In [305…
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error, accuracy_score
```

First, we'll import the spotify dataset to clean the data and conduct EDA.

In [306…
```
data = pd.read_csv('taylor_swift_spotify.csv')
data.head()
```

Out[306…

	Unnamed: 0	name	album	release_date	track_number	id	
0	0	Welcome To New York (Taylor's Version)	1989 (Taylor's Version) [Deluxe]	2023-10-27	1	4WUepByoeqcedHoYhSNHRt	spotify:track:4WUepByoeqcedHo
1	1	Blank Space (Taylor's Version)	1989 (Taylor's Version) [Deluxe]	2023-10-27	2	0108kcWLnn2HlH2kedi1gn	spotify:track:0108kcWLnn2HlH
2	2	Style (Taylor's Version)	1989 (Taylor's Version) [Deluxe]	2023-10-27	3	3Vpk1hfMAQme8VJ0SNRSkd	spotify:track:3Vpk1hfMAQme8VJ
3	3	Out Of The Woods (Taylor's Version)	1989 (Taylor's Version) [Deluxe]	2023-10-27	4	1OcSfkeCg9hRC2sFKB4IMJ	spotify:track:1OcSfkeCg9hRC2s
4	4	All You Had To Do Was Stay (Taylor's Version)	1989 (Taylor's Version) [Deluxe]	2023-10-27	5	2k0ZEeAqzvYMcx9Qt5aClQ	spotify:track:2k0ZEeAqzvYMcx9

In [307…

```
# Data cleaning
data["release_date"] = pd.to_datetime(data["release_date"])
data["year"] = data["release_date"].dt.year

# There are many versions of each album. For this part of our analysis, we'll use the original versions of
standard_albums = ['Taylor Swift', 'Fearless', 'Speak Now', 'Red', '1989', 'reputation', 'Lover', 'folklor
standard_albums_data = data.loc[data['album'].isin(standard_albums)]

standard_albums_data.head()
```

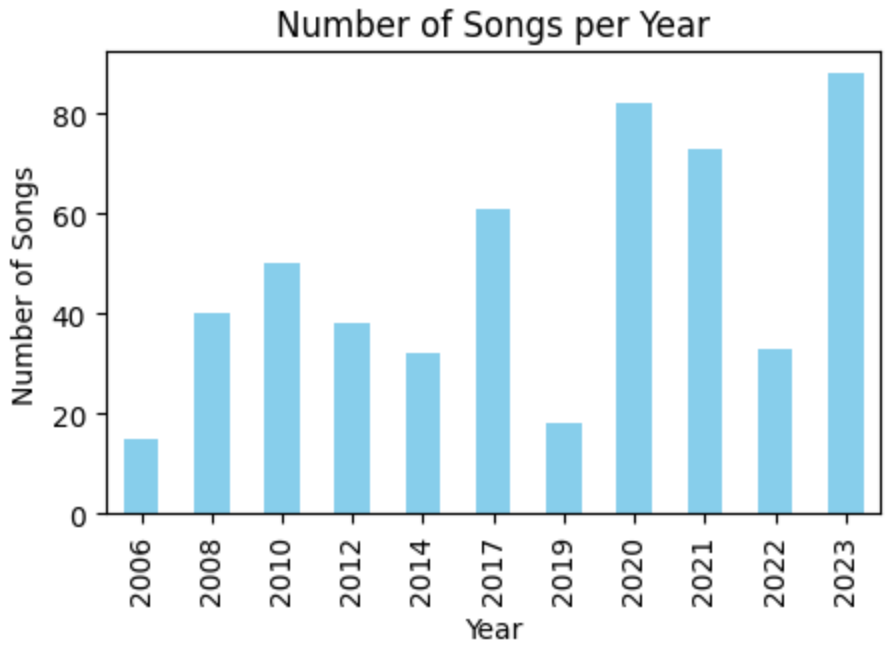
Out [307…

	Unnamed: 0	name	album	release_date	track_number	id	
108	108	Lavender Haze	Midnights	2022-10-21	1	5jQI2r1RdgtuT8S3iG8zFC	spotify:track:5jQI2r1Rdgtu
109	109	Maroon	Midnights	2022-10-21	2	3eX0NZfLtGzoLUxPNvRfqm	spotify:track:3eX0NZfLtGzoL
110	110	Anti-Hero	Midnights	2022-10-21	3	0V3wPSX9ygBnCm8psDIegu	spotify:track:0V3wPSX9ygBn
111	111	Snow On The Beach (feat. Lana Del Rey)	Midnights	2022-10-21	4	1wtOxkiel43cVs0Yux5Q4h	spotify:track:1wtOxkiel43c
112	112	You're On Your Own, Kid	Midnights	2022-10-21	5	4D7BCuvgdJlYvlX5WlN54t	spotify:track:4D7BCuvgdJlY

Let's make some graphs to better understand the data

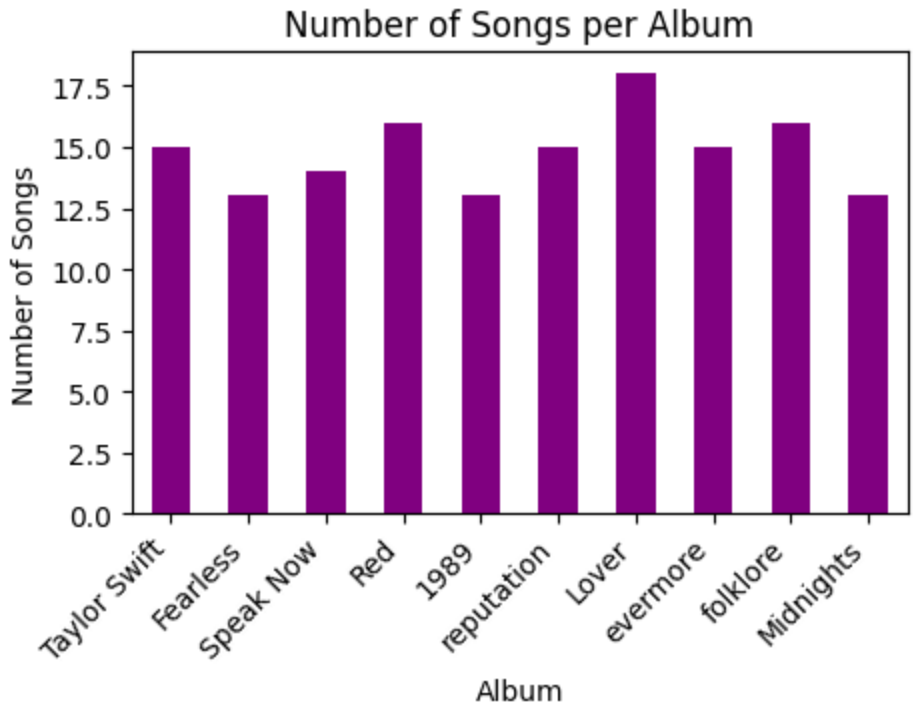
In [308…

```
songs_per_year = data.groupby('year')['name'].count()

plt.figure(figsize=(5, 3))
songs_per_year.plot(kind='bar', color='skyblue')
plt.title('Number of Songs per Year')
plt.xlabel('Year')
plt.ylabel('Number of Songs')
plt.show()
```



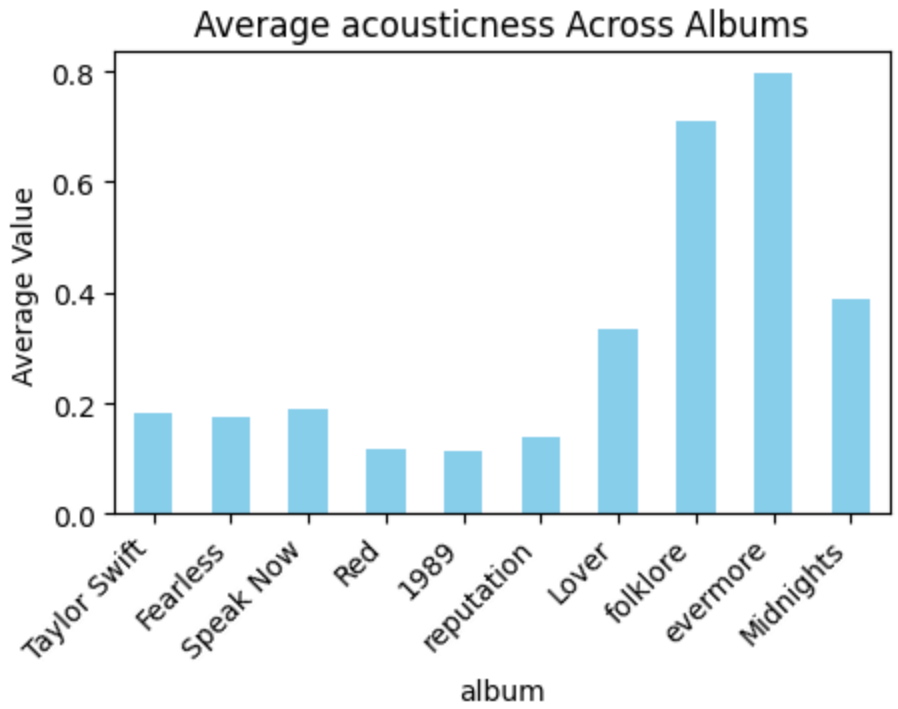
In [309…

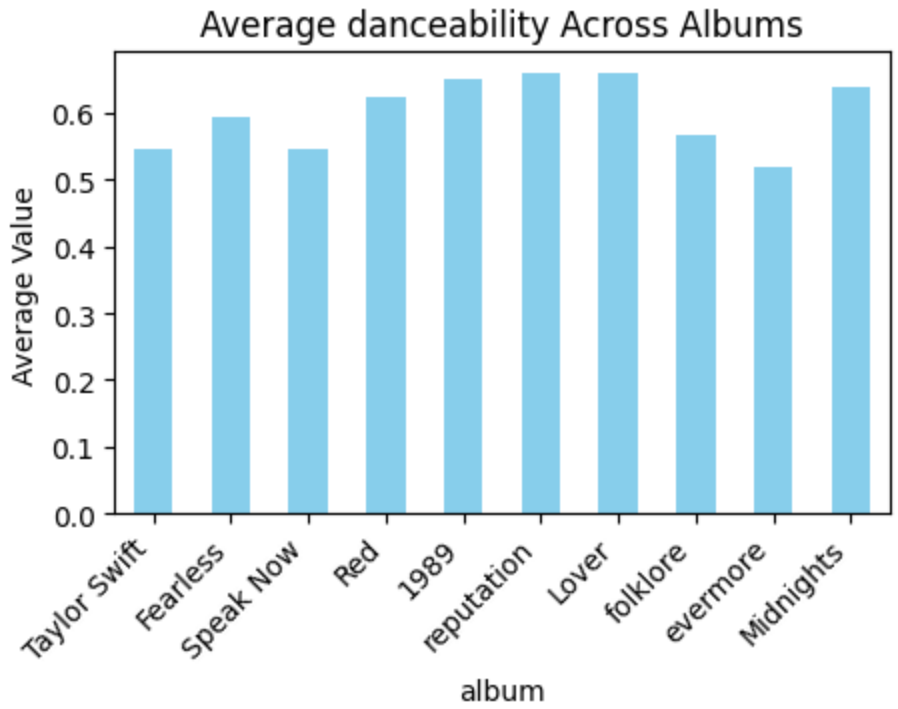
```
songs_per_album = standard_albums_data.groupby('album')['name'].count()

sorted_albums = standard_albums_data.sort_values('year')['album'].unique()

# Plotting
plt.figure(figsize=(5, 3))
songs_per_album.loc[sorted_albums].plot(kind='bar', color='purple')
plt.title('Number of Songs per Album')
plt.xlabel('Album')
plt.ylabel('Number of Songs')
plt.xticks(rotation=45, ha='right')
plt.show()
```

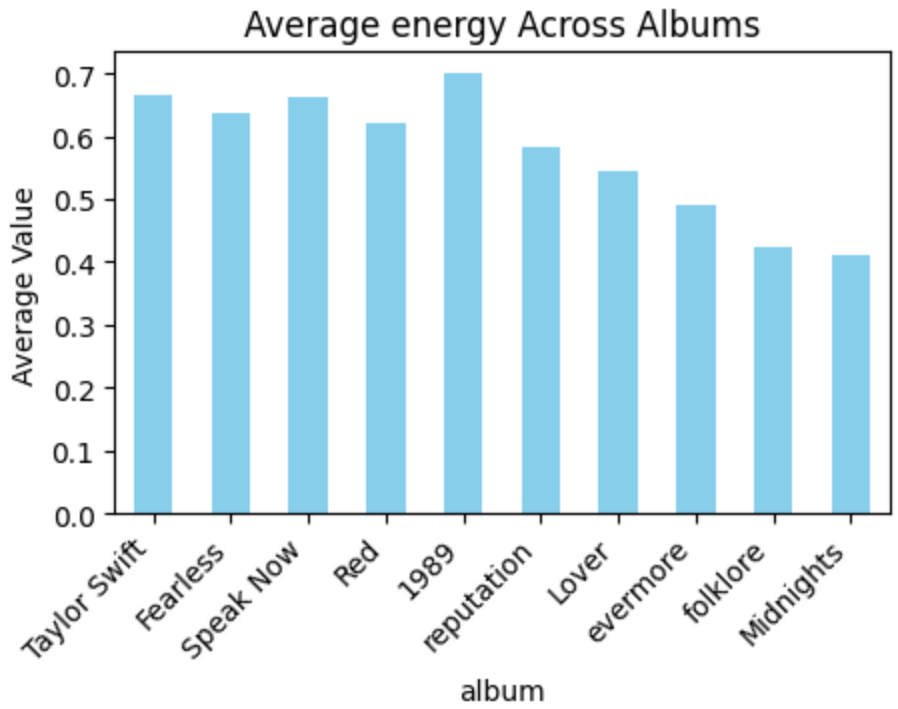


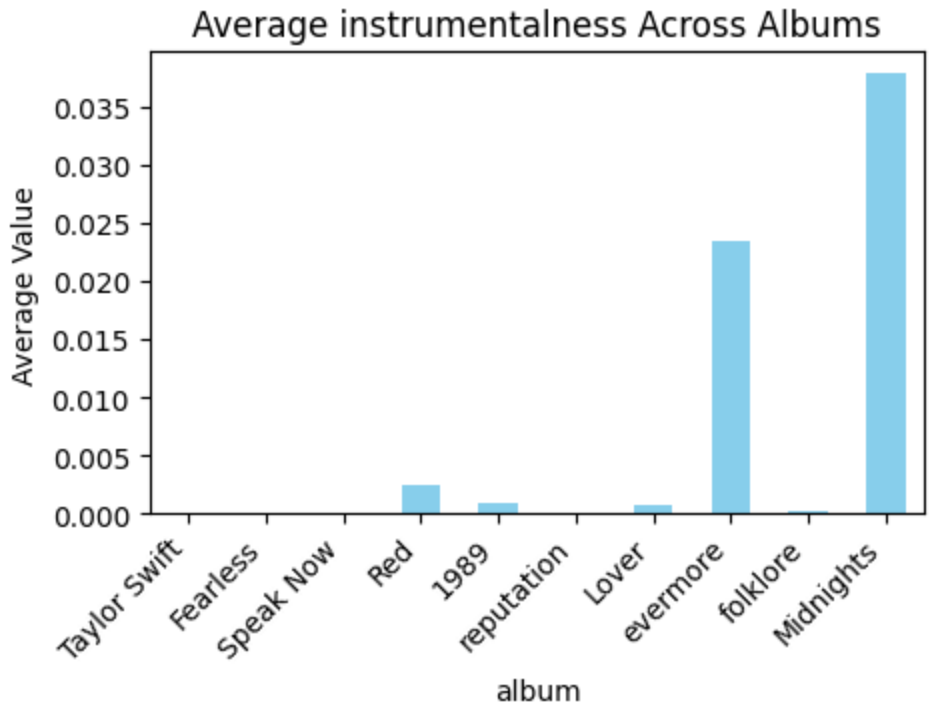
In [310…

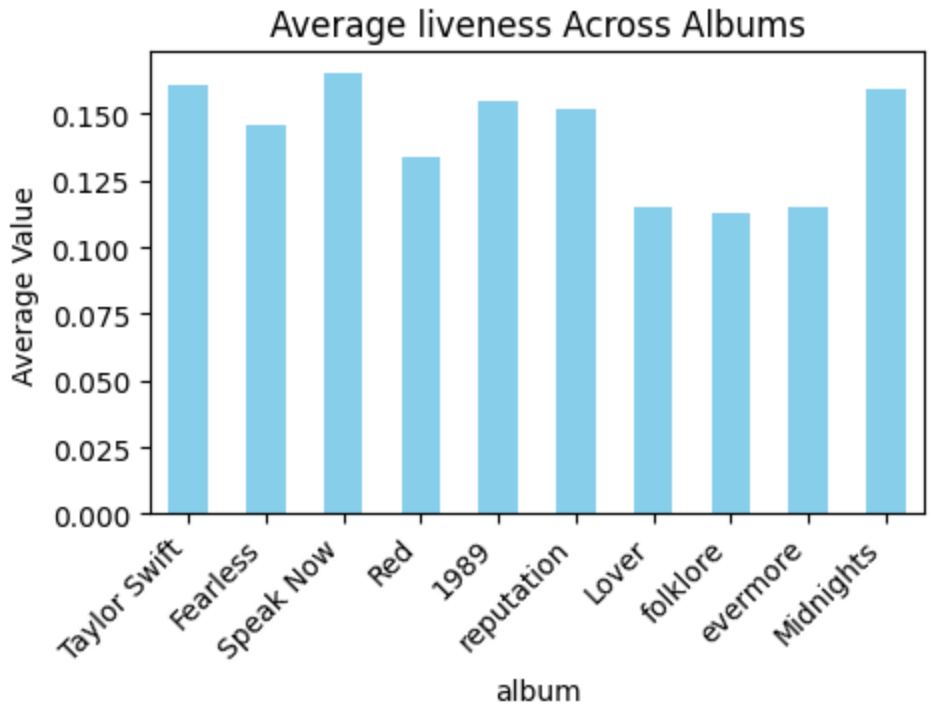
```
# Analyze columns of interest
columns = standard_albums_data.columns[7:-2]

for c in columns:
    plt.figure(figsize=(5, 3))
    plt.title(f'Average {c} Across Albums')
    plt.ylabel('Average Value')

    average_per_album = standard_albums_data.groupby('album')[c].mean()

    sorted_albums = standard_albums_data.sort_values(['year', c], ascending=[True, False])['album'].unique

    average_per_album.loc[sorted_albums].plot(kind='bar', color='skyblue')
    plt.xticks(rotation=45, ha='right')

    plt.show()
```

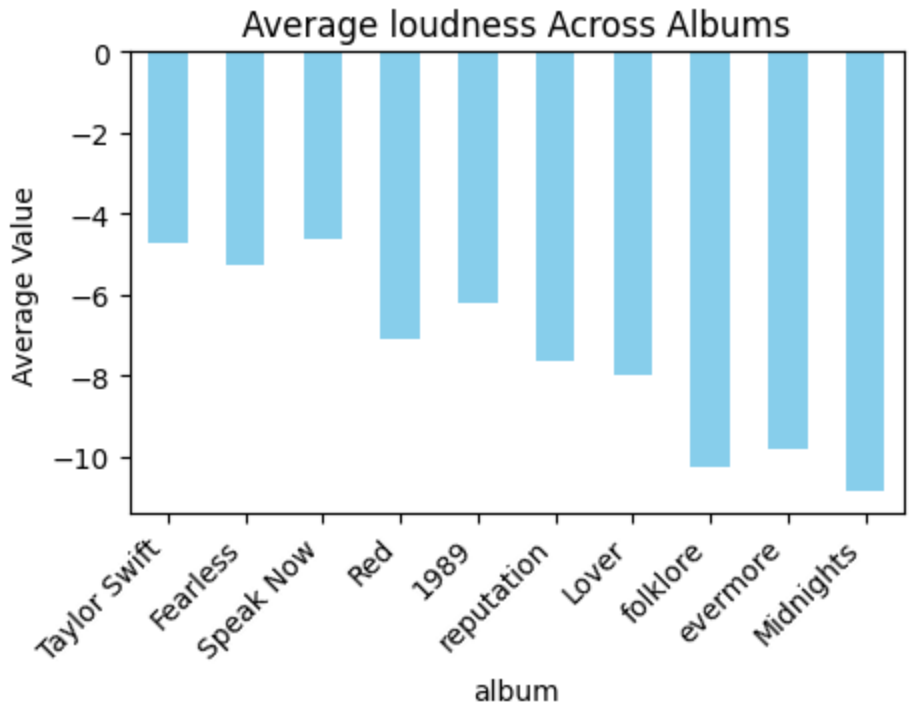

Average acousticness Across Albums
Average Value
0.0
0.2
0.4
0.6
0.8
Taylor Swift
Fearless
Speak Now
Red
1989
reputation
Lover
folklore
evermore
Midnights
album

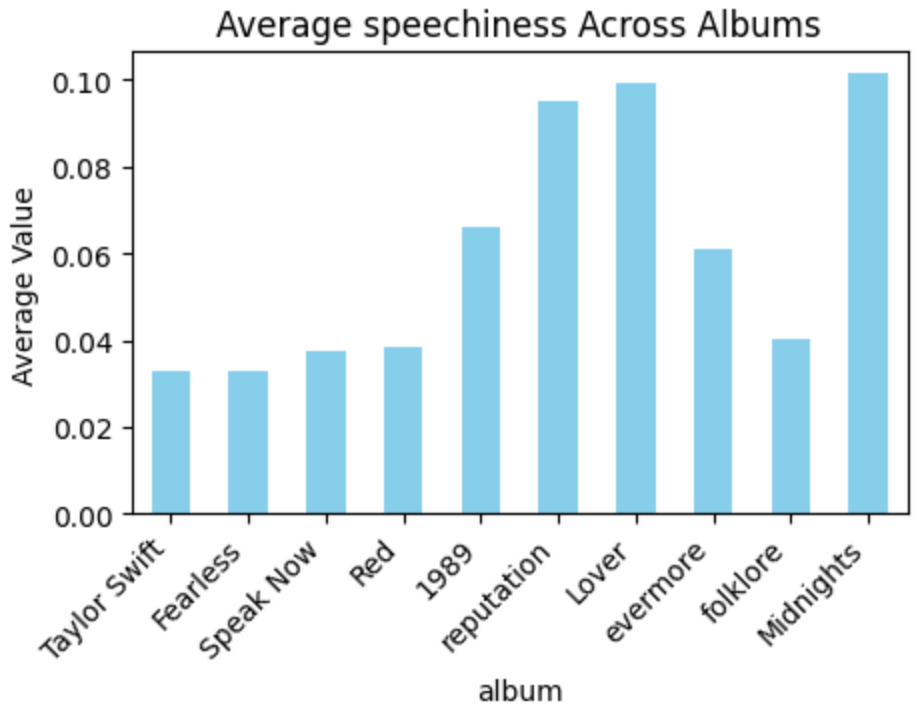

Average danceability Across Albums
Average Value
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Taylor Swift
Fearless
Speak Now
Red
1989
reputation
Lover
folklore
evermore
Midnights
album

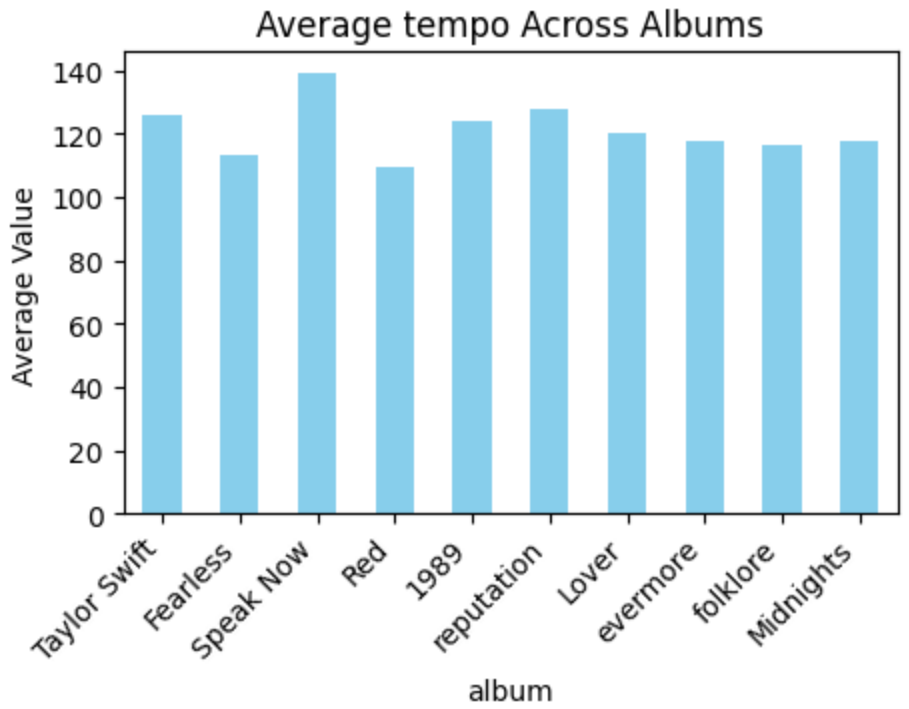

Average energy Across Albums
Average Value
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Taylor Swift
Fearless
Speak Now
Red
1989
reputation
Lover
evermore
folklore
Midnights
album

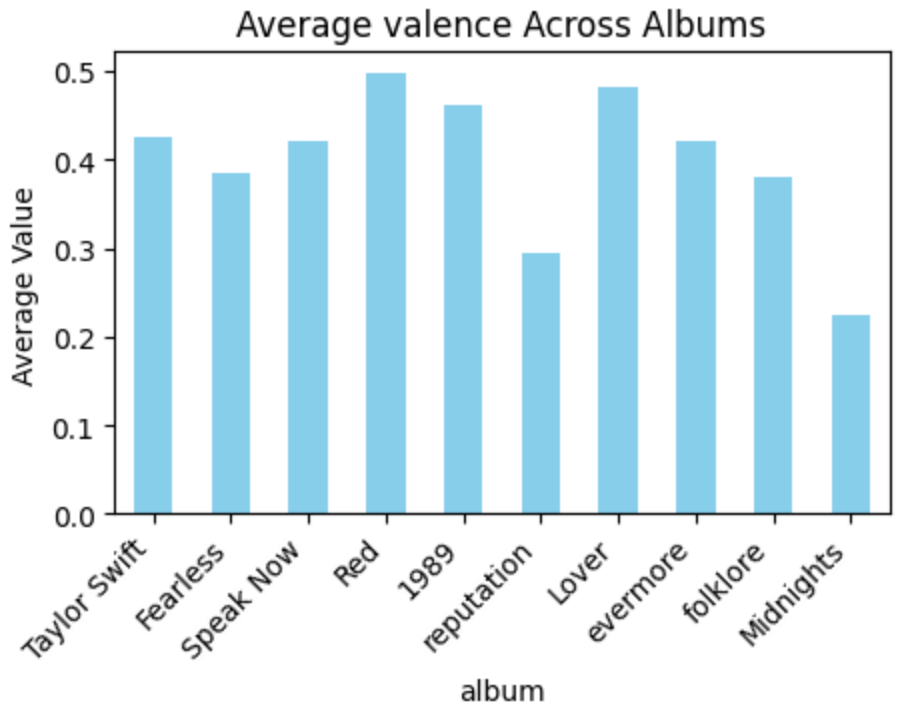

Average instrumentalness Across Albums
Average Value
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
Taylor Swift
Fearless
Speak Now
Red
1989
reputation
Lover
evermore
folklore
Midnights
album

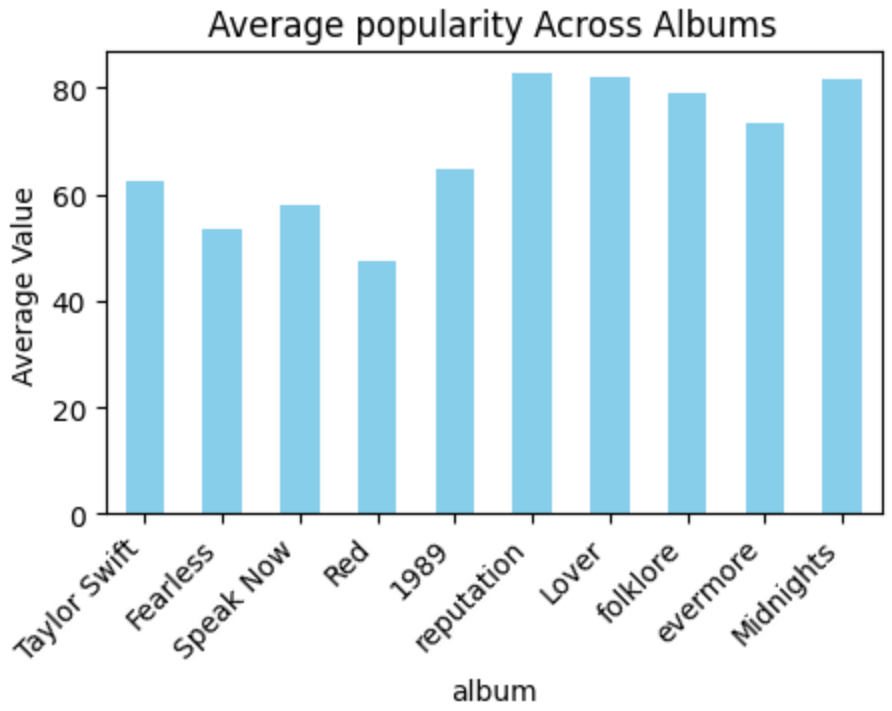

Average liveness Across Albums
Average Value
0.150
0.125
0.100
0.075
0.050
0.025
0.000
Taylor Swift
Fearless
Speak Now
Red
1989
reputation
Lover
folklore
evermore
Midnights
album


Average loudness Across Albums
Average Value
0
−2
−4
−6
−8
−10
Taylor Swift
Fearless
Speak Now
Red
1989
reputation
Lover
folklore
evermore
Midnights
album


Average speechiness Across Albums
Average Value
0.00
0.02
0.04
0.06
0.08
0.10
Taylor Swift
Fearless
Speak Now
Red
1989
reputation
Lover
evermore
folklore
Midnights
album


Average tempo Across Albums
Average Value
0
20
40
60
80
100
120
140
Taylor Swift
Fearless
Speak Now
Red
1989
reputation
Lover
evermore
folklore
Midnights
album


Average valence Across Albums
Average Value
0.0
0.1
0.2
0.3
0.4
0.5
Taylor Swift
Fearless
Speak Now
Red
1989
reputation
Lover
evermore
folklore
Midnights
album



Let's utilize some of the linear and logistic methods we've learned in class to try to make predictions about songs and albums based on their Spotify metrics.

In [311…
```
standard_albums_data.head()
X = standard_albums_data[['album', 'acousticness', 'danceability', 'energy', 'instrumentalness', 'liveness
y = standard_albums_data['popularity']

X = pd.get_dummies(X, columns=['album'], drop_first=True)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2)
```

In [312…
```
names, mses, rmses, r2s = [], [], [], []

def show_metrics(y_test, y_pred, model_type):
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, y_pred)
```

```
    result = f'''
    {model_type} MSE  : {mse}
    {model_type} RMSE : {rmse}
    {model_type} R^2  : {r2}
    '''

    names.append(model_type)
    mses.append(mse)
    rmses.append(rmse)
    r2s.append(r2)

    print(result)
```

Let's try to predict song popularity using linear regression.

In [313…
```
lr = LinearRegression()
lr.fit(X_train, y_train)
lr_pred = lr.predict(X_test)

show_metrics(y_test, lr_pred, "Linear Regression")
```

```
Linear Regression MSE  : 58.10695992805188
Linear Regression RMSE : 7.622792134648031
Linear Regression R^2  : 0.7397729745862608
```

In [314…
```
rfr = RandomForestRegressor(random_state=40)
rfr.fit(X_train, y_train)
rfr_pred = rfr.predict(X_test)

show_metrics(y_test, rfr_pred, "Random Forest Regressor")
```

```
Random Forest Regressor MSE  : 110.34071333333337
Random Forest Regressor RMSE : 10.50431879434994
Random Forest Regressor R^2  : 0.5058485997492086
```

Next, let's try using logistic regression for a classification task. Here, we want to see if we can fit songs to albums based on their Spotify metrics.

In [315…
```
X = standard_albums_data[['acousticness', 'danceability', 'energy', 'instrumentalness', 'liveness', 'loudn
y = standard_albums_data['album']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

rf_classifier = RandomForestClassifier(n_estimators=100, random_state=13)
rf_classifier.fit(X_train, y_train)

predictions = rf_classifier.predict(X_test)

accuracy = accuracy_score(y_test, predictions)
print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.26666666666666666
```

This is clearly not a very good model! Songs can have similar metrics across albums. Additionally, given that an album typically has less than 20 songs, there is limited training data and it is hard for the model to pick up on nuances.

When it comes to music, it may be better to look for patterns based on lyrics, since that's where the true essence of a song lies. Let's shift gears to analyze lyrical data using natural language processing techniques.

In [316…
```
import os
from pathlib import Path
import re

albums = []
songs = []
lyrics = []

"""
Many albums are duplicated (deluxe version, etc).
In order to prevent this sort of repetition, we're going to filter out the duplicates and live versions
of songs by taking the Taylor's Version of each album, and take the version with the most songs (most delu
If there is not a Taylor's Version of an album, we will just take the longest version of that album.
"""

standard_albums = ['TaylorSwift', 'Fearless_TaylorsVersion_', 'SpeakNow', 'Red_TaylorsVersion_', '1989', 'R

for dirname, _, filenames in os.walk('all_song_data/Albums'):
    album_name = os.path.split(dirname)[-1]
```

```
    if album_name in standard_albums:
        for filename in filenames:
                albums.append(album_name)
                songs.append(Path(filename).stem)

                with open(os.path.join(dirname, filename), 'r', encoding='utf-8') as f:
                    raw_lyrics = f.read()

                # Clean up the lyrics by replacing non-standard characters
                raw_lyrics = raw_lyrics.encode('ascii', 'replace').decode().replace('?', ' ')
                raw_lyrics = raw_lyrics.replace('\n', ' ')
                raw_lyrics = re.sub('(?!\n)\s+', ' ', raw_lyrics)

                # Remove any words between brackets (Chorus, etc.)
                pattern = re.compile(r'\[.*?\]')
                # Use sub() method to replace matched patterns with an empty string
                raw_lyrics = re.sub(pattern, '', raw_lyrics)

                # Remove lyrics header
                raw_lyrics = re.sub('.*Lyrics', '', raw_lyrics)

                # Remove end characters (number + 'Embed' or number + 'KEmbed')
                raw_lyrics = re.sub('[0-9]+KEmbed', '', raw_lyrics)
                raw_lyrics = re.sub('[0-9]+Embed', '', raw_lyrics)

                lyrics.append(raw_lyrics)

preprocessed_df = pd.DataFrame({'Album': albums, 'Song': songs, 'Lyrics': lyrics})
preprocessed_df.dropna()
preprocessed_df.head()
```

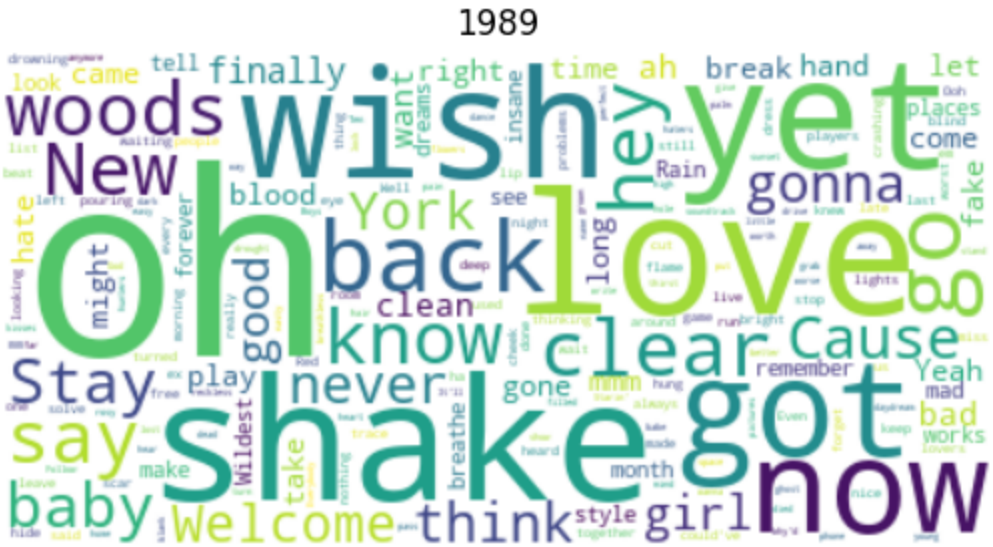
Out[316...

	Album	Song	Lyrics
0	Fearless_TaylorsVersion_	TheWayILovedYou_TaylorsVersion_	He is sensible and so incredible And all my s...
1	Fearless_TaylorsVersion_	Fifteen_TaylorsVersion_	You take a deep breath and you walk through t...
2	Fearless_TaylorsVersion_	DontYou_TaylorsVersion__FromtheVault_	Hey, I knew I'd run into you somewhere It's b...
3	Fearless_TaylorsVersion_	TheOtherSideoftheDoor_TaylorsVersion_	In the heat of the fight I walked away Ignori...
4	Fearless_TaylorsVersion_	JumpThenFall_TaylorsVersion_	I like the way you sound in the mornin' We're...

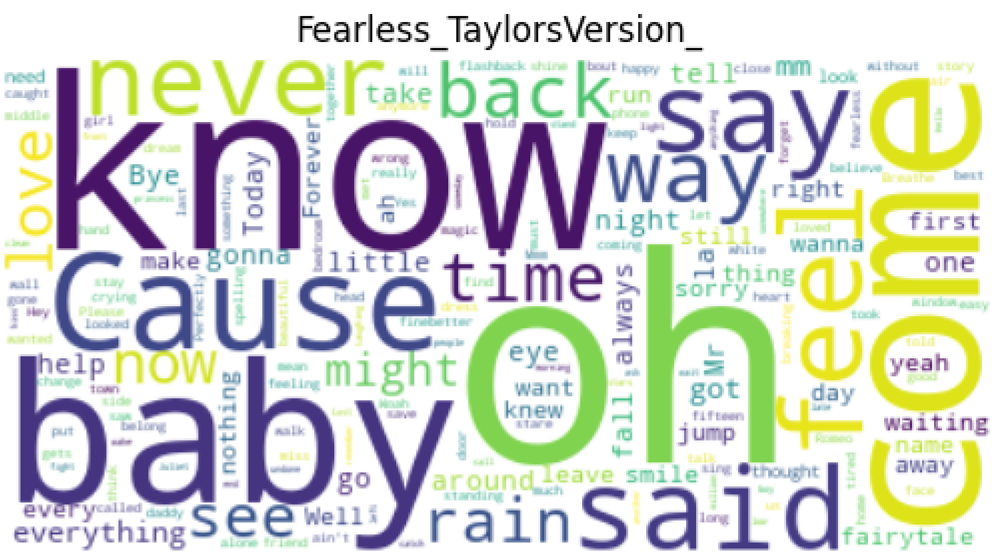
A common practice in NLP when doing sentiment analysis is creating word clouds to get a sense of the overall trends in a dataset and to make sure data is being represented the way we'd like.

In [317…

```
# Create word cloud and plot visualization for each album
from wordcloud import WordCloud
import matplotlib.pyplot as plt

album_group = preprocessed_df.groupby('Album')

for name, group in album_group:
    wordcloud = WordCloud(collocations = False,
                          background_color = 'white').generate(' '.join(group['Lyrics']))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.title(name)
    plt.axis("off")
    plt.show()
```




Evermore
know
now
never
think
ooh
love
one
time
take
might
died
man
better
right
day
good
hand
yeah
stay
thing
go
ah
got
eyes
Oh
Dorothea
closure
happiness
Evermore
Champagne
gonna
life
watch
problems
alive
covered
begging
prove
face
say
short
taken
feel
still
wrong
Stop
see
house
around
look
gold
meet
plans
Cause
Yes
making
call
pain
head
dead
leave
sit
lost
bone
goddamn
Wreck
real
story
rush
body
tell
back
bad
goes
every
always
light
letter
Long
stone
old
place
will
Past
way


Fearless_TaylorsVersion_
know
oh
baby
Cause
come
feel
said
say
never
back
way
love
rain
time
might
now
see
little
night
always
Today
Forever
Bye
gonna
make
help
go
everything
every
Well
around
leave
smile
thought
fairytale
waiting
yeah
got
eye
want
knew
jump
day
name
away
first
one
wanna
thing
sorry
still
right
take
tell
run
look
need
nothing
Mr
la

Folklore

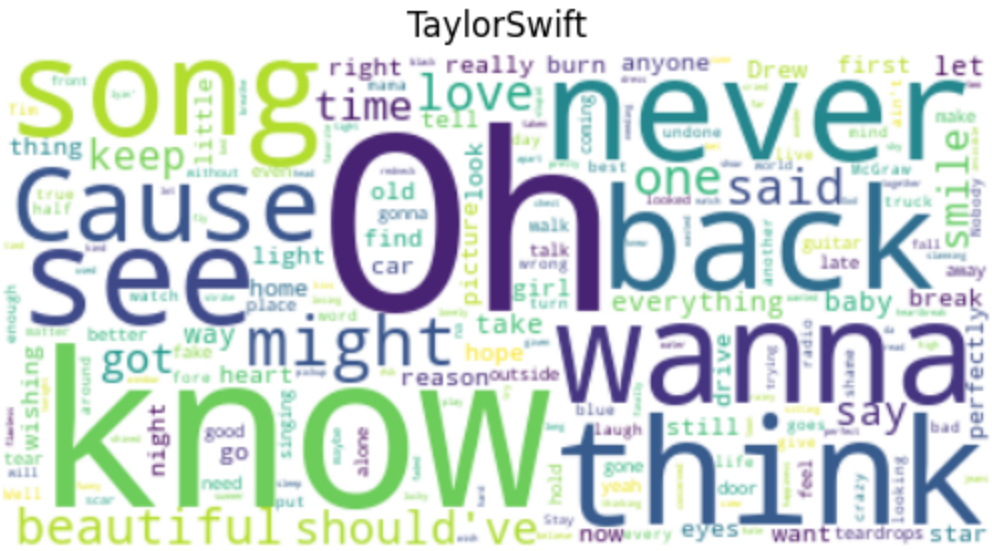
Lover

Midnights_TheTillDawnEdition_

Red_TaylorsVersion_

Reputation

SpeakNow



These word clouds give us a sense of common words used throughout albums. They also highlight some words we might not want to include in our NLP techniques - these include words with minimal emotional value like "oh" and "ooh."

Now, we will perform sentiment analysis using NLTK and Textblob. We'll compare the results in a nicely formatted table to keep track of differences.

In [318…

```
# Sentiment analysis by album
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer
from textblob import TextBlob
from termcolor import colored

# Display results in an easy-to-interpret table. Positive = green, negative = red
from prettytable import PrettyTable
nltk.download('vader_lexicon')

# Create a PrettyTable object
table = PrettyTable()
table.field_names = ["Song Title", "NLTK", "TextBlob"]

# Function for sentiment analysis using NLTK's VADER sentiment analysis tool
def analyze_sentiment_nltk(text):
```

```
    sia = SentimentIntensityAnalyzer()
    sentiment_scores = sia.polarity_scores(text)

    if sentiment_scores['compound'] >= 0.05:
        return 'Positive'
    elif sentiment_scores['compound'] <= -0.05:
        return 'Negative'
    else:
        return 'Neutral'

# Function for sentiment analysis using TextBlob
def analyze_sentiment_textblob(text):
    blob = TextBlob(text)

    if blob.sentiment.polarity > 0:
        return 'Positive'
    elif blob.sentiment.polarity < 0:
        return 'Negative'
    else:
        return 'Neutral'

# function to get the appropriate color for the given sentiment
def get_color(sentiment):
    if sentiment == 'Positive':
        return 'green'
    else:
        return 'red'

for name, group in album_group:

    songs = group.groupby('Song')

    for title, lyrics in songs:

        # Sentiment analysis using NLTK
        sentiment_nltk = analyze_sentiment_nltk(' '.join(lyrics['Lyrics']))
        nltk_color = get_color(sentiment_nltk)

        # Sentiment analysis using TextBlob
        sentiment_textblob = analyze_sentiment_textblob(' '.join(lyrics['Lyrics']))
        textblob_color = get_color(sentiment_textblob)
```

```
        row = [title, colored(sentiment_nltk, nltk_color), colored(sentiment_textblob, textblob_color)]
        table.add_row(row)

print(table)
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     /Users/neetidesai/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

Song Title	NLTK	TextBlob
1989_Booklet_	Positive	Positive
AllYouHadtoDoWasStay	Positive	Positive
BadBlood	Negative	Negative
BlankSpace	Negative	Negative
Clean	Positive	Positive
HowYouGetTheGirl	Negative	Negative
IKnowPlaces	Negative	Negative
IWishYouWould	Negative	Positive
OutOfTheWoods	Positive	Positive
ShakeItOff	Negative	Negative
Style	Positive	Negative
ThisLove	Positive	Positive
WelcometoNewYork	Positive	Positive
WildestDreams	Positive	Positive
champagneproblems	Negative	Positive
closure	Positive	Positive
coneyisland	Positive	Positive
cowboylikeme	Negative	Positive
dorothea	Negative	Negative
evermore	Negative	Positive
goldrush	Positive	Positive
happiness	Positive	Positive
ivy	Negative	Positive
longstoryshort	Negative	Positive
marjorie	Positive	Positive
nobody_nocrime	Negative	Positive
tisthedamnseason	Positive	Positive
tolerateit	Positive	Positive
willow	Positive	Positive
Breathe_TaylorsVersion_	Positive	Negative
ByeByeBaby_TaylorsVersion__FromtheVault_	Negative	Negative
Change_TaylorsVersion_	Positive	Positive
ComeInWithTheRain_TaylorsVersion_	Positive	Positive
DontYou_TaylorsVersion__FromtheVault_	Positive	Positive
Fearless_TaylorsVersion_	Positive	Positive
Fifteen_TaylorsVersion_	Positive	Positive
Forever_Always_PianoVersion__TaylorsVersion_	Negative	Positive
Forever_Always_TaylorsVersion_	Negative	Positive
HeyStephen_TaylorsVersion_	Positive	Positive

JumpThenFall_TaylorsVersion_	Positive	Positive
LoveStory_TaylorsVersion_	Positive	Positive
LoveStory_TaylorsVersion__ElviraRemix_	Positive	Positive
Mr_PerfectlyFine_TaylorsVersion__FromtheVault_	Positive	Positive
Superstar_TaylorsVersion_	Positive	Positive
TellMeWhy_TaylorsVersion_	Negative	Negative
ThatsWhen_TaylorsVersion__FromtheVault_	Positive	Negative
TheBestDay_TaylorsVersion_	Positive	Positive
TheOtherSideoftheDoor_TaylorsVersion_	Positive	Positive
TheWayILovedYou_TaylorsVersion_	Positive	Positive
TodayWasaFairytale_TaylorsVersion_	Positive	Positive
Untouchable_TaylorsVersion_	Positive	Negative
WeWereHappy_TaylorsVersion__FromtheVault_	Positive	Positive
WhiteHorse_TaylorsVersion_	Positive	Negative
YouAllOverMe_TaylorsVersion__FromtheVault_	Positive	Positive
YouBelongWithMe_TaylorsVersion_	Positive	Positive
YoureNotSorry_TaylorsVersion_	Positive	Positive
august	Positive	Negative
betty	Positive	Negative
cardigan	Positive	Positive
epiphany	Positive	Negative
exile	Negative	Positive
hoax	Negative	Positive
illicitaffairs	Negative	Negative
invisiblestring	Positive	Positive
madwoman	Negative	Negative
mirrorball	Positive	Positive
mytearsricochet	Negative	Negative
peace	Negative	Positive
seven	Positive	Positive
the1	Positive	Positive
thelastgreatamericandynasty	Positive	Positive
thisismetrying	Negative	Negative
Afterglow	Negative	Positive
CorneliaStreet	Positive	Negative
CruelSummer	Negative	Negative
Daylight	Positive	Positive
DeathbyaThousandCuts	Positive	Positive
FalseGod	Positive	Negative
IForgotThatYouExisted	Positive	Positive
IThinkHeKnows	Positive	Negative
ItsNicetoHaveaFriend	Positive	Positive

LondonBoy	Positive	Positive
Lover	Positive	Positive
ME_	Positive	Positive
MissAmericana_TheHeartbreakPrince	Negative	Positive
PaperRings	Positive	Negative
SoonYoullGetBetter	Positive	Positive
TheArcher	Positive	Positive
TheMan	Negative	Negative
YouNeedToCalmDown	Positive	Positive
Anti_Hero	Negative	Positive
Bejeweled	Positive	Positive
BiggerThanTheWholeSky	Negative	Positive
DearReader	Positive	Positive
Glitch	Positive	Negative
HighInfidelity	Negative	Positive
HitsDifferent	Positive	Positive
Karma	Negative	Negative
Karma_Remix_	Positive	Positive
Labyrinth	Positive	Positive
LavenderHaze	Negative	Positive
Maroon	Negative	Negative
Mastermind	Positive	Negative
MidnightRain	Positive	Positive
Paris	Positive	Positive
Question____	Positive	Positive
SnowOnTheBeach	Positive	Negative
SnowOnTheBeach_feat_MoreLanaDelRey_	Positive	Negative
SweetNothing	Positive	Positive
TheGreatWar	Positive	Positive
VigilanteShit	Positive	Negative
Wouldve_Couldve_Shouldve	Negative	Positive
YoureOnYourOwn_Kid	Positive	Positive
22_TaylorsVersion_	Positive	Positive
AMessageFromTaylor	Positive	Positive
AllTooWell_10MinuteVersion__TaylorsVersion__FromtheVault_	Positive	Positive
AllTooWell_TaylorsVersion_	Positive	Positive
Babe_TaylorsVersion__FromtheVault_	Positive	Negative
BeginAgain_TaylorsVersion_	Positive	Positive
BetterMan_TaylorsVersion__FromtheVault_	Positive	Positive
ComeBack___BeHere_TaylorsVersion_	Positive	Positive
EverythingHasChanged_TaylorsVersion_	Positive	Positive
ForeverWinter_TaylorsVersion__FromtheVault_	Positive	Positive

GirlAtHome_TaylorsVersion_	Neutral	Positive
HolyGround_TaylorsVersion_	Positive	Positive
IAlmostDo_TaylorsVersion_	Positive	Negative
IBetYouThinkAboutMe_TaylorsVersion__FromtheVault_	Positive	Positive
IKnewYouWereTrouble_TaylorsVersion_	Negative	Negative
MessageInABottle_TaylorsVersion__FromtheVault_	Positive	Positive
NothingNew_TaylorsVersion__FromtheVault_	Positive	Positive
Red_TaylorsVersion_	Positive	Positive
Ronan_TaylorsVersion_	Positive	Positive
Run_TaylorsVersion__FromtheVault_	Positive	Negative
SadBeautifulTragic_TaylorsVersion_	Positive	Positive
Starlight_TaylorsVersion_	Positive	Positive
StateOfGrace_AcousticVersion__TaylorsVersion_	Positive	Positive
StateofGrace_TaylorsVersion_	Positive	Positive
StayStayStay_TaylorsVersion_	Positive	Positive
TheLastTime_TaylorsVersion_	Positive	Positive
TheLuckyOne_TaylorsVersion_	Positive	Positive
TheMomentIKnew_TaylorsVersion_	Positive	Negative
TheVeryFirstNight_TaylorsVersion__FromtheVault_	Positive	Positive
Treacherous_TaylorsVersion_	Negative	Positive
WeAreNeverEverGettingBackTogether_TaylorsVersion_	Positive	Positive
CallItWhatYouWant	Positive	Positive
DancingWithOurHandsTied	Positive	Negative
Delicate	Positive	Positive
DontBlameMe	Negative	Positive
Dress	Positive	Positive
EndGame	Positive	Negative
GetawayCar	Negative	Positive
Gorgeous	Positive	Positive
IDidSomethingBad	Positive	Positive
IfYoureAnythingLikeMe_Poem_	Positive	Positive
KingofMyHeart	Positive	Negative
LookWhatYouMadeMeDo	Negative	Negative
NewYearsDay	Positive	Negative
ReputationMagazineVol_1	Neutral	Neutral
Reputation_Prologue_	Positive	Positive
SoItGoes___	Positive	Negative
ThisIsWhyWeCantHaveNiceThings	Positive	Positive
WhySheDisappeared_Poem_	Negative	Positive
___ReadyforIt_	Positive	Positive
BacktoDecember	Positive	Positive
BetterThanRevenge	Positive	Positive

DearJohn	Negative	Negative
Enchanted	Positive	Positive
Haunted	Negative	Negative
Innocent	Positive	Positive
LastKiss	Positive	Positive
LongLive	Negative	Positive
Mean	Negative	Negative
Mine	Positive	Positive
NeverGrowUp	Positive	Positive
SparksFly	Positive	Positive
SpeakNow	Positive	Positive
TheStoryOfUs	Negative	Positive
APerfectlyGoodHeart	Positive	Positive
APlaceInThisWorld	Positive	Positive
ColdasYou	Negative	Positive
ImOnlyMeWhenImWithYou	Positive	Positive
Invisible	Positive	Positive
MarysSong_OhMyMyMy_	Positive	Positive
OurSong	Positive	Negative
PicturetoBurn	Negative	Negative
ShouldveSaidNo	Negative	Positive
StayBeautiful	Positive	Positive
TaylorSwift_LinerNotes_	Positive	Positive
TeardropsOnMyGuitar	Positive	Positive
TeardropsonMyGuitar_PopVersion_	Positive	Positive
TheOutside	Negative	Positive
TiedTogetherwithaSmile	Negative	Positive
TimMcGraw	Positive	Positive

https://github.com/UtkarshRedd/Negation_handlingThe above results are okay, but not great. The basic idea behind sentiment analysis is that each word is analyze out of context (so for example, if the word 'important' is seen in Would've, Could've, Should've, it will be classified as positive even though in context the lyric is negative (But, Lord, you made me feel important/And then you tried to erase us)).

So, to fix this issue and get a more accurate sentiment analysis, we need to analyze words in context!

I will be using the code in this repo: https://github.com/UtkarshRedd/Negation_handling to do so.

In [319…

```
# Create a PrettyTable object
table = PrettyTable()
table.field_names = ["Song Title", "NLTK", "TextBlob", "NLTK Negation"]

def negation_handler(sentence):
    temp = int(0)
    for i in range(len(sentence)):
        if sentence[i-1] in ['not',"n't"]:
            antonyms = []
            for syn in wordnet.synsets(sentence[i]):
                syns = wordnet.synsets(sentence[i])
                w1 = syns[0].name()
                temp = 0
                for l in syn.lemmas():
                    if l.antonyms():
                        antonyms.append(l.antonyms()[0].name())
                max_dissimilarity = 0
                for ant in antonyms:
                    syns = wordnet.synsets(ant)
                    w2 = syns[0].name()
                    syns = wordnet.synsets(sentence[i])
                    w1 = syns[0].name()
                    word1 = wordnet.synset(w1)
                    word2 = wordnet.synset(w2)
                    if isinstance(word1.wup_similarity(word2), float) or isinstance(word1.wup_similarity(w
                        temp = 1 - word1.wup_similarity(word2)
                    if temp>max_dissimilarity:
                        max_dissimilarity = temp
                        antonym_max = ant
                        sentence[i] = antonym_max
                        sentence[i-1] = ''
    return sentence


# Function for sentiment analysis using NLTK's VADER sentiment analysis tool
def analyze_sentiment_nltk(text):
    sia = SentimentIntensityAnalyzer()
    sentiment_scores = sia.polarity_scores(text)

    if sentiment_scores['compound'] >= 0.05:
        return 'Positive'
```

```
    elif sentiment_scores['compound'] <= -0.05:
        return 'Negative'
    else:
        return 'Neutral'

# function to get the appropriate color for the given sentiment
def get_color(sentiment):
    if sentiment == 'Positive':
        return 'green'
    else:
        return 'red'

for name, group in album_group:
    songs = group.groupby('Song')

    for title, lyrics in songs:

        # Sentiment analysis using NLTK
        sentiment_nltk = analyze_sentiment_nltk(' '.join(lyrics['Lyrics']))
        nltk_color = get_color(sentiment_nltk)

        # Sentiment analysis using TextBlob
        sentiment_textblob = analyze_sentiment_textblob(' '.join(lyrics['Lyrics']))
        textblob_color = get_color(sentiment_textblob)

        # Sentiment analysis using NLTK with negation handling
        lyrics = negation_handler(' '.join(lyrics['Lyrics']))
        sentiment_nltk_negation = analyze_sentiment_nltk(lyrics)
        nltk_negation_color = get_color(sentiment_nltk_negation)

        row = [title, colored(sentiment_nltk, nltk_color), colored(sentiment_textblob, textblob_color), co
        table.add_row(row)

print(table)
```

Song Title	NLTK	TextBlob	NLTK Negation
1989_Booklet_	Positive	Positive	Positive
AllYouHadtoDoWasStay	Positive	Positive	Positive
BadBlood	Negative	Negative	Negative
BlankSpace	Negative	Negative	Negative
Clean	Positive	Positive	Positive
HowYouGetTheGirl	Negative	Negative	Negative
IKnowPlaces	Negative	Negative	Negative
IWishYouWould	Negative	Positive	Negative
OutOfTheWoods	Positive	Positive	Positive
ShakeItOff	Negative	Negative	Negative
Style	Positive	Negative	Positive
ThisLove	Positive	Positive	Positive
WelcometoNewYork	Positive	Positive	Positive
WildestDreams	Positive	Positive	Positive
champagneproblems	Negative	Positive	Negative
closure	Positive	Positive	Positive
coneyisland	Positive	Positive	Positive
cowboylikeme	Negative	Positive	Negative
dorothea	Negative	Negative	Negative
evermore	Negative	Positive	Negative
goldrush	Positive	Positive	Positive
happiness	Positive	Positive	Positive
ivy	Negative	Positive	Negative
longstoryshort	Negative	Positive	Negative
marjorie	Positive	Positive	Positive
nobody_nocrime	Negative	Positive	Negative
tisthedamnseason	Positive	Positive	Positive
tolerateit	Positive	Positive	Positive
willow	Positive	Positive	Positive
Breathe_TaylorsVersion_	Positive	Negative	Positive
ByeByeBaby_TaylorsVersion__FromtheVault_	Negative	Negative	Negative
Change_TaylorsVersion_	Positive	Positive	Positive
ComeInWithTheRain_TaylorsVersion_	Positive	Positive	Positive
DontYou_TaylorsVersion__FromtheVault_	Positive	Positive	Positive
Fearless_TaylorsVersion_	Positive	Positive	Positive
Fifteen_TaylorsVersion_	Positive	Positive	Positive
Forever_Always_PianoVersion__TaylorsVersion_	Negative	Positive	Negative
Forever_Always_TaylorsVersion_	Negative	Positive	Negative
HeyStephen_TaylorsVersion_	Positive	Positive	Positive

JumpThenFall_TaylorsVersion_	Positive	Positive	Positive
LoveStory_TaylorsVersion_	Positive	Positive	Positive
LoveStory_TaylorsVersion__ElviraRemix_	Positive	Positive	Positive
Mr_PerfectlyFine_TaylorsVersion__FromtheVault_	Positive	Positive	Positive
Superstar_TaylorsVersion_	Positive	Positive	Positive
TellMeWhy_TaylorsVersion_	Negative	Negative	Negative
ThatsWhen_TaylorsVersion__FromtheVault_	Positive	Negative	Positive
TheBestDay_TaylorsVersion_	Positive	Positive	Positive
TheOtherSideoftheDoor_TaylorsVersion_	Positive	Positive	Positive
TheWayILovedYou_TaylorsVersion_	Positive	Positive	Positive
TodayWasaFairytale_TaylorsVersion_	Positive	Positive	Positive
Untouchable_TaylorsVersion_	Positive	Negative	Positive
WeWereHappy_TaylorsVersion__FromtheVault_	Positive	Positive	Positive
WhiteHorse_TaylorsVersion_	Positive	Negative	Positive
YouAllOverMe_TaylorsVersion__FromtheVault_	Positive	Positive	Positive
YouBelongWithMe_TaylorsVersion_	Positive	Positive	Positive
YoureNotSorry_TaylorsVersion_	Positive	Positive	Positive
august	Positive	Negative	Positive
betty	Positive	Negative	Positive
cardigan	Positive	Positive	Positive
epiphany	Positive	Negative	Positive
exile	Negative	Positive	Negative
hoax	Negative	Positive	Negative
illicitaffairs	Negative	Negative	Negative
invisiblestring	Positive	Positive	Positive
madwoman	Negative	Negative	Negative
mirrorball	Positive	Positive	Positive
mytearsricochet	Negative	Negative	Negative
peace	Negative	Positive	Negative
seven	Positive	Positive	Positive
the1	Positive	Positive	Positive
thelastgreatamericandynasty	Positive	Positive	Positive
thisismetrying	Negative	Negative	Negative
Afterglow	Negative	Positive	Negative
CorneliaStreet	Positive	Negative	Positive
CruelSummer	Negative	Negative	Negative
Daylight	Positive	Positive	Positive
DeathbyaThousandCuts	Positive	Positive	Positive
FalseGod	Positive	Negative	Positive
IForgotThatYouExisted	Positive	Positive	Positive
IThinkHeKnows	Positive	Negative	Positive
ItsNicetoHaveaFriend	Positive	Positive	Positive

LondonBoy	Positive	Positive	Positive
Lover	Positive	Positive	Positive
ME_	Positive	Positive	Positive
MissAmericana_TheHeartbreakPrince	Negative	Positive	Negative
PaperRings	Positive	Negative	Positive
SoonYoullGetBetter	Positive	Positive	Positive
TheArcher	Positive	Positive	Positive
TheMan	Negative	Negative	Negative
YouNeedToCalmDown	Positive	Positive	Positive
Anti_Hero	Negative	Positive	Negative
Bejeweled	Positive	Positive	Positive
BiggerThanTheWholeSky	Negative	Positive	Negative
DearReader	Positive	Positive	Positive
Glitch	Positive	Negative	Positive
HighInfidelity	Negative	Positive	Negative
HitsDifferent	Positive	Positive	Positive
Karma	Negative	Negative	Negative
Karma_Remix_	Positive	Positive	Positive
Labyrinth	Positive	Positive	Positive
LavenderHaze	Negative	Positive	Negative
Maroon	Negative	Negative	Negative
Mastermind	Positive	Negative	Positive
MidnightRain	Positive	Positive	Positive
Paris	Positive	Positive	Positive
Question____	Positive	Positive	Positive
SnowOnTheBeach	Positive	Negative	Positive
SnowOnTheBeach_feat_MoreLanaDelRey_	Positive	Negative	Positive
SweetNothing	Positive	Positive	Positive
TheGreatWar	Positive	Positive	Positive
VigilanteShit	Positive	Negative	Positive
Wouldve_Couldve_Shouldve	Negative	Positive	Negative
YoureOnYourOwn_Kid	Positive	Positive	Positive
22_TaylorsVersion_	Positive	Positive	Positive
AMessageFromTaylor	Positive	Positive	Positive
AllTooWell_10MinuteVersion__TaylorsVersion__FromtheVault_	Positive	Positive	Positive
AllTooWell_TaylorsVersion_	Positive	Positive	Positive
Babe_TaylorsVersion__FromtheVault_	Positive	Negative	Positive
BeginAgain_TaylorsVersion_	Positive	Positive	Positive
BetterMan_TaylorsVersion__FromtheVault_	Positive	Positive	Positive
ComeBack___BeHere_TaylorsVersion_	Positive	Positive	Positive
EverythingHasChanged_TaylorsVersion_	Positive	Positive	Positive
ForeverWinter_TaylorsVersion__FromtheVault_	Positive	Positive	Positive

GirlAtHome_TaylorsVersion_	Neutral	Positive	Neutral
HolyGround_TaylorsVersion_	Positive	Positive	Positive
IAlmostDo_TaylorsVersion_	Positive	Negative	Positive
IBetYouThinkAboutMe_TaylorsVersion__FromtheVault_	Positive	Positive	Positive
IKnewYouWereTrouble_TaylorsVersion_	Negative	Negative	Negative
MessageInABottle_TaylorsVersion__FromtheVault_	Positive	Positive	Positive
NothingNew_TaylorsVersion__FromtheVault_	Positive	Positive	Positive
Red_TaylorsVersion_	Positive	Positive	Positive
Ronan_TaylorsVersion_	Positive	Positive	Positive
Run_TaylorsVersion__FromtheVault_	Positive	Negative	Positive
SadBeautifulTragic_TaylorsVersion_	Positive	Positive	Positive
Starlight_TaylorsVersion_	Positive	Positive	Positive
StateOfGrace_AcousticVersion__TaylorsVersion_	Positive	Positive	Positive
StateofGrace_TaylorsVersion_	Positive	Positive	Positive
StayStayStay_TaylorsVersion_	Positive	Positive	Positive
TheLastTime_TaylorsVersion_	Positive	Positive	Positive
TheLuckyOne_TaylorsVersion_	Positive	Positive	Positive
TheMomentIKnew_TaylorsVersion_	Positive	Negative	Positive
TheVeryFirstNight_TaylorsVersion__FromtheVault_	Positive	Positive	Positive
Treacherous_TaylorsVersion_	Negative	Positive	Negative
WeAreNeverEverGettingBackTogether_TaylorsVersion_	Positive	Positive	Positive
CallItWhatYouWant	Positive	Positive	Positive
DancingWithOurHandsTied	Positive	Negative	Positive
Delicate	Positive	Positive	Positive
DontBlameMe	Negative	Positive	Negative
Dress	Positive	Positive	Positive
EndGame	Positive	Negative	Positive
GetawayCar	Negative	Positive	Negative
Gorgeous	Positive	Positive	Positive
IDidSomethingBad	Positive	Positive	Positive
IfYoureAnythingLikeMe_Poem_	Positive	Positive	Positive
KingofMyHeart	Positive	Negative	Positive
LookWhatYouMadeMeDo	Negative	Negative	Negative
NewYearsDay	Positive	Negative	Positive
ReputationMagazineVol_1	Neutral	Neutral	Neutral
Reputation_Prologue_	Positive	Positive	Positive
SoItGoes___	Positive	Negative	Positive
ThisIsWhyWeCantHaveNiceThings	Positive	Positive	Positive
WhySheDisappeared_Poem_	Negative	Positive	Negative
___ReadyforIt_	Positive	Positive	Positive
BacktoDecember	Positive	Positive	Positive
BetterThanRevenge	Positive	Positive	Positive

DearJohn	Negative	Negative	Negative
Enchanted	Positive	Positive	Positive
Haunted	Negative	Negative	Negative
Innocent	Positive	Positive	Positive
LastKiss	Positive	Positive	Positive
LongLive	Negative	Positive	Negative
Mean	Negative	Negative	Negative
Mine	Positive	Positive	Positive
NeverGrowUp	Positive	Positive	Positive
SparksFly	Positive	Positive	Positive
SpeakNow	Positive	Positive	Positive
TheStoryOfUs	Negative	Positive	Negative
APerfectlyGoodHeart	Positive	Positive	Positive
APlaceInThisWorld	Positive	Positive	Positive
ColdasYou	Negative	Positive	Negative
ImOnlyMeWhenImWithYou	Positive	Positive	Positive
Invisible	Positive	Positive	Positive
MarysSong_OhMyMyMy_	Positive	Positive	Positive
OurSong	Positive	Negative	Positive
PicturetoBurn	Negative	Negative	Negative
ShouldveSaidNo	Negative	Positive	Negative
StayBeautiful	Positive	Positive	Positive
TaylorSwift_LinerNotes_	Positive	Positive	Positive
TeardropsOnMyGuitar	Positive	Positive	Positive
TeardropsonMyGuitar_PopVersion_	Positive	Positive	Positive
TheOutside	Negative	Positive	Negative
TiedTogetherwithaSmile	Negative	Positive	Negative
TimMcGraw	Positive	Positive	Positive

Unfortunately, the NLTK with negation handling didn't provide us with results we wanted. All the predictions are the same as they were using regular NLTK. Reasons for this include....

In [320…

```
#Import the libraries for preprocessing, stemming, lemmatization, removing english stopwords & punctuations
from pathlib import Path
import string
import nltk

nltk.download('wordnet')
nltk.download('stopwords')
nltk.download('punkt')
```

```
from nltk.corpus import stopwords
from nltk.corpus import wordnet
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.stem.porter import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer

porter=PorterStemmer()
wordnet_lemmatizer = WordNetLemmatizer()

stop_words = stopwords.words('english')

import string
exclude = set(string.punctuation)

#Create a single function to preprocess
def clean(doc):

    doc = doc.lower()

    doc_list = doc.split()

    # remove anything that's not an apostrophe so it doesn't interfere with stopword removal (ie words in c
    punc_free_1 = remove_punc_1(doc_list)
    stop_free = remove_stopwords(punc_free_1)
    punc_free_2 = remove_punc_2(stop_free)
    normalized = normalize(punc_free_2)

    return normalized

def remove_punc_1(doc):
    return_list = []

    for word in doc:
        return_list.append("".join(ch for ch in word if ch not in exclude or ch=="'"))

    return return_list

def remove_stopwords(doc):
    return [word for word in doc if word not in stop_words]

def remove_punc_2(doc):
```

```
    return_list = []

    for word in doc:
        return_list.append("".join(ch for ch in word if ch not in exclude))

    return return_list

def normalize(doc):
      return [wordnet_lemmatizer.lemmatize(word) for word in doc]

doc_clean = [clean(line) for line in preprocessed_df['Lyrics']]
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/neetidesai/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/neetidesai/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/neetidesai/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

First, let's try using gensim for Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA), two NLP techniques

We learned about these techniques and found some implementation examples here: https://www.datacamp.com/tutorial/what-is-topic-modeling

In [321…

```
#Import the Gensim libraries
import gensim
from gensim import corpora
dictionary = corpora.Dictionary(doc_clean)

#Create the corpora matrix
corpus = [dictionary.doc2bow(doc) for doc in doc_clean]
#Initiate the model
Lda = gensim.models.ldamodel.LdaModel

num_topics = 6

#Fit the model with corpora matrix with dictionary for 20 topics
```

```
ldamodel = Lda(corpus, num_topics=num_topics, id2word = dictionary, passes=50)

#Check the results for 20 topics
for idx, topic in ldamodel.print_topics(-1):
    print('Topic: {} Word: {}\n'.format(idx, topic))
```

```
Topic: 0 Word: 0.034*"like" + 0.015*"oh" + 0.014*"im" + 0.014*"think" + 0.012*"time" + 0.011*"got" + 0.010
*"shake" + 0.010*"never" + 0.010*"stay" + 0.009*"ohoh"

Topic: 1 Word: 0.021*"oh" + 0.020*"never" + 0.019*"like" + 0.018*"love" + 0.012*"im" + 0.011*"id" + 0.011
*"one" + 0.011*"wish" + 0.011*"back" + 0.010*"ever"

Topic: 2 Word: 0.021*"im" + 0.018*"oh" + 0.015*"like" + 0.015*"love" + 0.015*"back" + 0.013*"time" + 0.012
*"said" + 0.011*"cause" + 0.011*"say" + 0.010*"baby"

Topic: 3 Word: 0.034*"know" + 0.021*"like" + 0.020*"im" + 0.011*"wanna" + 0.010*"oh" + 0.010*"yeah" + 0.009
*"want" + 0.009*"cause" + 0.009*"one" + 0.009*"got"

Topic: 4 Word: 0.015*"know" + 0.013*"like" + 0.013*"get" + 0.012*"still" + 0.012*"red" + 0.012*"he" + 0.012
*"im" + 0.009*"cause" + 0.008*"never" + 0.008*"oh"

Topic: 5 Word: 0.016*"never" + 0.014*"like" + 0.013*"see" + 0.013*"im" + 0.012*"new" + 0.010*"daylight" +
0.010*"thing" + 0.009*"go" + 0.009*"york" + 0.009*"could"
```

In [322…

```
from gensim.models import LsiModel

# LSA model
lsamodel = LsiModel(corpus, num_topics=6, id2word = dictionary)

#Check the results for 20 topics
for idx, topic in lsamodel.print_topics(-1):
    print('Topic: {} Word: {}\n'.format(idx, topic))
```

```
Topic: 0 Word: -0.407*"like" + -0.298*"know" + -0.297*"im" + -0.291*"oh" + -0.188*"love" + -0.185*"never" +
-0.165*"time" + -0.143*"cause" + -0.131*"one" + -0.129*"back"

Topic: 1 Word: -0.849*"shake" + -0.242*"gonna" + -0.195*"fake" + -0.173*"hate" + -0.163*"break" + -0.162*"p
lay" + -0.103*"thats" + -0.087*"mmmmmm" + 0.085*"know" + -0.083*"im"

Topic: 2 Word: -0.721*"yet" + -0.427*"wood" + -0.417*"clear" + -0.158*"oh" + 0.152*"know" + -0.125*"good" +
-0.095*"remember" + -0.080*"" + -0.066*"looking" + 0.058*"like"

Topic: 3 Word: -0.680*"oh" + 0.360*"know" + -0.246*"ohoh" + -0.239*"love" + 0.195*"like" + 0.192*"yet" + -
0.155*"thats" + 0.111*"wood" + 0.104*"clear" + 0.100*"shake"

Topic: 4 Word: 0.735*"wish" + 0.353*"would" + -0.248*"know" + 0.214*"never" + 0.163*"right" + 0.157*"back"
+ -0.102*"im" + 0.097*"knew" + 0.096*"id" + -0.096*"oh"

Topic: 5 Word: -0.639*"know" + 0.264*"like" + -0.250*"better" + 0.228*"time" + -0.202*"oh" + 0.199*"call" +
-0.182*"wish" + 0.154*"im" + 0.137*"want" + 0.129*"never"
```

In [323…

```
from gensim.models.coherencemodel import CoherenceModel
Lda = gensim.models.ldamodel.LdaModel

def compute_coherence_values(dictionary, doc_term_matrix, doc_clean, stop, start=2, step=3, model=1):
    """
    Input   : dictionary : Gensim dictionary
              corpus : Gensim corpus
              texts : List of input texts
              stop : Max num of topics
    purpose : Compute c_v coherence for various number of topics
    Output  : model_list : List of topic models
              coherence_values : Coherence values corresponding to the model with respective number of topi
    """
    coherence_values = []
    model_list = []
    for num_topics in range(start, stop, step):
        # generate LSA model
        if model == 1:
            trained_model = LsiModel(doc_term_matrix, num_topics=num_topics, id2word = dictionary)  # trai
        elif model == 2:
            trained_model = Lda(corpus, num_topics=num_topics, id2word = dictionary, passes=50)

        model_list.append(trained_model)
```

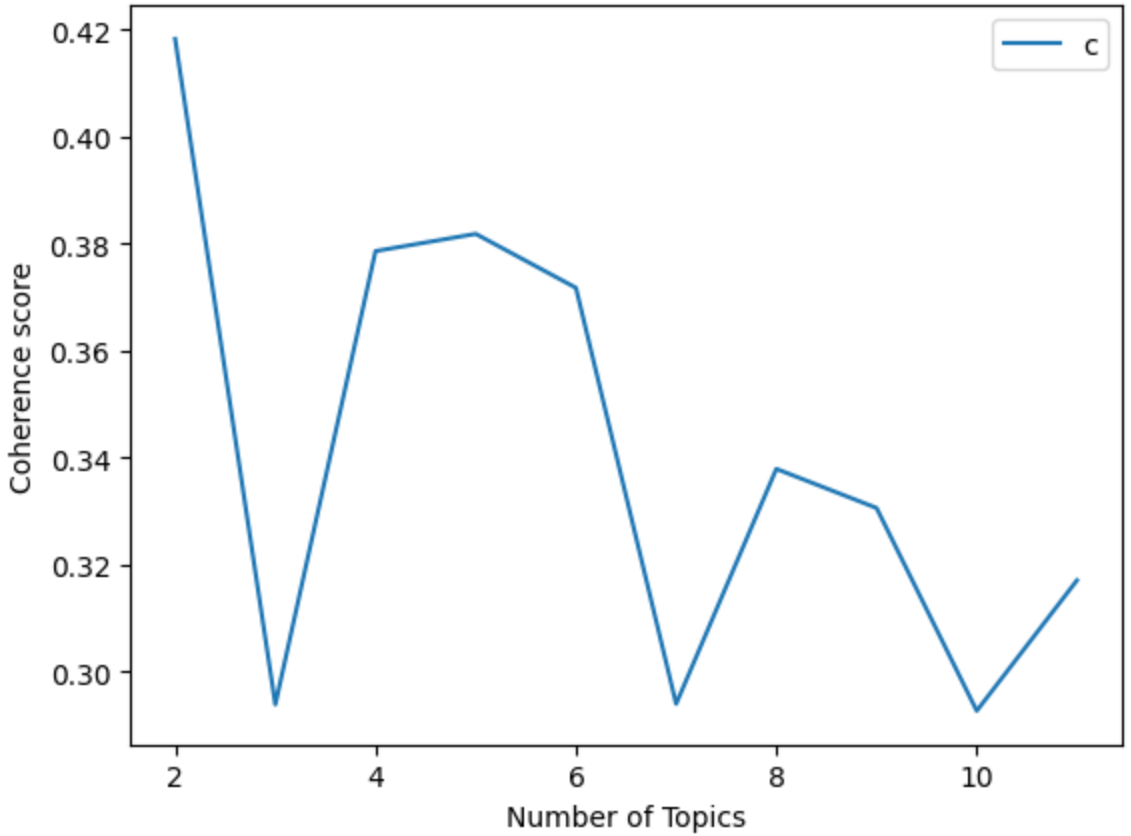
```
        coherencemodel = CoherenceModel(model=trained_model, texts=doc_clean, dictionary=dictionary, cohere
        coherence_values.append(coherencemodel.get_coherence())
    return model_list, coherence_values
```

Let's compare the coherence of gensim LDA and LSA with different numbers of topics

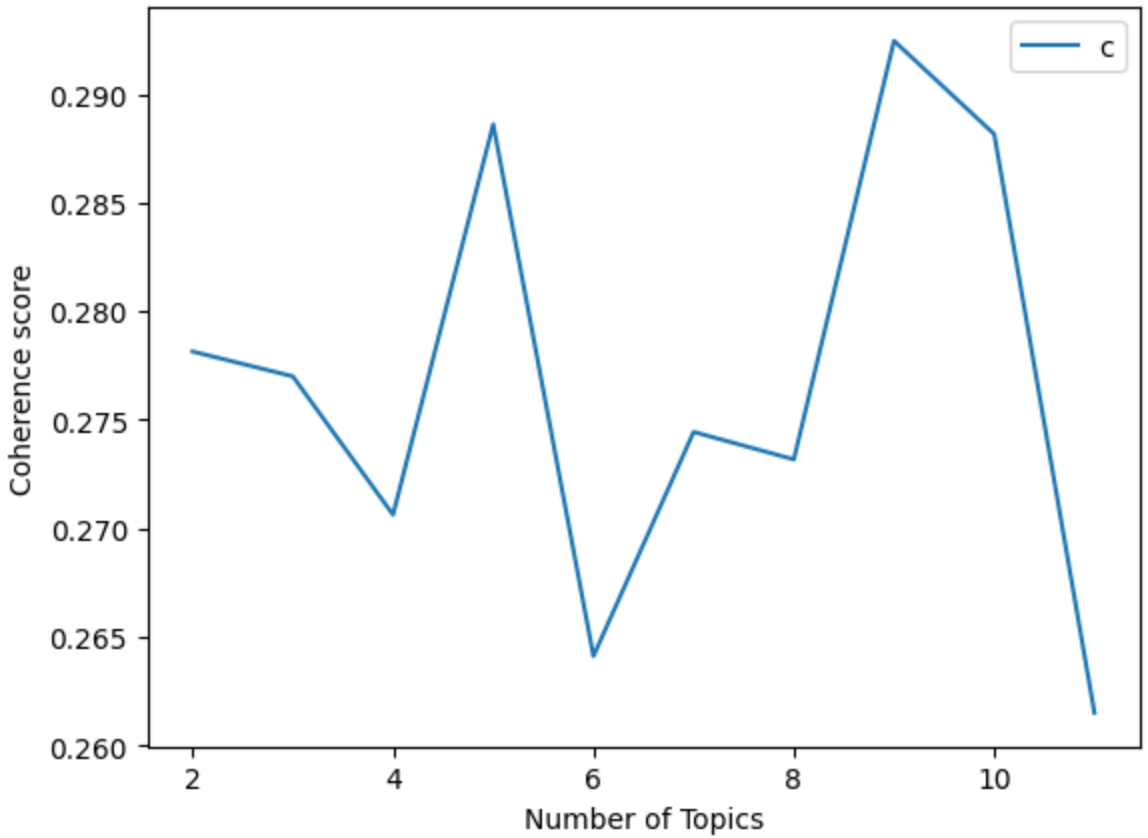
In [324…

```
def plot_graph(doc_clean,start, stop, step, model):
    dictionary = corpora.Dictionary(doc_clean)
    doc_term_matrix = [dictionary.doc2bow(doc) for doc in doc_clean]

    model_list, coherence_values = compute_coherence_values(dictionary, doc_term_matrix,doc_clean,
                                                            stop, start, step, model)
    # Show graph
    x = range(start, stop, step)
    plt.plot(x, coherence_values)
    plt.xlabel("Number of Topics")
    plt.ylabel("Coherence score")
    plt.legend(("coherence_values"), loc='best')
    plt.show()

start,stop,step=2,12,1
print("LSA Model")
plot_graph(doc_clean,start,stop,step, 1)
print("LDA Model")
plot_graph(doc_clean,start,stop,step, 2)
```

```
LSA Model
```


c
0.42
0.40
0.38
0.36
0.34
0.32
0.30
Coherence score
2
4
6
8
10
Number of Topics

```
LDA Model
```



We see that the LSA model has higher coherence values around 0.46 as compared to LDA's score of 0.31. In LSA, it appears that 5 topics are optimal, while LDA has the highest score for a model with 3 topics.

https://towardsdatascience.com/evaluate-topic-model-in-python-latent-dirichlet-allocation-lda-7d57484bb5d0 good resource for qualitative testing of out-of-the-box NLP models

In [ ]:
```
#Import the Gensim libraries
import gensim
from gensim import corpora
from gensim.models import CoherenceModel
import numpy as np
import tqdm
import warnings
warnings.filterwarnings('ignore')
```

```
# create dictionary
dictionary = corpora.Dictionary(doc_clean)
# create corpus
corpus = [dictionary.doc2bow(doc) for doc in doc_clean]

def calculate_lda_coherence(corpus, original_text, dictionary, k, a, b):

    lda_model = gensim.models.LdaMulticore(corpus=corpus,
                                           id2word=dictionary,
                                           num_topics=k,
                                           random_state=100,
                                           chunksize=100,
                                           passes=10,
                                           alpha=a,
                                           eta=b)

    coherence_model_lda = CoherenceModel(model=lda_model, texts=original_text, dictionary=dictionary, cohe

    return coherence_model_lda.get_coherence()

def calculate_lsa_coherence(corpus, original_text, dictionary, k, p, d):

    lsa_model = LsiModel(corpus=corpus,
                                         id2word=dictionary,
                                         num_topics=k,
                                         chunksize=100,
                                         power_iters=p,
                                         decay=d)

    coherence_model_lsa = CoherenceModel(model=lsa_model, texts=original_text, dictionary=dictionary, cohe

    return coherence_model_lsa.get_coherence()


# Topics range
min_topics = 3
max_topics = 10
step_size = 1
topics_range = range(min_topics, max_topics, step_size)

# Validation sets
```

```
num_docs = len(corpus)
corpus_sets = [gensim.utils.ClippedCorpus(corpus, int(num_docs*0.75)),
            corpus]

corpus_title = ['75% Corpus', '100% Corpus']

# Can take a long time to run
if 1 == 2:
    tuned_grid_lda = {}
    tuned_grid_lda['Validation_Set'] = {}

    # Alpha parameters
    alpha = ['symmetric', 'asymmetric', 0.1, 0.3, 0.5, 0.7, 0.9]

    # Beta parameters
    beta = ['symmetric', 0.1, 0.3, 0.5, 0.7, 0.9]

    model_results_lda = {'Validation_Set': [],
                    'Topics': [],
                    'Alpha': [],
                    'Beta': [],
                    'Coherence': []
                    }

    pbar = tqdm.tqdm(total=(len(beta)*len(alpha)*len(topics_range)*len(corpus_title)))

    # iterate through validation corpuses
    for i in range(len(corpus_sets)):
        # iterate through number of topics
        for k in topics_range:
            # iterate through alpha values
            for a in alpha:
                # iterare through beta values
                for b in beta:
                    print(a, k, b)
                    # get the coherence score for the given parameters
                    cv = calculate_lda_coherence(corpus=corpus_sets[i], original_text=doc_clean, dictionary
                                                 k=k, a=a, b=b)
                    # Save the model results
                    model_results_lda['Validation_Set'].append(corpus_title[i])
                    model_results_lda['Topics'].append(k)
                    model_results_lda['Alpha'].append(a)
```

```
                    model_results_lda['Beta'].append(b)
                    model_results_lda['Coherence'].append(cv)

                    print('coherence: ', cv)
                    pbar.update(1)

    pd.DataFrame(model_results_lda).to_csv('lda_tuning_results.csv', index=False)
    pbar.close()

print('done with lda')

# Can take a long time to run
if 1 == 1:
    tuned_grid_lsa = {}
    tuned_grid_lsa['Validation_Set'] = {}

    # power iter parameters
    power_iters = [0, 1, 4, 6, 8, 10]

    # decay parameters
    decay = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]

    model_results_lsa = {'Validation_Set': [],
                    'Topics': [],
                    'Power_iters': [],
                    'Decay': [],
                    'Coherence': []
                    }

    pbar = tqdm.tqdm(total=(len(beta)*len(alpha)*len(topics_range)*len(corpus_title)))

    # iterate through validation corpuses
    for i in range(len(corpus_sets)):
        # iterate through number of topics
        for k in topics_range:
            # iterate through alpha values
            for p in power_iters:
                # iterare through beta values
                for d in decay:

                    print(p, k, d)
                    # get the coherence score for the given parameters
```

```
                    cv = calculate_lsa_coherence(corpus=corpus_sets[i], original_text=doc_clean, dictionary
                                                 k=k, p=p, d=d)
                    # Save the model results
                    model_results_lsa['Validation_Set'].append(corpus_title[i])
                    model_results_lsa['Topics'].append(k)
                    model_results_lsa['Power_iters'].append(p)
                    model_results_lsa['Decay'].append(d)
                    model_results_lsa['Coherence'].append(cv)

                    print('coherence: ', cv)
                    pbar.update(1)

    pd.DataFrame(model_results_lsa).to_csv('lsa_tuning_results.csv', index=False)
    pbar.close()
```

In [325…

```
# optimal hyperparameters determined by our analysis, found from the .csv files of hyperparameter
# combinations and coherence score results:
lda_alpha = 0.5
lda_eta = 0.9

lsa_power_iters = 1
lsa_decay = 0
```

In [326…

```
def display_topics(model, no_top_words):
    topics = []
    for topic_idx in range(model.num_topics):
        topic = model.show_topic(topic_idx, topn=no_top_words)
        topic_words = ", ".join([word for word, _ in topic])
        topics.append(topic_words)

    return topics
```

In [327…

```
# step 1: LDA and LSA

# Run LDA and LSA using optimal hyperparameters determined above
lda = gensim.models.LdaMulticore(corpus=corpus,
                                 id2word=dictionary,
                                 num_topics=8,
                                 random_state=100,
                                 chunksize=75,
                                 passes=10,
```

```
                                        alpha=lda_alpha,
                                        eta=lda_eta)

lsa = LsiModel(corpus=corpus,
                    id2word=dictionary,
                    num_topics=3,
                    chunksize=75,
                    power_iters=lsa_power_iters,
                    decay=lsa_decay)

no_top_words = 10
lda_topics_1 = display_topics(lda, no_top_words)
lsa_topics_1 = display_topics(lsa, no_top_words)

print(lda_topics_1)
print(lsa_topics_1)
```

```
['im, time, like, oh, see, love, know, never, one, tell', 'know, like, love, oh, im, back, never, go, caus
e, wish', 'like, made, call, wanna, look, im, want, time, one, say', 'shake, like, im, stay, come, thats, r
ed, think, oh, baby', 'oh, think, new, like, ohoh, rain, thats, york, welcome, mr', 'like, im, run, oh, ye
t, ill, time, cause, say, remember', 'happiness, evermore, catching, behind, pain, cant, hurt, feeling, lea
ve, thing', 'like, never, im, know, ohoh, love, cause, everybody, one, girl']
['know, like, im, time, never, better, oh, last, back, cause', 'know, time, better, im, last, asking, oh, n
ever, wanna, trouble', 'time, last, asking, never, im, trouble, like, eye, know, oh']
```

Let's experiment with additional stopwords

In [328…

```
stop_words = stopwords.words('english')
stop_words.extend(['oh', 'ooh', 'ah', 'like', "i'm", "that's", "i'd", "us", "we", "i've", "ya", "'til", "ol

filtered_tokens = [clean(line) for line in preprocessed_df['Lyrics']]

# Now create the dictionary
stopword_dictionary = corpora.Dictionary(filtered_tokens)
stopword_corpus = [stopword_dictionary.doc2bow(doc) for doc in filtered_tokens]
```

In [329…

```
# step 2: LDA and LSA with additional stopwords
lda = gensim.models.LdaMulticore(corpus=stopword_corpus,
                                        id2word=stopword_dictionary,
                                        num_topics=8,
                                        random_state=100,
                                        chunksize=75,
```

```
                                        passes=10,
                                        alpha=lda_alpha,
                                        eta=lda_eta)

lsa = LsiModel(corpus=stopword_corpus,
                     id2word=stopword_dictionary,
                     num_topics=3,
                     chunksize=75,
                     power_iters=lsa_power_iters,
                     decay=lsa_decay)


no_top_words = 10
lda_topics_2 = display_topics(lda, no_top_words=no_top_words)
lsa_topics_2 = display_topics(lsa, no_top_words=no_top_words)

print(lda_topics_2)
print(lsa_topics_2)
```

```
['wish, would, say, never, time, love, back, take, could, man', 'shake, red, never, gonna, street, hate, fa
ke, break, hit, play', 'new, could, one, said, love, cant, rain, never, think, waiting', 'never, want, yet,
call, look, made, he, time, wood, one', 'run, girl, man, mr, look, get, know, everybody, say, ever', 'neve
r, go, time, back, one, say, wanna, see, baby, got', 'love, time, karma, still, daylight, snow, see, bette
r, last, might', 'think, bet, get, want, cant, got, bless, know, hope, aint']
['time, never, last, back, asking, go, eye, love, ever, remember', 'time, last, asking, never, trouble, ey
e, ever, grow, remember, could', 'trouble, knew, never, walked, flew, shame, place, remember, could, love']
```

In [304…

```
from itertools import zip_longest
from prettytable import PrettyTable

# make a table to compare topics before and after stop word customization
table_lda = PrettyTable()
table_lda.field_names = ["Topic Number", "LDA (no stopword modification)", "LDA (stopword modification)"]

max_topics = max(len(lda_topics_1), len(lda_topics_2))

# Use zip_longest to ensure all topics are considered
for idx, topic in enumerate(zip_longest(lda_topics_1, lda_topics_2, fillvalue='')):
    row = [f"Topic {idx + 1}"]
    row.extend(topic)
    table_lda.add_row(row)
```

```
table_lsa = PrettyTable()
table_lsa.field_names = ["Topic Number", "LSA (no stopword modification)", "LSA (stopword modification)"]

max_topics = max(len(lsa_topics_1), len(lsa_topics_2))

# Use zip_longest to ensure all topics are considered
for idx, topic in enumerate(zip_longest(lsa_topics_1, lsa_topics_2, fillvalue='')):
    row = [f"Topic {idx + 1}"]
    row.extend(topic)
    table_lsa.add_row(row)

print(table_lda)
print(table_lsa)
```

Topic Number	LDA (no stopword modification)	LDA (stopword modification)
Topic 1	im, time, like, oh, see, love, know, never, one, tell	wish, would, say, never, time, love, back, take, could, man
Topic 2	know, like, love, oh, im, back, never, go, cause, wish	shake, red, never, gonna, street, hate, fake, break, hit, play
Topic 3	like, made, call, wanna, look, im, want, time, one, say	new, could, one, said, love, cant, rain, never, think, waiting
Topic 4	shake, like, im, stay, come, thats, red, think, oh, baby	never, want, yet, call, look, made, he, time, wood, one
Topic 5	oh, think, new, like, ohoh, rain, thats, york, welcome, mr	run, girl, man, mr, look, get, know, everybody, say, ever
Topic 6	like, im, run, oh, yet, ill, time, cause, say, remember	never, go, time, back, one, say, wanna, see, baby, got
Topic 7	happiness, evermore, catching, behind, pain, cant, hurt, feeling, leave, thing	love, time, karma, still, daylight, snow, see, better, last, might
Topic 8	like, never, im, know, ohoh, love, cause, everybody, one, girl	think, bet, get, want, cant, got, bless, know, hope, aint

Topic Number	LSA (no stopword modification)	LSA (stopword modification)
Topic 1	know, like, im, time, never, better, oh, last, back, cause	time, never, last, back, asking, go, eye, love, ever, remember
Topic 2	know, time, better, im, last, asking, oh, never, wanna, trouble	time, last, asking, never, trouble, eye, ever, grow, remember, could
Topic 3	time, last, asking, never, im, trouble, like, eye, know, oh	trouble, knew, never, walked, flew, shame, place, remember, could, love