Stephen Hullender
CIS 3715 001
April 24, 2022

Predicting and Identifying Genre of Song Based on Lyrical Content: A Final Report

Introduction

Music is integral in our daily lives, and the composure of a song's lyrical content is an important attribute to consider when listening to music. Lyrics can influence our choice of music, and impact the song's significant meaning, as well as influence the reputability of a genre. When looking at the words of a song, it can be interesting to consider these factors and figure out what words define different types of mainstream music. What distinguishes rock music from pop, or rock from metal? How about gospel music to romance?

This project seeks to understand how certain vocabulary influences genres, such as measuring what words are common among different genres and/or artists. The goal of the project is to accumulate the lyrical content of a variety of popular songs and predict the genre of any given sample of lyrics provided. The worksheet will also look at the most common words for all genres provided, assuming that the genres meet the criteria of having a feasible number of songs. For this project, I was the sole contributor of the project, responsible for all implementation of libraries and Python code in these worksheets, and am the sole author of the report provided.

Approach

The main approaches that will be followed include using K-nearest neighbors to determine how many nearest neighbors are possible, logistic regression to calculate the accuracy and F-1 scores, and K-means classifier to distribute the top ten most common words across clusters representative of genres. The purpose of using KNN is to read accuracy of the data provided and to read the best parameters (i.e. nearest neighbors). Logistic regression was initially chosen as the project was intended on researching more into other variables such as artist and popularity score, two factors that may or may not have influenced how lyrics would be predicted. Lastly, KMeans was used to arrange a cluster of words that most resemble the given features, or genres.

To begin the project, we are introduced to two datasets: the artists dataset, and the lyrics dataset. These datasets were downloaded from Kaggle as 'Song lyrics from 79 musical genres'[1]. The artists dataset consists of the following columns; "Artist", "Genres", "Songs", "Popularity", and "Link". The lyrics dataset contains columns: "Artist Link", "Song Name", "Song Link", "Lyric", and "language". The common identifier is the artist link, "Link" and "ALink" for artist data and lyrical data, respectively.

To reduce the amount of clutter from the given information, we use Exploratory Data Analysis to clean out unwanted data. First, we delete unnecessary columns such as artist name, number of songs, and popularity score of the artist. The lyrical column is deleted because the lyrics provided are only a sample and don't represent the song's entirety, but will be replaced with a full string of lyrics eventually. Second, we halve our data by eliminating non-English entries, indicated by the "language" column in the lyrics

data; if an artist has no more songs as a result, they are eliminated from the artist data. Next, we iterate through each genre item in the "Genre" column, and eliminate any entries with certain genres. Some genres are removed for the following reasons:

- not enough entries
- redundant genres that can be perfectly matched with other existing, and bigger, genres
- a reasonable assumption that the majority of songs within these genres are non-English (even if non-English entries are already cut out of the dataset)

Afterwards, we cleared out the Genre column and replaced it with two columns for the first and second (if applicable) genre. The reason why I chose two choices for each song is to analyze the results for the genres provided (i.e. "is the genre provided accurate? How about the second genre choice?"). By this point, we have cut down from almost 380,000 songs to around 184,300 songs. In all that we have processed, the end result will only contain about 10,000 songs; this is mainly due to a conflict between time and the large amounts of data being processed, as well as filtering out bad entries from the datasets.

After we are done getting the songs we want, we start to get the lyrical data we need. All web scraping and data filtering is to be done through the Python Beautiful Soup library, which pulls up the HTML code for the web page containing a song's lyric; the link to this page is provided through the "Song Link" column of the lyrics dataset. The HTTP requests are also performed by another Python library called urllib. Another major library that will be used to clean up words is to use the Natural Language Toolkit (nltk) to remove stopwords. Stopwords are words that take up space, such as "I", "you", "and", "a(n)", etc.

One other approach in relation to cleaning text is using regular expressions to clear out most punctuation symbols, with the exception of the single quote, for words like "i'll" or "we've". With that, there is an important issue that should be brought to attention when handling the datasets: all of our lyrical content is to be extracted using web scraping tools. We will be pulling data from a lyrics website called Vagalume.com (as the author of the dataset got their data from this source). It is possible that the format of the lyrics received will vary, and will require some discretion even after cleaning up unneeded characters. This also means that it is possible that the lyrical data may not accurately predict the genres due to garbage text that can only be caught by chance.

Due to time constraints, and lack of knowledge of parallel processing, despite attempts to connect relevant libraries to expedite the process, I was left with little choice but to iterate through each entry to get the lyrical content of the song, as well as summating the number of instances of each word based on genre (this is to compare what words are used most in a given genre). Because of my naive proposition, the maximum that I allowed is around half, or 95,000 songs, to be processed into our next step. To save the data and save time, I came up with the idea of printing it to a .csv file and running the project on two worksheets. An aforementioned, and last minute, decision was to pick a randomized 10,000 songs instead.

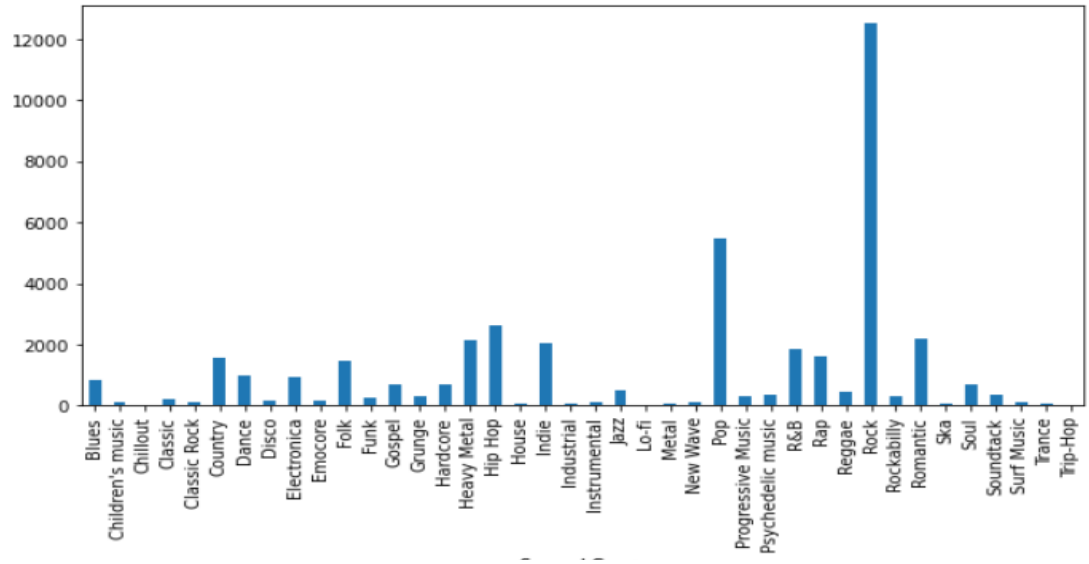
Since the project involves extracting text, one approach is to use the Bag-of-Words model, which includes the use of TF-IDF, or term frequency-inverse document frequency. This model measures how relevant a word is on the condition of its presence among lyrics from several songs. The reason why we also use logistic regression to begin making our predictions is because logistic regression is easy to

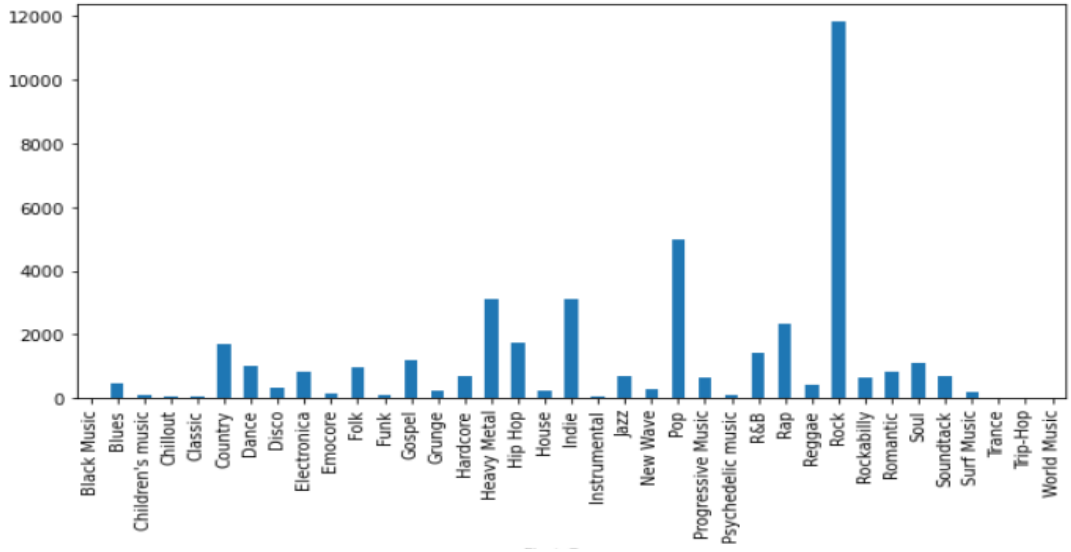
implement and fits with the data that we want to classify, such that our data contains 0's and 1's for every word entry provided per document (binary classification), as well as data involving the frequency of each word using the aforementioned Bag-of-words approach. In the case of clustering words, we decided that K-Means would be a great fit for arranging the words for each genre.

Results

Based on the data given for all words captured for all genres, we have the following visualizations, giving us the top 10 words from each major genre:

First genre choice results (top), second genre choice results (bottom)





The results indicate an obvious skew, which is that the majority of songs belong in a select few gernes, noticeably 'Rock', 'Pop', along with secondary genres such as 'Heavy Metal' and 'Indie'.

The reason why I chose to look through two sets of genres is based on the multitude of genres that the majority of the songs given provided. Because processing the data with more than one genre at once would probably complicate results, it was easier to iterate through both TF-IDF vectorization and logistic regression using each set of genres each time. Furthermore, two columns were used as an assumed factor that the majority of songs would average out to about 2 genres per artist (i.e. the distribution of genres per artist would be equal among artists that have 1, 2, or 3 genres, but no more than that).

Initially when performing TF-IDF and logistic regression, the implementation of the models were not clear, mostly from lack of knowledge on how to use certain sklearn libraries. Because of this, the results given for the first implementation show very different, and bad, results compared to the final results. For the next few paragraphs, I will be discussing the results made based on these implementations:

Rough comparison of genre predictions vs. actual genre names

	Real	Predictions	Correct?
0	Hip Hop	Pop	False
1	Rock	Romantic	False
2	House	Pop	False
3	Indie	Romantic	False
4	Gothic Rock	Folk	False
...	...	...	...
12543	Heavy Metal	Heavy Metal	True
12544	R&B	Country	False
12545	Pop	Pop	True
12546	Rock	Progressive Music	False
12547	Soul	Romantic	False

	Real	Predictions	Correct?
0	R&B	R&B	True
1	Rock	Romantic	False
2	Electronica	Hip Hop	False
3	Surf Music	Romantic	False
4	Gothic Rock	Country	False
...	...	...	...
12543	Hard Rock	Progressive Music	False
12544	Hip Hop	Hip Hop	True
12545	Pop	R&B	False
12546	Romantic	Progressive Music	False
12547	R&B	Romantic	False

The two results were an attempt at comparing predicted genres to actual song genres based on first and second genre choice, respectively. The results show that the majority of predictions point to an incorrect result. Remarkably, despite only making up only a small percentage of the total songs, “Rap” appears in a significant number of predictions, even if the genres do not match whatsoever.

Comparison of real vs. predicted genres for first and second genre choice

	Real	Predictions	Correct?
count	12548	12548	12548
unique	42	34	2
top	Rock	Rap	False
freq	2126	4567	10115

	Real	Predictions	Correct?
count	12548	12548	12548
unique	47	34	2
top	Rock	Rap	False
freq	2266	4250	10499

Accuracy, F-1 (micro), recall, precision scores for first and second genre choice

```
accuracy is 0.13232023721275019,
 f1 score is 0.13232023721275019,
 recall is 0.13232023721275019,
 precision is 0.13232023721275019
```

```
accuracy is 0.09673832468495182,
 f1 score is 0.09673832468495182,
 recall is 0.09673832468495182,
 precision is 0.09673832468495182
```

One observation, from using various sample sizes for TF-IDF and logistic regression, is that the bigger the size of the processed dataset, the better the results are. Between the accuracy, recall, precision and F1 scores, all shared about 0.1 when using around 90 songs. This increased to around 0.13 when the final processing returned around 8000 songs. When looking at the results from a bird's eye view, the overall scores are lacking, possibly due to incorrect implementation of logistic regression and/or TF-IDF. The formulas for TF-IDF were created manually, before implementing the TF-IDF vectorizer from sklearn, and therefore took longer than expected.

One of the reasons why the logistic regression most likely did not work as expected is because the values that we want to compare are text-based. Unlike other lab assignments where regression models were practiced using binary values for data (0's or 1's), or even numerical data, we were dealing with text. Consequently, this worksheet was scrapped and a new implementation of TF-IDF and Logistic Regression was created.

After revamping the models, I started by adding the TF-IDF vectorizer to the solution; the vectorizer converts a list of text into assorted numerical values accordingly by using the TF-IDF equation. The Logistic Regression model differs in that the original implementation excluded GridSearchCV, and attempted to calculate all metrics, including precision and recall, which would always return the same, low, result as the accuracy and F-1 score. GridSearchCV, part of the model selection library from sklearn, is used to parametrize data using a cross-validation method; this was used in both the K-neighbors classifier and logistic regression models to consider different combinations of data when passing it through their respective models.

Best scores from GridSearchCV, using KNearestClassifier, for first & second genre choice

```
0.18073170731707316
{'n_neighbors': 4}
Wall time: 10.8 s
```

```
0.175
{'n_neighbors': 4}
Wall time: 11.3 s
```

Accuracy and F-1 (macro & micro) scores, logistic regression, first & second genre choice

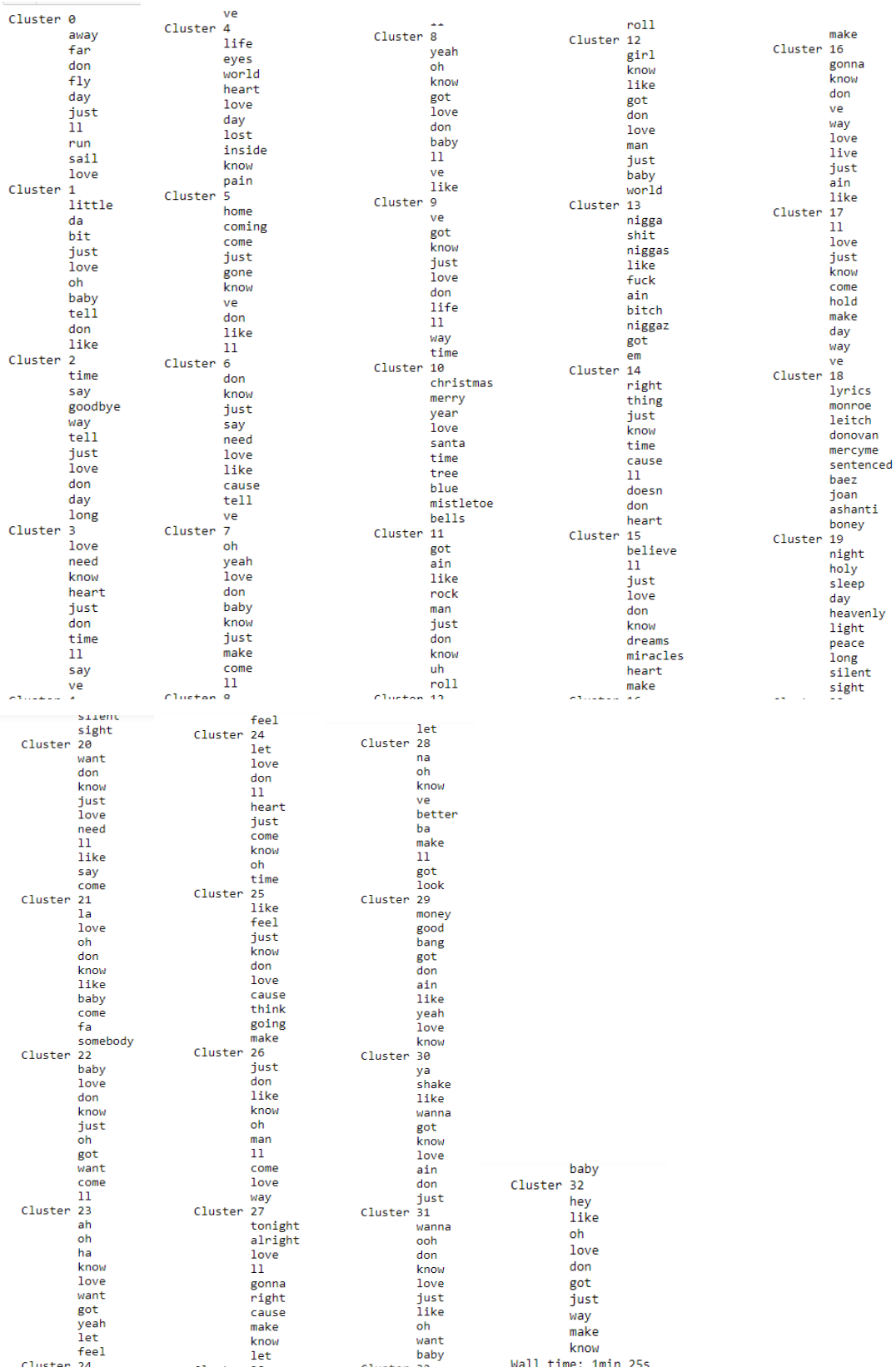
```
{'C': 3}
0.35555555555555557 0.11614313256404439 0.35555555555555557
Wall time: 39min 44s
```

```
{'C': 2}
0.3522222222222222 0.0679471926576408 0.3522222222222222
Wall time: 50min 6s
```

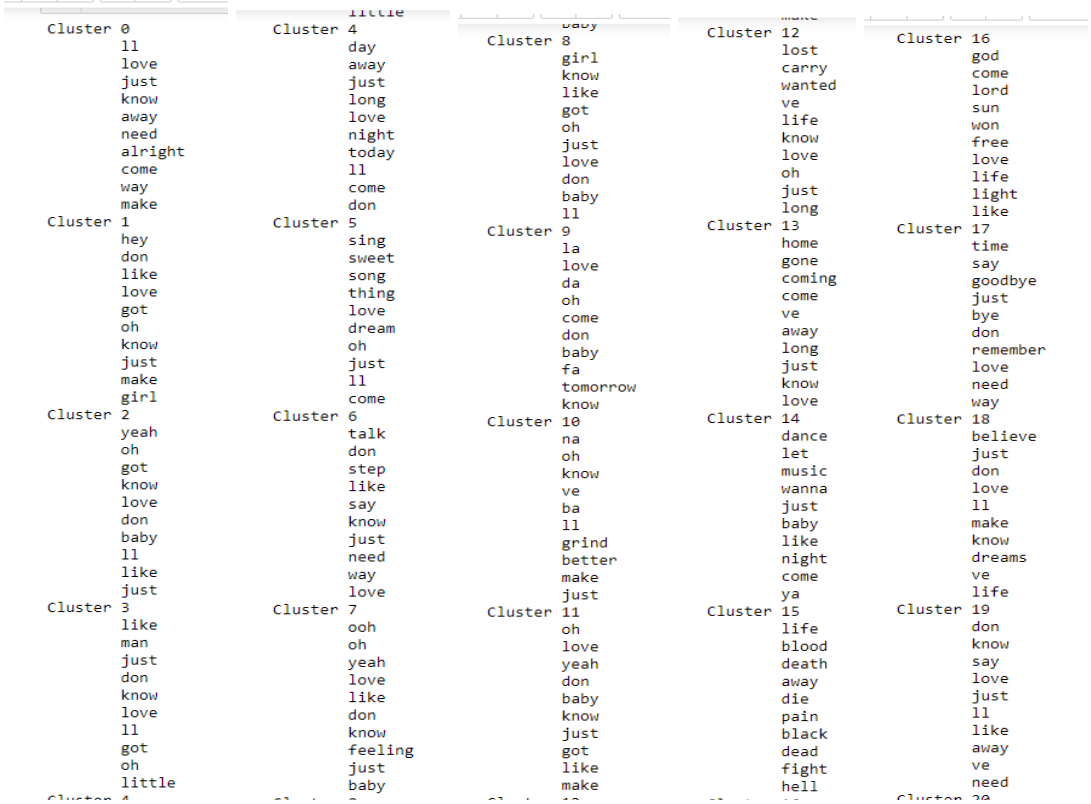
The accuracy scores show that there are some possible hindrances with getting the best data with the lyrics provided. Possible errors may be the result of bad web scraping or words that have made it past filtering for stopwords and/or punctuation. Overall, the accuracy and F-1 scores improved much more compared to the first implementation of the TF-IDF and logistic regression models. As predicted, the results also suggest that the micro F-1 score works better than the macro F-1 score; this was originally chosen under the condition that there were multiple variables that would determine the outcome of the predictions made (i.e. measuring more than just binary data).

Finally, we were able to create clusters to fit in the most common words per genre. All clusters were generated using the K-means classifier, and retrieving the cluster centers provided. For every cluster we obtained, we iterated through each center and got the most common words using argsort(), which sorts out the most frequent values. Here are the results provided:

Results for each cluster, first genre choices



Results for each cluster, second genre choices



The results are not the best, and despite efforts to contain stopwords, as well as onomatopoeic sounds like ‘oh’ and ‘lalala’, some words still show up. Due to this being a last-minute approach, I was

unable to identify which genre is which based on the clusters given, although a few were identified using context clues with the words given. For example, it is very likely that Cluster 10 for the first set of genres, and Cluster 23 for the second set, contain holiday or Christmas music; this is based on words such as “christmas”, “merry”, “santa”, etc. Likewise, a genre like “Gospel” is identified from Cluster 19 (first genre choice) and Cluster 31 (second cluster choice) with words such as “jesus”, “holy”, “heavenly”, etc.

Finally, for reference, when extracting lyrical data, I also made the step of storing all words that ever existed (within the constraints of the lyrics dataset) into one, and very heavy, matrix array. Here is a sample of the data provided:

All word count of words to ever exist from all genres and all songs

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD
1		Rock	Indie	Pop	Romantic	Folk	Grunge	Jazz	Classic	New Wa	Rap	Hip Hop	Metal	Heavy M	Country	Hardcore	Rockabil	Blues	R&B	Soul	Dance	Gospel	Funk	Emocore	Punk	Psychec	Classic F	House	Electroni	Progress
2	know	36604	6010	25871	7260	2760	803	861	275	539	11573	14783	47	4340	2779	1928	679	1812	9961	3162	6367	2014	887	566	772	492	420	555	3192	800
3	love	34587	5204	29736	9761	3409	587	1915	581	565	5254	9166	9	3149	4072	829	1628	2278	12316	6252	7866	2524	1438	431	354	877	438	956	3854	875
4	like	29495	5241	22918	5643	2313	595	827	326	459	18904	20845	64	3733	2547	1536	641	1242	9926	2376	6319	1622	941	503	608	472	258	592	3372	697
5	oh	26644	4868	22753	6018	2135	533	560	205	474	4571	6852	14	2448	2037	794	732	1380	7386	3245	6668	1584	850	270	511	505	379	646	3193	496
6	got	23047	3654	15399	3665	1620	551	591	102	299	13387	14792	29	2441	1912	1137	618	1666	7004	2111	4263	705	814	206	405	406	376	399	2293	470
7	time	23001	3143	12420	3719	1759	366	539	191	460	4647	5517	58	4077	1881	1398	495	1015	4302	1850	2767	986	569	403	384	362	266	321	1412	612
8	never	22618	3111	12424	3874	1535	466	565	248	355	4516	5379	55	4034	1599	1504	434	727	3919	1501	2550	1117	365	493	398	212	176	278	1291	578
9	one	22317	3104	13160	3888	1717	415	627	194	364	5631	6848	70	4060	1902	1289	463	934	4731	1833	3224	1258	502	402	313	299	251	404	1676	738
10	go	21632	3438	14655	3835	1839	405	630	183	292	6921	8385	33	2613	1706	1204	507	1048	5511	1804	4129	999	611	394	443	314	253	405	2293	651
11	see	21134	3224	12116	3277	1750	361	505	217	328	6581	7494	55	3799	1440	1259	403	780	4522	1762	2827	1270	477	361	270	318	214	303	1454	766
12	get	20816	3243	15502	3501	1351	442	507	136	273	13470	15261	30	2391	1523	1217	437	1228	6802	1954	4648	714	818	244	353	341	325	464	2689	588
13	cant	19460	2704	12282	3465	1062	500	478	102	305	5391	6098	46	3116	1473	1333	379	940	4637	1670	3066	704	497	359	317	246	207	308	1608	465
14	want	18022	2776	11904	3031	1084	335	352	80	350	4325	5324	29	2009	906	1011	282	739	4157	1471	3448	777	556	210	341	216	187	357	1593	374
15	come	17345	2649	10062	2858	1481	376	537	225	282	4364	5200	55	2817	1418	690	480	986	3801	1552	2381	1134	583	216	237	319	252	349	1419	510
16	take	17235	2254	9919	2708	947	271	297	141	318	4269	5357	36	2987	1197	1147	269	586	3519	1229	2425	881	332	249	243	233	148	224	1287	532
17	yeah	17114	2419	14358	3163	659	459	151	45	128	7941	9540	5	1763	1071	518	418	1188	6112	2460	4397	625	980	143	294	450	286	355	1723	294
18	way	17028	2278	10074	3054	1294	361	409	177	246	3368	4338	37	2647	1600	928	397	851	3545	1422	2524	897	418	237	305	212	229	283	1096	573
19	away	16847	2189	7180	2254	1246	377	361	127	253	1303	1895	37	3183	1111	996	273	520	1705	893	1554	780	183	299	216	190	145	151	937	554
20	say	15778	2386	11303	3307	1195	319	511	158	281	4625	5853	29	1721	1236	850	352	1141	4612	2032	2597	732	574	260	378	238	140	264	1251	461
21	ill	15616	2435	9558	2986	1273	336	587	169	249	2700	3349	26	2199	1459	960	539	716	2959	1209	2303	679	268	285	318	256	97	238	1194	378
22	back	15436	2295	8577	2356	1301	370	254	83	184	6155	6954	53	2209	1531	926	429	719	3161	948	1858	621	218	278	234	171	137	174	1255	366
23	gonna	15156	1972	10039	2527	1041	224	381	107	195	2157	3227	15	1821	1417	606	513	1159	3013	1369	2338	549	409	136	209	259	286	175	996	274
24	let	15026	2134	11728	3197	1009	292	397	169	224	4596	5906	57	2340	1229	916	312	545	4460	1638	3109	1256	321	323	275	235	115	342	1682	399
25	life	14651	1774	7494	2231	855	227	394	125	225	3310	3711	84	3299	909	946	201	503	2161	1150	1673	1241	305	163	238	146	158	259	941	532
26	feel	14499	2331	9658	2571	793	374	251	98	240	2700	3691	51	2385	806	820	213	509	3369	1369	2693	641	376	281	250	190	116	477	1663	409
27	make	14176	1917	10843	2732	949	256	367	112	237	5634	6968	26	1864	959	868	281	561	4669	1712	2802	695	608	257	260	204	175	266	1510	402
28	well	14067	2101	5710	1892	1783	262	447	154	229	1511	1776	11	1581	1691	603	751	1211	1425	921	1038	492	222	233	191	319	206	103	560	426
29	baby	13870	1955	13908	4473	1003	249	493	56	153	4473	7455	14	1022	1634	279	931	2164	6858	3725	4153	329	1263	69	174	313	265	458	1598	140
30	cause	12720	1898	11647	3181	810	228	326	97	157	6699	7676	23	1494	1240	836	261	548	4200	1253	2941	732	335	178	255	148	89	230	1340	156
31	right	12087	1532	7878	2310	784	241	299	89	159	3974	4908	12	1587	1146	708	279	704	3538	1318	2153	461	384	156	195	177	130	285	1071	321
32	world	11870	1297	5892	1666	714	202	184	119	226	1841	2247	71	2479	830	691	219	406	1825	984	1320	953	290	165	169	178	118	145	604	498
33	need	11692	1805	8737	2468	756	236	208	77	188	3243	4419	22	1569	781	532	190	477	3617	1255	2272	741	324	170	219	233	147	231	1171	325
34	could	11682	2057	7445	2287	1132	195	281	89	219	2205	2892	12	1255	1117	549	203	410	2202	824	1677	655	320	203	202	132	93	227	872	436
35	day	11679	1717	5683	1865	1191	179	435	216	200	2188	2566	39	1959	1190	595	335	603	1834	935	1180	789	292	110	172	251	138	92	696	475
36	night	11593	1715	7195	2225	1104	167	423	204	213	1720	2495	26	2138	1148	411	374	651	2316	1062	1947	417	259	164	146	182	240	347	1167	376
37	wanna	11207	1752	10733	2479	534	158	149	40	123	5014	6800	12	1116	522	512	132	431	4680	1440	3350	507	528	82	340	92	125	472	1738	83
38	heart	11134	1675	8178	3115	1029	147	581	202	204	1082	1639	15	1769	1353	457	411	548	2297	1280	1926	1184	251	176	97	176	108	203	906	375
39	tell	10842	1546	7507	2000	855	223	311	130	174	3460	4523	12	1377	894	508	276	625	3052	1008	1903	498	267	147	212	153	132	219	922	367
40	give	9767	1233	6380	1941	567	169	280	66	150	3057	3798	44	1436	643	549	202	407	2686	968	1608	694	357	129	130	134	96	155	1016	281
41	eyes	9690	1472	4892	1594	842	176	245	103	180	1089	1308	39	2062	669	592	140	302	1180	548	1100	546	136	191	124	129	111	119	612	391

Sample of top words per genre, among 5 most common genres

	Rock
know	36604
love	34587
like	29495
oh	26644
got	23047
time	23001
never	22618
one	22317
go	21632

	Pop
love	29736
know	25871
like	22918
oh	22753
get	15502
got	15399
go	14655
yeah	14358
baby	13908

	Romantic	F
love	9761	
know	7260	
oh	6018	
like	5643	
baby	4473	
one	3888	
never	3874	
go	3835	
time	3719	

	Indie	P
know	6010	
like	5241	
love	5204	
oh	4868	
got	3654	
go	3438	
get	3243	
see	3224	
time	3143	

	Folk	C
love	3409	
know	2760	
like	2313	
lyrics	2250	
oh	2135	
go	1839	
well	1783	
time	1759	
see	1750	

The majority of the words provided throughout the worksheet comes from genres that make up a significant share of the dataset, like Rock and Pop music, but there were a couple outliers that stood out, such as Rap. Although this data was not used for the final product, and was realized to be futile despite taking a lot of time and RAM to render, I found it fascinating given the vast array of words available, as well as the frequency in which the words are used among all given genres. Some of the words provided can also be recognized in the clusters generated too.

Conclusion

In conclusion, the results indicate that the song's genre can be difficult to predict based on the lyrical content of the song. By using approaches such as the Bag-of-words model (with TF-IDF), logistic regression, and the K-means classifier, we were able to make a reasonable estimate on how well a song can be matched to its genre based on the types of vocabulary that is used in the composition. The results indicate that the accuracy scores were not as great, but were improved after redesigning the logistic regression and TF-IDF models. Clusters made from the K-means classifier were shown to have given accurate results based on the features of the data provided.

To summarize the biggest problems with this project, they either involved lack of preparation and knowledge using text analysis libraries (TF-IDF), waiting a long time to gather the data, and variations of certain text messing up the results despite attempts to clean the data while extracting the lyrics. A lack of knowledge on text in relation to the logistic regression model also made it challenging to create a functional model for our results, whereas numerical data was handled much more easily in previous works. Handling large datasets was also a challenge given that previous projects were handled with smaller, and much more manageable, data.

Acknowledgements

The idea of this project came about after skimming through numerous articles that explained how music impacts people's mood. One journal written by Christenson et al. argues that lyrics are important because they reflect the consensus of the audience they are targeting. The lyrical content matters most when the listener can resonate with the connotation of the song. There is also some notion of certain genres being assigned a certain image due to stereotypes involving the content of the song's lyrics, such as a controversial case with rock or metal music being correlated with violence. More information about this journal is linked in *References* as an external link.

Although I was inspired to create a project about the linguistics of music, I initially struggled to pick a topic that could relate to the methods of data science. The reason why I chose to analyze the genre of the song is because it is the best factor that can be predicted based on the lyrical content of the song. When brainstorming, there was the consideration of predicting the artist based on the song, but that proved to be more difficult in practice because of the wider arrangement of artists compared to genres.

The flow of the project was influenced by a similar project done by Alda Sianipar, a graduate data science student from UCLA, who had performed similar approaches in creating the project[2]. The only major difference between the inspired project and the project undergone here is that lemmatization was removed; lemmatization is the process of grouping words based on a common root word, like, for instance, "mobile", "mobility", and "mobilize". The reason for skipping over this was because of unfamiliarity with lemmatization.

Assistance from this project was also possible with other articles and sources like GitHub and *Towards Data Science.* One of the biggest motivations for using these sources came from remodeling my approaches when trying to display the best data possible using TF-IDF and logistic regression. The rough

draft implementation of the logistic regression and TF-IDF model was inspired by a GitHub-uploaded IPYNB worksheet from data scientist Idil Ismiguzel, who also wrote an article in relation to how and why she chose logistic regression for text classification[3][4]. The idea of clustering also came about when revisiting the sources provided for Labs 8 and 9, with an article that explains how K-Means clustering works and is demonstrated[5].

References

- [1] Song lyrics from 79 musical genres, *Kaggle*, https://www.kaggle.com/datasets/neisse/scrapped-lyrics-from-6-genres?select=lyrics-data.csv
- [2] Sianipar, Alda. "Predicting a Song's Genre Using Natural Language Processing", *Better Programming*, 24 July 2019, https://betterprogramming.pub/predicting-a-songs-genre-using-natural-language-processing-7b354ed5bd80
- [3] Ismiguzel, Idil. "Applying Text Classification Using Logistic Regression", *Analytics Vidhya*, 7 May 2020, https://medium.com/analytics-vidhya/applying-text-classification-using-logistic-regression-a-comparison-between-bow-and-tf-idf-1f1ed1b83640
- [4] Text Classification on Amazon Find Food Reviews Dataset, *GitHub*, https://github.com/Idilismiguzel/NLP-with-Python/blob/master/Text-Classification.ipynb
- [5] Foley, Daniel. "K-Means Clustering: Making Sense of Text Data using Unsupervised Learning", *Towards Data Science*, 8 Feb 2019, https://towardsdatascience.com/k-means-clustering-8e1e64c1561c
- **EXTERNAL LINK** Christenson et al., "What has America been singing about? Trends in themes in the U.S. top-40 songs: 1960-2010", *Sage Journals*, 23 Jan 2018, https://journals.sagepub.com/doi/full/10.1177/0305735617748205