IST736_Project_AHK_WV-FINAL

June 17, 2022

[7]:
```python
# import necessary packages for the project
import os
import re
import sys
import random
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk import FreqDist
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report
from sklearn import svm
from sklearn.metrics import confusion_matrix
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
```

[8]:
```python
# load song data into a data frame
df = pd.read_csv('C:/Users/klein/OneDrive/SU Classes/Quarter 4/IST 736/Project/
 ↪IST 736 Lyrics Dataset v2.csv', engine = 'python', encoding='unicode_escape')
df
```

[8]:
```
                                                Lyrics  \
0    I heard life is what passes when you're too bu…
1    If two fill-ups is all it costs, I guess I'll …
2    He was a boy who was a dreamer\nAnd he flew so…
3    I'm condemned, I'm condemned\nOh, my heart is …
4    In a life where we work out there's a house up…
..                                                 …
235  Oh, yeah\nOh, yeah\nEverything gonna be alrigh…
236  Lie to me\nAnd tell me everything is alright\n…
```

```
237  You sing a song, While sitting at a red light\…
238  Come on, oh baby, don't you wanna go?\nCome on…
239  Uh, aw, sookie sookie now!\nHey, ow, uh, come …

                         Song          Artist    Genre
0                   Late July      Zach Bryan  Country
1               Crooked Teeth      Zach Bryan  Country
2               Heading South      Zach Bryan  Country
3                   Condemned      Zach Bryan  Country
4    A Life Where We Work Out  Flatland Calvary  Country
..                        …             …       …
235                Mannish Boy     Muddy Water    Blues
236                 Lie to Me      Jonny Lang    Blues
237                 Red Light      Jonny Lang    Blues
238         Sweet Home Chicago    Eric Clapton    Blues
239                 Groove Me      King Floyd    Blues

[240 rows x 4 columns]
```

[9]:
```python
# create function to \n replace dataframe
def NReplace(Text):
    Text = Text.replace('\n', ' ')
    return Text
```

[10]:
```python
# apply \n replace function to Lyrics field in dataframe
df['Lyrics'] = df['Lyrics'].apply(NReplace)
df
```

[10]:
```
                                                Lyrics  \
0    I heard life is what passes when you're too bu…
1    If two fill-ups is all it costs, I guess I'll …
2    He was a boy who was a dreamer And he flew so …
3    I'm condemned, I'm condemned Oh, my heart is o…
4    In a life where we work out there's a house up…
..                                                 …
235  Oh, yeah Oh, yeah Everything gonna be alright …
236  Lie to me And tell me everything is alright Li…
237  You sing a song, While sitting at a red light …
238  Come on, oh baby, don't you wanna go? Come on,…
239  Uh, aw, sookie sookie now! Hey, ow, uh, come o…

                         Song          Artist    Genre
0                   Late July      Zach Bryan  Country
1               Crooked Teeth      Zach Bryan  Country
2               Heading South      Zach Bryan  Country
3                   Condemned      Zach Bryan  Country
4    A Life Where We Work Out  Flatland Calvary  Country
```

```
..                      …             …      …
235             Mannish Boy      Muddy Water    Blues
236               Lie to Me       Jonny Lang    Blues
237               Red Light       Jonny Lang    Blues
238      Sweet Home Chicago     Eric Clapton    Blues
239               Groove Me       King Floyd    Blues

[240 rows x 4 columns]
```

[11]:
```
# create function to clean dataframe
punctuation = r"[.?%!,;:-'^/#@$*()]"
def Cleaner(Text):
    Text = Text.lower()
    Text = Text.replace('[^A-Za-z0-9]+', '')
    Text = Text.translate(str.maketrans("", "", punctuation))
    return Text
```

[12]:
```
# apply cleaner function to Lyrics field in dataframe
df['Lyrics'] = df['Lyrics'].apply(Cleaner)
df
```

[12]:
```
                                                Lyrics  \
0    i heard life is what passes when youre too bus…
1    if two fillups is all it costs i guess ill mak…
2    he was a boy who was a dreamer and he flew so …
3    im condemned im condemned oh my heart is on th…
4    in a life where we work out theres a house upo…
..                                                 …
235  oh yeah oh yeah everything gonna be alright th…
236  lie to me and tell me everything is alright li…
237  you sing a song while sitting at a red light y…
238  come on oh baby dont you wanna go come on oh b…
239  uh aw sookie sookie now hey ow uh come on baby…

                          Song           Artist    Genre
0                   Late July       Zach Bryan  Country
1               Crooked Teeth       Zach Bryan  Country
2               Heading South       Zach Bryan  Country
3                   Condemned       Zach Bryan  Country
4    A Life Where We Work Out  Flatland Calvary  Country
..                        …             …      …
235             Mannish Boy      Muddy Water    Blues
236               Lie to Me       Jonny Lang    Blues
237               Red Light       Jonny Lang    Blues
238      Sweet Home Chicago     Eric Clapton    Blues
239               Groove Me       King Floyd    Blues
```

```
[240 rows x 4 columns]
```

[13]:
```
# Define stemmer function and apply to df
porter = nltk.PorterStemmer()
def Stemmer(Text):
    Words = Text.split()
    StemmedWords = [porter.stem(Word) for Word in Words]
    return ' '.join(StemmedWords)

df['Lyrics'] = df['Lyrics'].apply(Stemmer)
df
```

[13]:
```
                                                Lyrics  \
0    i heard life is what pass when your too busi l…
1    if two fillup is all it cost i guess ill make …
2    he wa a boy who wa a dreamer and he flew so hi…
3    im condemn im condemn oh my heart is on the me…
4    in a life where we work out there a hous upon …
..                                                 …
235  oh yeah oh yeah everyth gonna be alright thi m…
236  lie to me and tell me everyth is alright lie t…
237  you sing a song while sit at a red light you t…
238  come on oh babi dont you wanna go come on oh b…
239  uh aw sooki sooki now hey ow uh come on babi h…

                         Song           Artist    Genre
0                   Late July       Zach Bryan  Country
1               Crooked Teeth       Zach Bryan  Country
2               Heading South       Zach Bryan  Country
3                   Condemned       Zach Bryan  Country
4    A Life Where We Work Out  Flatland Calvary  Country
..                        …               …       …
235               Mannish Boy      Muddy Water    Blues
236                Lie to Me        Jonny Lang    Blues
237                Red Light        Jonny Lang    Blues
238        Sweet Home Chicago     Eric Clapton    Blues
239                Groove Me        King Floyd    Blues

[240 rows x 4 columns]
```

[14]:
```
# Vectorize the dataframe
CountVec = CountVectorizer(encoding='latin-1')
CountVecB = CountVectorizer(encoding='latin-1', ngram_range=(1,2))
df.dropna(how='any',inplace=True)
```

[15]:
```
# Create test and train datasets from df for MNB
```

```
X_Train, X_Test, Y_Train, Y_Test = train_test_split(df['Lyrics'], df['Genre'],␣
↪stratify=df['Genre'], test_size=0.3, random_state=1)
```

[16]:
```
# create CountVectorizer Training and Testing data sub-sets
CTrainX = CountVec.fit_transform(X_Train)
CTestX = CountVec.transform(X_Test)
CTrainXB = CountVecB.fit_transform(X_Train)
CTestXB = CountVecB.transform(X_Test)
```

[17]:
```
# Train Multinomial Naive Bayes Model on unigrams and run classification report
NbModel = MultinomialNB().fit(CTrainX, Y_Train)
print(classification_report(Y_Test, NbModel.predict(CTestX)))
```

```
              precision    recall  f1-score   support

       Blues       0.67      0.17      0.27        12
     Country       0.60      0.50      0.55        12
         Pop       0.35      0.67      0.46        12
         R&B       0.27      0.25      0.26        12
         Rap       0.92      0.92      0.92        12
        Rock       0.46      0.50      0.48        12

    accuracy                           0.50        72
   macro avg       0.54      0.50      0.49        72
weighted avg       0.54      0.50      0.49        72
```

[18]:
```
# run k-fold cross validation on MNB with unigrams
Output_Scores = cross_val_score(NbModel, CTestX, Y_Test, cv=10)
Accuracy = Output_Scores.mean()
print('Multinomial Naive Bayes Unigrams Percentage Accuracy = %0.2f' %␣
↪(Accuracy *100))
```

```
Multinomial Naive Bayes Unigrams Percentage Accuracy = 41.96
```

[19]:
```
# Train Multinomial Naive Bayes Model with Unigrams and Bigrams and run␣
↪classification report
NbModelB = MultinomialNB().fit(CTrainXB, Y_Train)
print(classification_report(Y_Test, NbModelB.predict(CTestXB)))
```

```
              precision    recall  f1-score   support

       Blues       1.00      0.17      0.29        12
     Country       0.78      0.58      0.67        12
         Pop       0.28      0.67      0.39        12
         R&B       0.54      0.58      0.56        12
         Rap       0.73      0.92      0.81        12
        Rock       0.50      0.17      0.25        12
```

```
    accuracy                           0.51        72
   macro avg       0.64      0.51      0.49        72
weighted avg       0.64      0.51      0.49        72
```

[20]:
```
# run k-fold cross validation on MNB with unigrams and bigrams
Output_ScoresB = cross_val_score(NbModelB, CTestXB, Y_Test, cv=10)
AccuracyB = Output_ScoresB.mean()
print('Multinomial Naive Bayes Unigrams and Bigrams Percentage Accuracy = %0.
 ↪2f' % (AccuracyB *100))
```

```
Multinomial Naive Bayes Unigrams and Bigrams Percentage Accuracy = 32.32
```

[21]:
```
# create tfidf vectorizer and use it to create training and testing data␣
 ↪sub-sets
tfidf_Vec = TfidfVectorizer()
tfidf_VecB = TfidfVectorizer(ngram_range=(1,2))
tfidf_TrainX = tfidf_Vec.fit_transform(X_Train)
tfidf_TrainXB = tfidf_VecB.fit_transform(X_Train)
tfidf_TestX = tfidf_Vec.transform(X_Test)
tfidf_TestXB = tfidf_VecB.transform(X_Test)
print(tfidf_TestX.shape)
```

```
(72, 3556)
```

[22]:
```
# set the svm algorithm
svm = LinearSVC(C=1)
```

[23]:
```
# fit the svm with our data for Unigrams
svm.fit(tfidf_TrainX, Y_Train)
```

[23]:
```
LinearSVC(C=1)
```

[58]:
```
# create classification report for svm model with unigrams
SVM_Y_Prediction = svm.predict(tfidf_TestX)
print(classification_report(Y_Test, SVM_Y_Prediction))
```

```
              precision    recall  f1-score   support

       Blues       0.50      0.50      0.50        12
     Country       0.89      0.67      0.76        12
         Pop       0.33      0.25      0.29        12
         R&B       0.30      0.25      0.27        12
         Rap       0.85      0.92      0.88        12
        Rock       0.37      0.58      0.45        12

    accuracy                           0.53        72
   macro avg       0.54      0.53      0.53        72
weighted avg       0.54      0.53      0.53        72
```

[59]:
```
# run k-fold cross validation on SVM with unigrams
Output_Scores2 = cross_val_score(svm, tfidf_TestX, Y_Test, cv=10)
Accuracy2 = Output_Scores2.mean()
print('SVM Unigrams Percentage Accuracy = %0.2f' % (Accuracy2 *100))
```

```
SVM Unigrams Percentage Accuracy = 46.96
```

[26]:
```
# set the svm algorithm
svmB = LinearSVC(C=1)
```

[27]:
```
# fit the svm with unigrams and bigrams data
svmB.fit(tfidf_TrainXB, Y_Train)
```

[27]:
```
LinearSVC(C=1)
```

[28]:
```
# create classification report for svm model with unigrams and bigrams
SVM_Y_PredictionB = svmB.predict(tfidf_TestXB)
print(classification_report(Y_Test, SVM_Y_PredictionB))
```

```
              precision    recall  f1-score   support

       Blues       0.57      0.33      0.42        12
     Country       0.57      0.67      0.62        12
         Pop       0.40      0.33      0.36        12
         R&B       0.20      0.17      0.18        12
         Rap       0.75      1.00      0.86        12
        Rock       0.33      0.42      0.37        12

    accuracy                           0.49        72
   macro avg       0.47      0.49      0.47        72
weighted avg       0.47      0.49      0.47        72
```

[29]:
```
# run k-fold cross validation on SVM with unigrams and bigrams
Output_Scores2B = cross_val_score(svmB, tfidf_TestXB, Y_Test, cv=10)
Accuracy2B = Output_Scores2B.mean()
print('SVM Unigrams and Bigrams Percentage Accuracy = %0.2f' % (Accuracy2B␣
 ↪*100))
```

```
SVM Unigrams and Bigrams Percentage Accuracy = 44.11
```

[60]:
```
# train RandomForest model with unigrams and run classification report
RF = RandomForestClassifier()
RF.fit(tfidf_TrainX, Y_Train)
RF_Y_Prediction = RF.predict(tfidf_TestX)
print(classification_report(Y_Test, RF_Y_Prediction))
```

```
              precision    recall  f1-score   support

       Blues       0.50      0.75      0.60        12
     Country       0.64      0.58      0.61        12
         Pop       0.50      0.50      0.50        12
         R&B       0.50      0.42      0.45        12
         Rap       0.92      1.00      0.96        12
        Rock       0.62      0.42      0.50        12

    accuracy                           0.61        72
   macro avg       0.61      0.61      0.60        72
weighted avg       0.61      0.61      0.60        72
```

[63]:
```
# run k-fold cross validation on RandomForest with unigrams
Output_Scores4 = cross_val_score(RF, tfidf_TestX, Y_Test, cv=10)
Accuracy4 = Output_Scores4.mean()
print('RandomForest Unigrams Percentage Accuracy = %0.2f' % (Accuracy4 *100))
```

```
RandomForest Unigrams Percentage Accuracy = 52.68
```

[56]:
```
# train RandomForest model using unigrams and bigrams data and run␣
 ↪classification report
RFB = RandomForestClassifier()
RFB.fit(tfidf_TrainXB, Y_Train)
RF_Y_PredictionB = RFB.predict(tfidf_TestXB)
print(classification_report(Y_Test, RF_Y_PredictionB))
```

```
              precision    recall  f1-score   support

       Blues       0.62      0.83      0.71        12
     Country       0.50      0.42      0.45        12
         Pop       0.62      0.42      0.50        12
         R&B       0.33      0.33      0.33        12
         Rap       1.00      1.00      1.00        12
        Rock       0.21      0.25      0.23        12

    accuracy                           0.54        72
   macro avg       0.55      0.54      0.54        72
weighted avg       0.55      0.54      0.54        72
```

[57]:
```
# run k-fold cross validation on RandomForest with unigrams and bigrams
Output_Scores4B = cross_val_score(RFB, tfidf_TestXB, Y_Test, cv=10)
Accuracy4B = Output_Scores4B.mean()
print('RandomForest Unigrams and Bigrams Percentage Accuracy = %0.2f' %␣
 ↪(Accuracy4B *100))
```

```
RandomForest Unigrams and Bigrams Percentage Accuracy = 45.71
```

```
[46]: # train neural network model MLP with unigrams and run the classification report
MLP = MLPClassifier(solver='lbfgs', random_state=8, hidden_layer_sizes =␣
 ↪(5,4,3), max_iter=10000)
MLP.fit(CTrainX, Y_Train)
NnMLP_Y_Prediction = MLP.predict(CTestX)
print(classification_report(Y_Test, NnMLP_Y_Prediction))
```

```
              precision    recall  f1-score   support

       Blues       0.40      0.33      0.36        12
     Country       0.44      0.67      0.53        12
         Pop       0.33      0.25      0.29        12
         R&B       0.25      0.17      0.20        12
         Rap       0.44      0.58      0.50        12
        Rock       0.18      0.17      0.17        12

    accuracy                           0.36        72
   macro avg       0.34      0.36      0.34        72
weighted avg       0.34      0.36      0.34        72
```

```
[47]: # run k-fold cross validation on NN MLP with unigrams
Output_Scores5 = cross_val_score(MLP, tfidf_TestX, Y_Test, cv=10)
Accuracy5 = Output_Scores5.mean()
print('Neural Network MLP Unigrams Percentage Accuracy = %0.2f' % (Accuracy5␣
 ↪*100))
```

```
Neural Network MLP Unigrams Percentage Accuracy = 27.86
```

```
[49]: # train neural network model MLP with unigrams and bigrams and run␣
 ↪classification report
MLPB = MLPClassifier(solver='lbfgs', random_state=2, hidden_layer_sizes =␣
 ↪(5,4,3), max_iter=100000)
MLPB.fit(CTrainXB, Y_Train)
NnMLP_Y_PredictionB = MLPB.predict(CTestXB)
print(classification_report(Y_Test, NnMLP_Y_PredictionB))
```

```
              precision    recall  f1-score   support

       Blues       0.44      0.33      0.38        12
     Country       0.40      0.33      0.36        12
         Pop       0.57      0.33      0.42        12
         R&B       0.21      0.25      0.23        12
         Rap       0.50      0.67      0.57        12
        Rock       0.38      0.50      0.43        12

    accuracy                           0.40        72
   macro avg       0.42      0.40      0.40        72
weighted avg       0.42      0.40      0.40        72
```

```
[50]:  # run k-fold cross validation on NN MLP with unigrams and bigrams
       Output_Scores5B = cross_val_score(MLPB, tfidf_TestXB, Y_Test, cv=10)
       Accuracy5B = Output_Scores5B.mean()
       print('Neural Network MLP Unigrams and Bigrams Percentage Accuracy = %0.2f' %␣
        ↪(Accuracy5B *100))
```

```
Neural Network MLP Unigrams and Bigrams Percentage Accuracy = 19.46
```