Songs Lyrics Classification into gender baseline models

In [1]:
```
import numpy as np
import pandas as pd
import re
import nltk
from nltk.corpus import stopwords
import matplotlib.pyplot as plt
import string
import multiprocessing
import os
import sklearn
import pprint
import seaborn as sns
nltk.download('stopwords')
%matplotlib inline
stop = stopwords.words('english')
from subprocess import check_output
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\LENOVO\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [2]:
```
df1 = pd.read_csv('lyrics_6.csv')
df1.head()

df1
```

Out[2]:

	Artist	Gender	Lyrics
0	The Weeknd	male	Yeah\r\r\nYeah\r\r\nYeah\r\r\nYour man on the ...
1	The Weeknd	male	I'm tryna put you in the worst mood, ah\r\r\nP...
2	The Weeknd	male	And I know she'll be the death of me\r\r\nAt l...
3	The Weeknd	male	We found each other\r\r\nI helped you out of a...
4	The Weeknd	male	NaN
...	...	...	...
1045	Taylor Swift	Female	Once upon a time, a few mistakes ago\r\r\nI wa...
1046	Taylor Swift	Female	We're all bored, we're all so tired of everyth...
1047	Taylor Swift	Female	We could leave the Christmas lights up 'til Ja...
1048	Taylor Swift	Female	It feels like a perfect night\r\r\nTo dress up...
1049	Taylor Swift	Female	Flashing lights, and we\r\r\nTook a wrong turn...

1050 rows × 3 columns

In [3]:
```
df1=df1.drop(columns=['Artist'])
df1.head()
```

Out[3]:

	Gender	Lyrics
0	male	Yeah\r\r\nYeah\r\r\nYeah\r\r\nYour man on the ...
1	male	I'm tryna put you in the worst mood, ah\r\r\nP...
2	male	And I know she'll be the death of me\r\r\nAt l...
3	male	We found each other\r\r\nI helped you out of a...
4	male	NaN

In [4]:
```
df1=df1.dropna(subset=['Lyrics'])
df1
```

Out[4]:

	Gender	Lyrics
0	male	Yeah\r\r\nYeah\r\r\nYeah\r\r\nYour man on the ...
1	male	I'm tryna put you in the worst mood, ah\r\r\nP...
2	male	And I know she'll be the death of me\r\r\nAt l...
3	male	We found each other\r\r\nI helped you out of a...
5	male	Seneler sürer her günüm\r\r\nOoh yeah, ooh yea...
...	...	...
1045	Female	Once upon a time, a few mistakes ago\r\r\nI wa...
1046	Female	We're all bored, we're all so tired of everyth...
1047	Female	We could leave the Christmas lights up 'til Ja...
1048	Female	It feels like a perfect night\r\r\nTo dress up...
1049	Female	Flashing lights, and we\r\r\nTook a wrong turn...

1020 rows × 2 columns

In [5]:
```
#Removing \n carriage returns
df1=df1.replace({'\n':' '},regex=True)
df1.head()
```

Out[5]:

	Gender	Lyrics
0	male	Yeah\r\r Yeah\r\r Yeah\r\r Your man on the roa...
1	male	I'm tryna put you in the worst mood, ah\r\r P1...
2	male	And I know she'll be the death of me\r\r At le...
3	male	We found each other\r\r I helped you out of a ...
5	male	Seneler sürer her günüm\r\r Ooh yeah, ooh yeah...

Top words in lyrics

In [6]:
```
import nltk
from nltk.tokenize import word_tokenize
lyricss= df1.Lyrics.str.cat(sep=' ')
#function to split text into word
tokens = word_tokenize(lyricss)
vocabulary = set(tokens)
print(len(vocabulary))
frequency_dist = nltk.FreqDist(tokens)
sorted(frequency_dist,key=frequency_dist.__getitem__, reverse=True)[0:50]
```

```
15465
```

Out[6]:
```
[',',
 'I',
 'you',
 'the',
 'a',
 'me',
 'it',
 "n't",
 "'",
 'my',
 'to',
 'that',
 "'m",
 'in',
 "'s",
 'on',
 'and',
 'do',
 'like',
 'up',
 'got',
 'know',
 'your',
 'And',
 '?',
 'with',
 'of',
 'You',
 'be',
 'for',
 'all',
 'we',
 'just',
 'yeah',
 'get',
 'na',
 'out',
 'ai',
 'love',
 'no',
 'was',
 'what',
 'they',
 'this',
 '’',
```

```
 'so',
 'is',
 "'re",
 'she',
 'want']
```

In [7]:
```
df1.Lyrics[2]
print(df1.shape)
```

```
(1020, 2)
```

Remove special characters

In [9]:
```
def standardize_text(df, text_field):
    df[text_field] = df[text_field].str.lower()
    df[text_field] = df[text_field].str.replace(r"\[.*?\]", "")
    df[text_field] = df[text_field].str.replace(r"\r", "")
    df[text_field] = df[text_field].str.replace(r"\w*\d\w*", " ")
    return df

df1["Lyrics"]=df1["Lyrics"].apply(str)
df1 = standardize_text(df1, "Lyrics")

#x_train.to_csv("clean_data.csv")
df1
```

Out[9]:

	Gender	Lyrics
0	male	yeah yeah yeah your man on the road, he doin' ...
1	male	i'm tryna put you in the worst mood, ah clea...
2	male	and i know she'll be the death of me at least ...
3	male	we found each other i helped you out of a brok...
5	male	seneler sürer her günüm ooh yeah, ooh yeah, oo...
...	...	...
1045	Female	once upon a time, a few mistakes ago i was in ...
1046	Female	we're all bored, we're all so tired of everyth...
1047	Female	we could leave the christmas lights up 'til ja...
1048	Female	it feels like a perfect night to dress up like...
1049	Female	flashing lights, and we took a wrong turn, and...

1020 rows × 2 columns

Removing Stop Words

In [10]:
```
from nltk.corpus import stopwords

english_stop_words = stopwords.words('english')
def remove_stop_words(corpus):
    removed_stop_words = []
    for review in corpus:
        removed_stop_words.append(
            ' '.join([word for word in review.split()
                      if word not in english_stop_words])
        )
    return removed_stop_words

text=df1['Lyrics']
df1['Lyrics'] =remove_stop_words(text)
```

In [13]:
```
Y=df1['Gender']
Y.head(5)
```

Out[13]:
```
0    male
1    male
2    male
3    male
5    male
Name: Gender, dtype: object
```

Tokenization

In [14]:
```
def get_lemmatized_text(corpus):
    from nltk.stem import WordNetLemmatizer
    lemmatizer = WordNetLemmatizer()
    return [' '.join([lemmatizer.lemmatize(word) for word in review.split()])
text=df1["Lyrics"]
df1['Lyrics'] = get_lemmatized_text(text)
```

In [15]:
```
df1
```

Out[15]:

	Gender	Lyrics
0	male	yeah yeah yeah man road, doin' promo said, "ke...
1	male	i'm tryna put worst mood, ah cleaner church sh...
2	male	know she'll death least we'll numb she'll alwa...
3	male	found helped broken place gave comfort falling...
5	male	seneler sürer günüm ooh yeah, ooh yeah, ooh ye...
...	...	...
1045	Female	upon time, mistake ago sights, got alone found...
1046	Female	we're bored, we're tired everything wait train...
1047	Female	could leave christmas light 'til january place...
1048	Female	feel like perfect night dress like hipster mak...
1049	Female	flashing lights, took wrong turn, fell rabbit ...

1020 rows × 2 columns

In [16]:
```
def get_stemmed_text(corpus):
    from nltk.stem.porter import PorterStemmer
    stemmer = PorterStemmer()
    return [' '.join([stemmer.stem(word) for word in review.split()]) for rev

text=df1["Lyrics"]
df1["Lyrics"] = get_stemmed_text(text)
df1.head()
```

Out[16]:

	Gender	Lyrics
0	male	yeah yeah yeah man road, doin' promo said, "ke...
1	male	i'm tryna put worst mood, ah cleaner church sh...
2	male	know she'll death least we'll numb she'll alwa...
3	male	found help broken place gave comfort fall mist...
5	male	senel sürer günüm ooh yeah, ooh yeah, ooh yeah...

In [17]:
```
from nltk.tokenize import RegexpTokenizer

tokenizer = RegexpTokenizer(r'\w+')

df1['Lyrics'] =df1['Lyrics'].apply(tokenizer.tokenize)
df1
```

Out[17]:

	Gender	Lyrics
0	male	[yeah, yeah, yeah, man, road, doin, promo, sai...
1	male	[i, m, tryna, put, worst, mood, ah, cleaner, c...
2	male	[know, she, ll, death, least, we, ll, numb, sh...
3	male	[found, help, broken, place, gave, comfort, fa...
5	male	[senel, sürer, günüm, ooh, yeah, ooh, yeah, oo...
...	...	...
1045	Female	[upon, time, mistak, ago, sights, got, alon, f...
1046	Female	[we, r, bored, we, r, tire, everyth, wait, tra...
1047	Female	[could, leav, christma, light, til, januari, p...
1048	Female	[feel, like, perfect, night, dress, like, hips...
1049	Female	[flash, lights, took, wrong, turn, fell, rabbi...

1020 rows × 2 columns

In [18]:
```
df1
```

Out[18]:

	Gender	Lyrics
0	male	[yeah, yeah, yeah, man, road, doin, promo, sai...
1	male	[i, m, tryna, put, worst, mood, ah, cleaner, c...
2	male	[know, she, ll, death, least, we, ll, numb, sh...
3	male	[found, help, broken, place, gave, comfort, fa...
5	male	[senel, sürer, günüm, ooh, yeah, ooh, yeah, oo...
...	...	...
1045	Female	[upon, time, mistak, ago, sights, got, alon, f...
1046	Female	[we, r, bored, we, r, tire, everyth, wait, tra...
1047	Female	[could, leav, christma, light, til, januari, p...
1048	Female	[feel, like, perfect, night, dress, like, hips...
1049	Female	[flash, lights, took, wrong, turn, fell, rabbi...

1020 rows × 2 columns

Stemming

Feature Extraction using Vectorize

In [19]:
```
from sklearn.preprocessing import LabelBinarizer
encoder = LabelBinarizer()
Y = encoder.fit_transform(df1['Gender'])
```

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

def cv(data):
    count_vectorizer = CountVectorizer()

    emb = count_vectorizer.fit_transform(data)

    return emb, count_vectorizer

list_corpus = df1["Lyrics"].tolist()
#list_labels = df1["Artist"].tolist()

X_train, X_test, y_train, y_test = train_test_split(list_corpus, Y, test_size


X_train_counts, count_vectorizer = cv(X_train)
X_test_counts = count_vectorizer.transform(X_test)
```

Feature extraction using tf-idf

In [28]:
```
def tfidf(data):
    tfidf_vectorizer = TfidfVectorizer()

    train = tfidf_vectorizer.fit_transform(data)

    return train, tfidf_vectorizer

X_train_tfidf, tfidf_vectorizer = tfidf(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

Baseline model

Logistic Regression with count vectorize

In [29]:
```
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression()
clf.fit(X_train_counts, y_train)

y_predicted_counts = clf.predict(X_test_counts)
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall

def get_metrics(y_test, y_predicted_counts):
    # true positives / (true positives+false positives)
    precision = precision_score(y_test, y_predicted_counts, pos_label=None,
                                average='weighted')
    # true positives / (true positives + false negatives)
    recall = recall_score(y_test, y_predicted_counts, pos_label=None,
                          average='weighted')

    # harmonic mean of precision and recall
    f1 = f1_score(y_test, y_predicted_counts, pos_label=None, average='weight

    # true positives + true negatives/ total
    accuracy = accuracy_score(y_test, y_predicted_counts)
    return accuracy, precision, recall, f1

accuracy, precision, recall, f1 = get_metrics(y_test, y_predicted_counts)
print("accuracy = %.3f, precision = %.3f, recall = %.3f, f1 = %.3f" % (accura
```

```
accuracy = 0.804, precision = 0.802, recall = 0.804, f1 = 0.801

C:\Users\LENOVO\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.
py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
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 (https://sci
kit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession (https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression)
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Logistic regression with tfidf

In [30]:
```
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression()
clf.fit(X_train_tfidf, y_train)

y_predicted_tfidf = clf.predict(X_test_tfidf)
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall

def get_metrics(y_test, y_predicted_tfidf):
    # true positives / (true positives+false positives)
    precision = precision_score(y_test, y_predicted_tfidf, pos_label=None,
                                average='weighted')
    # true positives / (true positives + false negatives)
    recall = recall_score(y_test, y_predicted_tfidf, pos_label=None,
                          average='weighted')

    # harmonic mean of precision and recall
    f1 = f1_score(y_test, y_predicted_tfidf, pos_label=None, average='weighte

    # true positives + true negatives/ total
    accuracy = accuracy_score(y_test, y_predicted_tfidf)
    return accuracy, precision, recall, f1

accuracy, precision, recall, f1 = get_metrics(y_test, y_predicted_tfidf)
print("accuracy = %.3f, precision = %.3f, recall = %.3f, f1 = %.3f" % (accura
```

```
accuracy = 0.740, precision = 0.760, recall = 0.740, f1 = 0.715
```

Naive Bayes using Count Vectorize

In [31]:
```
from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB().fit(X_train_counts, y_train)
y_predicted_counts=clf.predict(X_test_counts)

from sklearn.metrics import accuracy_score, f1_score, precision_score, recall

def get_metrics(y_test, y_predicted_counts):
    # true positives / (true positives+false positives)
    precision = precision_score(y_test, y_predicted_counts, pos_label=None,
                                average='weighted')
    # true positives / (true positives + false negatives)
    recall = recall_score(y_test, y_predicted_counts, pos_label=None,
                          average='weighted')

    # harmonic mean of precision and recall
    f1 = f1_score(y_test, y_predicted_counts, pos_label=None, average='weight

    # true positives + true negatives/ total
    accuracy = accuracy_score(y_test, y_predicted_counts)
    return accuracy, precision, recall, f1

accuracy, precision, recall, f1 = get_metrics(y_test, y_predicted_counts)
print("accuracy = %.3f, precision = %.3f, recall = %.3f, f1 = %.3f" % (accura
```

```
accuracy = 0.789, precision = 0.787, recall = 0.789, f1 = 0.786
```

In [32]:
```
from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB().fit(X_train_tfidf, y_train)
y_predicted_counts=clf.predict(X_test_tfidf)

from sklearn.metrics import accuracy_score, f1_score, precision_score, recall

def get_metrics(y_test, y_predicted_counts):
    # true positives / (true positives+false positives)
    precision = precision_score(y_test, y_predicted_counts, pos_label=None,
                                average='weighted')
    # true positives / (true positives + false negatives)
    recall = recall_score(y_test, y_predicted_counts, pos_label=None,
                          average='weighted')

    # harmonic mean of precision and recall
    f1 = f1_score(y_test, y_predicted_counts, pos_label=None, average='weight

    # true positives + true negatives/ total
    accuracy = accuracy_score(y_test, y_predicted_counts)
    return accuracy, precision, recall, f1

accuracy, precision, recall, f1 = get_metrics(y_test, y_predicted_counts)
print("accuracy = %.3f, precision = %.3f, recall = %.3f, f1 = %.3f" % (accura
```

```
accuracy = 0.642, precision = 0.698, recall = 0.642, f1 = 0.542
```