pred))
print(len(y_test))
```

```
 [[22  5  7  6]
 [ 9 17 11 19]
 [ 9  8 29  4]
 [10 18  9 15]]
198
```

```
The logistic regression is a lot more spread out. More 6-9 value
inaccurate classifications. The biggest Mess up is the music row
Apparently a lot of death poems and affection seem to be classif
which makes sense to me. I couldn't find exactly how to fix the
found a way to add weight to specific words like "death, dead" a
'
```

The logistic regression is a lot more spread out. More 6-9 values in the inaccurate classifications. The biggest Mess up is the music row and column. Apparently a lot of death poems and affection seem to be classified as music which makes sense to me. I couldn't find exactly how to fix the issues unless i found a way to add weight to specific words like "death, dead" and 'love, care. '

Neural Network

```
from sklearn.pipeline import Pipeline

# read in data, split raw data into train and test, then use pipeline to transform
df = pd.read_csv('PoemTrain.csv', header=0)
df2 = pd.read_csv('PoemTest.csv', header=0)
frames = [df, df2]
df = pd.concat(frames)
#)

for i, p in enumerate(df.get("Poem")):  #lemmatize text but it didnt improve accuracy
```

```
    if pd.isna(p):
      df.drop(index=i, axis=0, inplace=True)


vectorizer = TfidfVectorizer(binary=True)


x = vectorizer.fit_transform(df.Poem)
y = df.Genre

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, train_size=0.8, random_state=3255)

from sklearn.neural_network import MLPClassifier
classifier = MLPClassifier(solver='lbfgs', alpha=1e-6,
                   hidden_layer_sizes=(5, 2), random_state=6)
classifier.fit(x_train, y_train)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:541: ConvergenceWarning: lbfgs faile
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```
MLPClassifier(alpha=1e-06, hidden_layer_sizes=(5, 2), random_state=6,
              solver='lbfgs')
```

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score
pred = classifier.predict(x_test)
print('\n\n', confusion_matrix(y_test, pred))
print(len(y_test))
print('accuracy score: ', accuracy_score(y_test, pred))
#print('precision score: ', precision_score(y_test, pred))
#print('recall score: ', recall_score(y_test, pred))
#print('f1 score: ', f1_score(y_test, pred))
```

```
 [[13  5  6 28]
 [ 6  7  2 29]
 [10  1 17 26]
 [ 7  8  7 26]]
198
accuracy score:  0.3181818181818182
```

Double-click (or enter) to edit

Anything above 5 for the first number of hidden layers seems to group the classifications in the right column or left column. So the accuracy drops significantly with only the top left or bottom right value being correct. Even now with 5 nodes for the first hidden layer, a majority of the classifications are the right column (Which I think is Music).

✓ 0s completed at 11:22 PM