Project Title: "AI MUSIC RECOMMENDATION AND GENERATION"

Team Name : Byte Spark

Team Members :

Allam Gowri Shankar	22101A010003
Anish Kunda	22101A010006
Damarla Pavan	22101A010045
Vaibhav Tripathi	22101A010185

Under Guidance of

Dr. J. Avanija
Professor
Dept of CSE-AIML
MBU

Head

Dr. B. Narendra Kumar Rao
Prof & Head
Dept of CSE-AIML
MBU

Abstract

AI-based playlist creation and music recommendation system is presented in this work. To find similarities between songs, the algorithm makes use of sensory characteristics and genre classifications obtained from music data. Through an analysis of the user's past listening choices or an existing playlist, the system suggests new music based on shared attributes.

K-Means clustering is used in the recommendation process to classify songs according to their auditory characteristics. The songs in the dataset that are most comparable to the user's preferences are then found using a distance metric. With this method, the system may create playlists that suit the user's musical preferences.

Keywords*:* Music Recommendation System, Music Information Retrieval, K-Means Clustering, Audio Feature Analysis.

Table of Contents

1. Introduction

1.1 Introduction

Human civilization has always been greatly influenced by music. It has the power to arouse a wide range of emotions, emotionally connect us, and transport us to many locations and times. The digital era has changed how we listen to music. We now have access to enormous song libraries at our fingertips because to the growth of music streaming services. This wealth of options, though, may sometimes be debilitating. Finding new music that suits our own tastes can be an arduous and time-consuming process.

Conventional approaches to finding music, including filtering by popularity or genre, might be restrictive. Genre classifications are frequently arbitrary and cover a broad spectrum of tones. Songs' originality or quality may not always be reflected in popularity numbers.

1.2 Statement of the Problem

Information overload has become a problem for music consumers due to the growing amount of music accessible through streaming services. It could be difficult to sift through millions of music to discover new favourites. The recommendation systems that streaming providers currently provide are frequently based on basic algorithms that might not adequately represent each user's unique taste. These systems may prioritize recommending songs that are currently popular or that are similar to other songs by the same artist or genre, which might result in a monotonous listening experience.

The goal of this project is to construct a more advanced AI-based playlist

generation and music recommendation system in order to address this difficulty. The technology will make recommendations for new songs that are likely to connect with users based on their own musical inclinations.

1.3 Objectives

The principal aim of this project is to design a system for proposing music that utilizes artificial intelligence (AI) to generate customized listening experiences. The way the system will do this is by:

Comprehensive Music Analysis

Beyond simple genre categorization, the algorithm will examine more aspects of audio files, including instrumentals, danceability, energy, valence, and tempo. These elements offer a more impartial and advanced view of a song's musical qualities.

Determining Song Similarities

Through the analysis of these audio characteristics, the system will be able to recognize song links and patterns. This will enable the system to create a rich tapestry of musical links by grouping songs that have similar auditory characteristics.

Creating User Profiles

By analysing a user's listening history, which includes the songs they most frequently listen to, the artists they follow, and the playlists they make, the system will be able to determine their musical preferences.

Personalized Recommendations Generation

The system will suggest new songs that are similar to the music the user already appreciates, based on the user's profile and the identified song relationships. With this method, the system may propose music that is actually customized to the user's tastes, going beyond basic genre recommendations.

Making Interactive Playlists

The playlists that the system creates will be customized to the unique needs and emotions of each user. The system may, for instance, generate relaxing music after an active day or an energizing playlist for working out.

2. Literature Survey

Algorithms for Producing Music

Tackling the complex task of music generation, the research explores a range of

algorithms in the machine learning and artificial intelligence fields. By utilizing methods like transformer models, generative adversarial networks (GAN), Long Short-Term Memory (LSTM), Convolutional Neural Networks (CNN), and variational auto encoders (VAE), the research seeks to capture complex melodies and produce original works. CNNs are good at processing unprocessed audio data, whereas LSTM models are great at capturing long-range dependencies. Transformer models are experts at capturing sequential relationships in music creation, whereas GANs and VAEs provide methods for producing varied and excellent music.

Limitations

Although these algorithms show great potential, they are not without flaws. Obstacles include potential overfitting with small datasets, managing long-term dependencies, and poor accuracy in capturing subtle musical expressions. Furthermore, it is yet difficult to include domain-specific musical information into the creation process, which affects how expressive and coherent songs are.

Algorithms Employed in Reinforcement Learning for Music Generation

Autonomous learning in music composition systems is clarified by investigating reinforcement learning techniques, such as Deep Q-Networks (DQN), Proximal Policy Optimization (PPO), Policy Gradient Methods, and Deep Deterministic Policy Gradient (DDPG). By using actor-critic models to balance exploration and exploitation, these techniques allow for the creation of nuanced music. However, issues with representing intricate musical structures, expressing subjective musical originality, and handling computational complexity continue to exist, which has an impact on the effectiveness and interpretability of generated music.

Limitations

The drawbacks of using reinforcement learning algorithms to create music highlight problems with computing complexity, modeling, sample efficiency, and interpretability. These algorithms have the potential to be useful, but they might have trouble capturing the complex and individualized aspects of musical creation, which could limit their application in practical settings.

Methodology for Literature Review and Data Classification

Study finds, categorizes, and evaluates pertinent publications using a methodical literature review approach. The goal of the research is to comprehend the distribution and features of the examined literature using an organized method that includes data classification according to publication type, institution, and geographic territory. However, restrictions on language use, the study's scope, and the possibility of missing pertinent works could affect how thorough the results

Limitations

The literature review process has limitations, such as the possibility of leaving out pertinent works and exclusions based only on language. Furthermore, limitations

concerning the accessibility of unpublished works and the study's breadth in thoroughly covering various AI applications in music composition could impact the breadth of knowledge gained from the investigation.

3. Data Collection and Preprocessing

3.1 Description of the Dataset Used

Datasets used for AI MUSIC RECOMMENDATION AND GENERATION SYSTEM:

1. data.csv
2. data_by_artist.csv
3. data_by_geners.csv
4. data_by_years.csv
5. data_w_geners.csv

Purpose of dataset:

- Dataset is probably useful tool for researching music trends, popular artists, and genre traits.
- This data can be used for a variety of analysis and insights by industry experts, music Enthusiasts, and researchers.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	valence	year	acousticness	artists	danceability	duration_ms	energy	explicit	id	instrumentalness	key	liveness	loudness	mode	name	popularity	release_date	speechiness	tempo
2	0.0594	1921	0.982	['Sergei Ra	0.279	831667	0.211	0	4BJqT	0.878	10	0.665	-20.096	1	Piano Con	4	1921	0.0366	80.954
3	0.963	1921	0.732	['Dennis D	0.819	180533	0.341	0	7xPhf	0	7	0.16	-12.441	1	Clancy Lov	5	1921	0.415	60.936
4	0.0394	1921	0.961	['KHP Krid	0.328	500062	0.166	0	1o6I8	0.913	3	0.101	-14.85	1	Gati Bali	5	1921	0.0339	110.339
5	0.165	1921	0.967	['Frank Pa	0.275	210000	0.309	0	3ftBP	2.77E-05	5	0.381	-9.316	1	Danny Boy	3	1921	0.0354	100.109
6	0.253	1921	0.957	['Phil Rega	0.418	166693	0.193	0	4d6HC	1.68E-06	3	0.229	-10.096	1	When Irish	2	1921	0.038	101.665
7	0.196	1921	0.579	['KHP Krid	0.697	395076	0.346	0	4pyw	0.168	2	0.13	-12.506	1	Gati Mard	6	1921	0.07	119.824
8	0.406	1921	0.996	['John Mc	0.518	159507	0.203	0	5uNZ	0	0	0.115	-10.589	1	The Weari	4	1921	0.0615	66.221
9	0.0731	1921	0.993	['Sergei Ra	0.389	218773	0.088	0	02GD	0.527	1	0.363	-21.091	0	Morceaux	2	1921	0.0456	92.867
10	0.721	1921	0.996	['Ignacio C	0.485	161520	0.13	0	05xDj	0.151	5	0.104	-21.508	0	La MaÃ±ar	0	20-03-1921	0.0483	64.678
11	0.771	1921	0.982	['FortugÃ(	0.684	196560	0.257	0	08zfJv	0	8	0.504	-16.415	1	Il Etait Syr	0	1921	0.399	109.378
12	0.826	1921	0.995	['Maurice	0.463	147133	0.26	0	0BMk	0	9	0.258	-16.894	1	Dans La Vi	0	1921	0.0557	85.146
13	0.578	1921	0.994	['Ignacio C	0.378	155413	0.115	0	0F30V	0.906	10	0.11	-27.039	0	Por Que M	0	20-03-1921	0.0414	70.37
14	0.493	1921	0.99	['Georgel'	0.315	190800	0.363	0	0H3k2	0	5	0.292	-12.562	0	La VipÃ¨re	0	1921	0.0546	174.532
15	0.212	1921	0.912	['Mehmet	0.415	184973	0.42	0	0LcXz	0.89	8	0.108	-10.766	0	Ud Taksim	0	1921	0.114	70.758
16	0.493	1921	0.0175	['Zay Gatsl	0.527	205072	0.691	1	0MJZ	0.384	7	0.358	-7.298	1	Power Is F	0	27-03-1921	0.0326	159.935
17	0.282	1921	0.989	['Sergei Ra	0.384	221013	0.171	0	0NFe	0.82	7	0.116	-20.476	0	10 PrÃ©lu	4	1921	0.0319	107.698
18	0.218	1921	0.957	['Phil Rega	0.259	186467	0.212	0	0Nk5f	0.000222	2	0.236	-13.3	1	Come Bac	1	1921	0.0358	85.726
19	0.664	1921	0.996	['Hector B	0.541	250747	0.283	0	0POO	0.898	9	0.393	-14.808	1	RÃ¡kÃ³czy	0	1921	0.0477	108.986
20	0.0778	1921	0.148	['THE GUY'	0.604	204957	0.418	1	0QQm	0.0382	4	0.102	-11.566	0	When We	0	11-09-1921	0.0417	80.073
21	0.527	1921	0.971	['Christop	0.54	122000	0.0848	0	0QU5	0.00196	5	0.0887	-16.055	0	A Ballynur	0	1921	0.075	100.296
22	0.672	1921	0.994	['FortugÃ(	0.67	191333	0.113	0	0QvUI	0	10	0.213	-16.57	0	Je Suis To	0	1921	0.196	87.162
23	0.24	1921	0.994	['John Mc	0.4	187333	0.155	0	0RPK	4.33E-05	4	0.103	-13.976	1	Mother M	0	1921	0.0873	170.251

Fig (1): data.csv

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	mode	count	acousticness	artists	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence	popularity	key
2	1	9	0.590111111	"Cats" 1981	0.467222222	250318.5556	0.394003	0.011399851	0.2908333	-14.448	0.210388889	117.5181	0.3895	38.3333333	5
3	1	26	0.862538462	"Cats" 1983	0.441730769	287280	0.406808	0.081158264	0.3152154	-10.69	0.176211538	103.0442	0.268865	30.5769231	5
4	1	7	0.856571429	"Fiddler On	0.348285714	328920	0.286571	0.024592949	0.3257857	-15.23071	0.118514286	77.37586	0.354857	34.8571429	0
5	1	27	0.884925926	"Fiddler On	0.425074074	262890.963	0.24577	0.073587279	0.2754815	-15.63937	0.1232	88.66763	0.37203	34.8518519	0
6	1	7	0.510714286	"Joseph Anc	0.467142857	270436.1429	0.488286	0.009400291	0.195	-10.23671	0.098542857	122.8359	0.482286	43	5
7	1	36	0.609555556	"Joseph Anc	0.487277778	205091.9444	0.309906	0.004695657	0.2747667	-18.26639	0.098022222	118.6489	0.441556	32.7777778	5
8	1	2	0.725	"Mama" Hel	0.637	135533	0.512	0.186	0.426	-20.615	0.21	134.819	0.885	0	8
9	1	2	0.927	"Test for Vic	0.734	175693	0.474	0.0762	0.737	-10.544	0.256	132.788	0.902	3	10
10	1	122	0.173145082	"Weird Al" Y	0.662786885	218948.1967	0.695393	4.98E-05	0.1611016	-9.768705	0.084536066	133.0312	0.751344	34.2295082	9
11	1	15	0.544466667	$NOT	0.7898	137910.4667	0.532933	0.023062583	0.1803	-9.149267	0.293686667	112.3448	0.4807	67.5333333	1
12	1	2	0.239	$atori Zoom	0.883	141519	0.625	0	0.0765	-4.098	0.245	126.677	0.871	67	6
13	1	1	0.000122	$pyda	0.514	331240	0.899	0.0793	0.367	-5.115	0.0602	174.028	0.266	59	7
14	1	2	0.1481	$tupid Youn	0.854	190572	0.683	2.08E-06	0.1885	-6.997	0.221	100.7245	0.6255	57.5	1
15	1	125	0.14148488	$uicideBoy$	0.749344	146386.392	0.635552	0.045675096	0.2022528	-6.631304	0.156108	115.022	0.287286	61.8	1
16	1	13	0.624769231	'In The Heig	0.563615385	314023.6154	0.457692	8.69E-06	0.2043846	-8.338462	0.152453846	117.0068	0.467538	47.6923077	7
17	1	9	0.553888889	'Legally Blor	0.648444444	304211.8889	0.441111	2.54E-05	0.2146667	-11.45978	0.495111111	114.8084	0.524778	48.6666667	2
18	1	2	0.6045	'Legally Blor	0.7735	361780	0.3095	0	0.2222	-12.669	0.289	105.7	0.5965	48	10
19	1	16	0.10555625	'Til Tuesday	0.557125	255213.5	0.61225	0.023300253	0.1275875	-9.638125	0.03215	103.0803	0.532625	34.625	0
20	1	2	0.847	(((O)))	0.41	311837	0.169	0.00327	0.117	-11.422	0.0485	89.494	0.208	67	3
21	1	2	0.538	(Con La Part	0.731	361440	0.794	2.39E-05	0.0736	-4.182	0.0408	88.003	0.873	43	5
22	0	4	0.011235	(G)I-DLE	0.6405	196969	0.8	0	0.252	-4.4735	0.04875	151.0133	0.375	79.25	1
23	1	10	0.0116314	(Hed) P.E.	0.5892	253677.4	0.8792	2.03E-06	0.17112	-5.2598	0.17214	134.9694	0.427	45.8	2

Fig (2): data_by_artist.csv

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	mode	genres	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence	popularity	key
2	1	21st century	0.979333333	0.162883333	160297.6667	0.071317	0.60683367	0.3616	-31.514333	0.040566667	75.3365	0.103783	27.83333333	6
3	1	432hz	0.49478	0.299333333	1048887.333	0.450678	0.477761667	0.131	-16.854	0.076816667	120.2857	0.22175	52.5	5
4	1	8-bit	0.762	0.712	115177	0.818	0.876	0.126	-9.18	0.047	133.444	0.975	48	7
5	1	[]	0.65141702	0.52909256	232880.8903	0.419146	0.20530919	0.2186959	-12.288965	0.107871559	112.8574	0.513604	20.85988219	7
6	1	a cappella	0.676557305	0.538961246	190628.5409	0.316434	0.003003441	0.1722541	-12.479387	0.08285144	112.1104	0.448249	45.82007123	7
7	1	abstract	0.45921	0.516166667	343196.5	0.442417	0.849666667	0.1180667	-15.472083	0.046516667	127.8858	0.307325	43.5	1
8	1	abstract bea	0.342146667	0.623	229936.2	0.5278	0.333602612	0.0996533	-7.918	0.116373333	112.4138	0.493507	58.93333333	10
9	1	abstract hip	0.243854063	0.694570937	231849.2342	0.646235	0.024231263	0.1685429	-7.3493278	0.2142577	108.245	0.571391	39.79070248	2
10	0	accordeon	0.323	0.588	164000	0.392	0.441	0.0794	-14.899	0.0727	109.131	0.709	39	2
11	1	accordion	0.446125	0.6248125	167061.5625	0.373438	0.193738394	0.1603	-14.487063	0.0785375	112.8724	0.658688	21.9375	2
12	0	acid house	0.067950538	0.6774	297188.0538	0.724403	0.385891031	0.2334885	-9.3812	0.055855385	126.4921	0.567677	46.63846154	7
13	1	acid rock	0.256914508	0.447238993	259203.9143	0.580649	0.150925677	0.2304515	-11.433073	0.067599853	125.6595	0.534447	32.67595533	2
14	1	acid trance	0.00683	0.663	221160	0.925	0.703	0.185	-6.775	0.0449	132.687	0.843	62	9
15	1	acousmatic	0.917019106	0.420458285	185040.2596	0.238524	0.616248413	0.2610062	-20.964565	0.080617739	104.0746	0.324447	9.342524155	5
16	1	acoustic blue	0.761723539	0.606914772	204321.2509	0.360659	0.098918136	0.1830982	-12.554905	0.068815364	113.2845	0.633786	23.74606008	7
17	1	acoustic pop	0.490235026	0.535108317	235379.7836	0.47644	0.03333828	0.1577505	-9.2972479	0.041770665	117.8874	0.379415	53.11550475	7
18	1	acoustic pun	0.404900022	0.53911157	192332.1185	0.615824	0.061939951	0.2094806	-8.0107741	0.063570523	122.3085	0.730519	40.37649219	9
19	1	acoustic rock	0.613200794	0.524396825	195036	0.416003	0.321474922	0.1405184	-10.54062	0.034681085	122.6557	0.380299	51.10978836	7
20	0	action rock	0.229	0.412	198400	0.938	0.000259	0.106	-0.253	0.182	97.489	0.32	47	4
21	1	adoracion	0.432857143	0.504714286	302906.0714	0.52075	0	0.3159857	-8.0100714	0.038203571	130.7107	0.311571	46.39285714	9
22	1	adult standa	0.655648376	0.496327635	197331.5016	0.382054	0.118193574	0.1967767	-12.221776	0.053157818	113.3812	0.499376	31.63490211	0
23	1	adventista	0.784143333	0.5315	202876.65	0.319383	7.45E-06	0.14129	-9.3387667	0.028986667	113.3885	0.514517	27.26666667	6

Fig (3): data_by_geners.csv

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	mode	year	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence	popularity	key
2	1	1921	0.886896	0.418597333	260537.1667	0.231815	0.344878059	0.20571	-17.048667	0.073662	101.5315	0.379327	0.653333333	2
3	1	1922	0.938591549	0.482042254	165469.7465	0.237815	0.43419487	0.24072	-19.275282	0.11665493	100.8845	0.535549	0.14084507	10
4	1	1923	0.957246791	0.577340541	177942.3622	0.262406	0.371732725	0.227462	-14.129211	0.093948649	114.0107	0.625492	5.389189189	0
5	1	1924	0.94019986	0.549894068	191046.7076	0.344347	0.581700914	0.235219	-14.231343	0.092089407	120.6896	0.663725	0.661016949	10
6	1	1925	0.96260705	0.573863309	184986.9245	0.278594	0.418297361	0.237668	-14.146414	0.111917986	115.5219	0.621929	2.604316547	5
7	1	1926	0.660817217	0.599880261	156881.6575	0.211467	0.333093111	0.23237	-18.492538	0.483703628	109.648	0.43691	1.422351234	9
8	1	1927	0.936179455	0.648268293	184993.5984	0.264321	0.391328499	0.16845	-14.422374	0.113609593	114.8465	0.6597	0.801626016	7
9	1	1928	0.938616504	0.534287867	214827.9064	0.207948	0.49483548	0.175289	-17.191983	0.159911499	106.7723	0.495713	1.525773196	1
10	1	1929	0.601426586	0.647669853	168999.4128	0.241801	0.215204031	0.236	-16.530376	0.490000735	110.9484	0.63653	0.340336134	7
11	1	1930	0.936714937	0.518175884	195150.2853	0.333524	0.352205928	0.221311	-12.869221	0.119909667	109.8712	0.616238	0.926715177	2
12	1	1931	0.833039959	0.595221739	171553.4255	0.234497	0.221419642	0.227428	-16.516094	0.453618944	109.0253	0.513117	0.170807453	0
13	1	1932	0.935770518	0.55779761	195749.3725	0.302068	0.226356579	0.232496	-13.364056	0.139007371	115.1198	0.58816	2.151394422	7
14	1	1933	0.899897909	0.570290304	196219.2576	0.279899	0.18394894	0.209072	-13.069009	0.091123155	112.522	0.59941	6.89869754	7
15	1	1934	0.89114875	0.528705882	189356.1263	0.262131	0.276382267	0.213453	-14.756875	0.10245692	115.3908	0.558805	1.257785467	0
16	1	1935	0.77838556	0.55586917	220124.247	0.246367	0.225873298	0.2293	-15.41475	0.353912385	110.668	0.545578	1.501317523	7
17	1	1936	0.77231208	0.558005545	220809.1864	0.308389	0.257109996	0.221438	-14.612999	0.279029364	109.8888	0.564064	5.080909091	10
18	1	1937	0.865435785	0.54215723	223216.5586	0.311048	0.327087622	0.225968	-13.115143	0.085678843	115.2632	0.585789	3.328767123	7
19	1	1938	0.919280391	0.479977977	249177.0995	0.280981	0.378424635	0.237111	-14.290582	0.095957259	111.6253	0.514911	2.096247961	0
20	1	1939	0.9087381	0.5126828	221602.065	0.282672	0.277682469	0.239102	-13.90057	0.1284627	113.1746	0.559925	4.36	0
21	1	1940	0.84764405	0.52189235	182227.9445	0.310893	0.3168487	0.264335	-13.684048	0.24295765	108.4493	0.616709	0.93	7
22	1	1941	0.895737656	0.480481354	201904.575	0.265643	0.444951639	0.20184	-15.755536	0.091365208	107.9151	0.479456	1.357291667	7
23	1	1942	0.85293437	0.464633888	222361.1683	0.256079	0.392882048	0.212878	-15.029032	0.083678359	106.0084	0.477409	1.126634958	7

Fig (4): data_by_year.csv

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	genres	artists	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence	popularity	key	mode	count
2	['show tunes']	"Cats" 1981	0.590111111	0.467222222	250318.5556	0.394003	0.011399851	0.290833	-14.448	0.210388889	117.5181	0.3895	38.333333	5	1	9
3	[]	"Cats" 1983	0.862538462	0.441730769	287280	0.406808	0.081158264	0.315215	-10.69	0.176211538	103.0442	0.268865	30.576923	5	1	26
4	[]	"Fiddler Or	0.856571429	0.348285714	328920	0.286571	0.024592949	0.325786	-15.23071	0.118514286	77.37586	0.354857	34.857143	0	1	7
5	[]	"Fiddler Or	0.884925926	0.425074074	262890.963	0.24577	0.073587279	0.275481	-15.63937	0.1232	88.66763	0.37203	34.851852	0	1	27
6	[]	"Joseph An	0.510714286	0.467142857	270436.1429	0.488286	0.009400291	0.195	-10.23671	0.098542857	122.8359	0.482286	43	5	1	7
7	[]	"Joseph An	0.609555556	0.487277778	205091.9444	0.309906	0.004695657	0.274767	-18.26639	0.098022222	118.6489	0.441556	32.777778	5	1	36
8	[]	"Mama" He	0.725	0.637	135533	0.512	0.186	0.426	-20.615	0.21	134.819	0.885	0	8	1	2
9	[]	"Test for Vi	0.927	0.734	175693	0.474	0.0762	0.737	-10.544	0.256	132.788	0.902	3	10	1	2
10	['comedy rock', 'comic',	"Weird Al"	0.173145082	0.662786885	218948.1967	0.695393	4.98E-05	0.161102	-9.768705	0.084536066	133.0312	0.751344	34.229508	9	1	122
11	['emo rap', 'florida rap'	$NOT	0.544466667	0.7898	137910.4667	0.532933	0.023062583	0.1803	-9.149267	0.293686667	112.3448	0.4807	67.533333	1	1	15
12	['dark trap', 'meme rap	$atori Zoor	0.239	0.883	141519	0.625	0	0.0765	-4.098	0.245	126.677	0.871	67	6	1	2
13	[]	$pyda	0.000122	0.514	331240	0.899	0.0793	0.367	-5.115	0.0602	174.028	0.266	59	7	1	1
14	['asian american hip ho	$tupid You	0.1481	0.854	190572	0.683	2.08E-06	0.1885	-6.997	0.221	100.7245	0.6255	57.5	1	1	2
15	['dark trap', 'new orlea	$uicideBoy	0.14148488	0.749344	146386.392	0.635552	0.045675096	0.202253	-6.631304	0.156108	115.022	0.287286	61.8	1	1	125
16	['broadway', 'show tun	'In The Hei	0.624769231	0.563615385	314023.6154	0.457692	8.69E-06	0.204385	-8.338462	0.152453846	117.0068	0.467538	47.692308	7	1	13
17	['broadway', 'hollywoo	'Legally Blc	0.553888889	0.648444444	304211.8889	0.441111	2.54E-05	0.214667	-11.45978	0.495111111	114.8084	0.524778	48.666667	2	1	9
18	['show tunes']	'Legally Blc	0.6045	0.7735	361780	0.3095	0	0.2222	-12.669	0.289	105.7	0.5965	48	10	1	2
19	['boston rock', 'dance r	'Til Tuesda	0.10555625	0.557125	255213.5	0.61225	0.023300253	0.127588	-9.638125	0.03215	103.0803	0.532625	34.625	0	1	16
20	['experimental hip hop	(((O)))	0.847	0.41	311837	0.169	0.00327	0.117	-11.422	0.0485	89.494	0.208	67	3	1	2
21	[]	(Con La Par	0.538	0.731	361440	0.794	2.39E-05	0.0736	-4.182	0.0408	88.003	0.873	43	5	1	2
22	['k-pop', 'k-pop girl gro	(G)I-DLE	0.011235	0.6405	196969	0.8	0	0.252	-4.4735	0.04875	151.0133	0.375	79.25	1	0	4
23	['alternative metal', 'in	(Hed) P.E.	0.0116314	0.5892	253677.4	0.8792	2.03E-06	0.17112	-5.2598	0.17214	134.9694	0.427	45.8	2	1	10

Fig (5): data_w_geners.csv

Analysis of Dataset:

Each track's specific details are provided in the columns, including:

Valence: The degree of positivity in music that a piece conveys.

Year: The year the song was made available.

Acousticness: A confidence metric indicating the acoustic nature of the track.

Artist: The names of the musicians who were featured on the track.

Danceability: A measure of a song's suitability for dancing, based on its steady speed and rhythm.

Duration: musical piece is expressed in milliseconds (ms).

Energy: Denotes fervor and movement; upbeat music has a quick, boisterous tone.

Explicit Content: This field indicates whether or not the song has explicit material (0 for no, 1 for yes).

Instrumentalness: Indicates the degree of vocal presence.
Key: Denotes the key in which the composition is composed.
Liveness: Shows whether sounds from the audience are present.
Decibels (dB): used to measure total loudness.
Modality: Indicated by the mode (major as 1, minor as 0).
Track Name: The song or piece's title.
Popularity: Measures the song's popularity on a scale from 0% to 100%.
Release Date: Indicates the actual year or date the song was released.
Speechiness: Quantifies the amount of words that are said.
Tempo: Indicates the beats per minute (bpm) at which the song plays.

3.2 Data Cleaning and preprocessing steps

```
#  Reading data

data = pd.read_csv("data.csv")

genere_data = pd.read_csv('data_by_generes.csv')

year_data = pd.read_csv('data_by_year.csv')
```

Code reads the csv files: **"data.csv", "data_by_geners.csv", "data_by_years.csv"**

Each dataset contains information about "**music tracks, such as audio features, generes, and relese years".**

```
#  Handling missing values

# Drop rows with NaN values

data = data.dropna()

# Drop rows with infinite values

data = data.replace([np.inf, -np.inf], np.nan).dropna()
```

The above code snippet can understand that we are droping null values at first. Then we are replacing positive and negative infinite infinite values to null and then droping them.

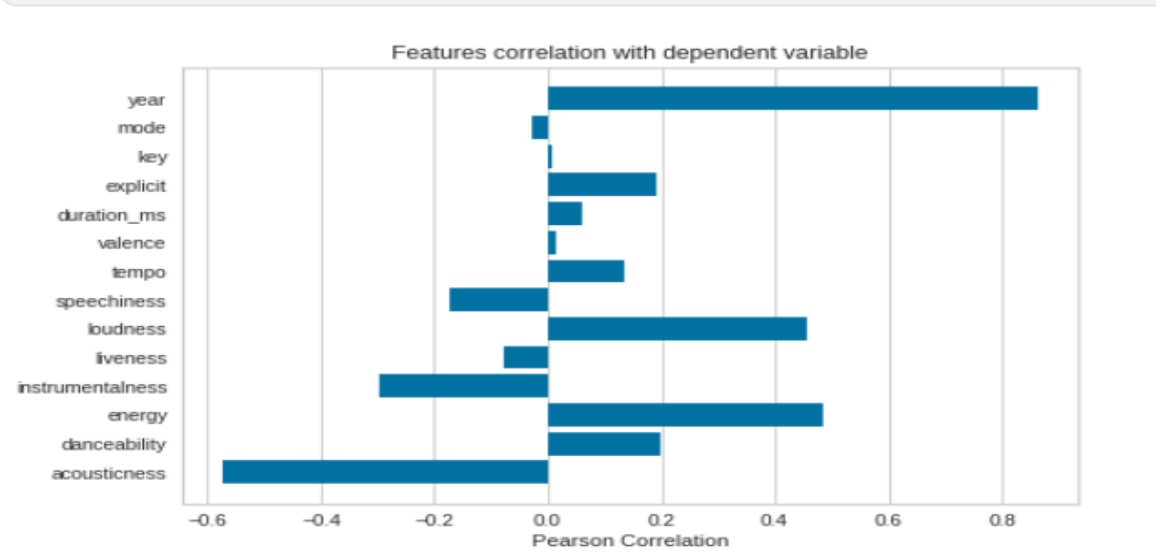

Fig (6): showing correlation with dependent variable

Data columns: year, mode, key, explicit, duration_ms, valence, tempo, energy, speechiness, loudness, liveness, instrumentalness, danceability, acousticness.

Dependent variable: popularity

Music over Time

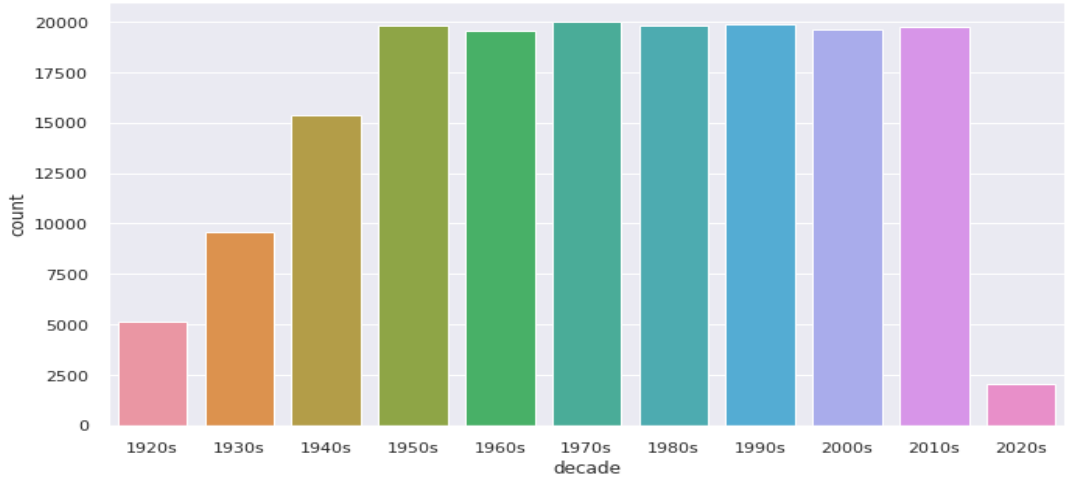

Fig (7): showing how music getting changes from 1920's to 2020's. Data is groped by year, we can understand how the overall sound has changed from 1921 to 2020.

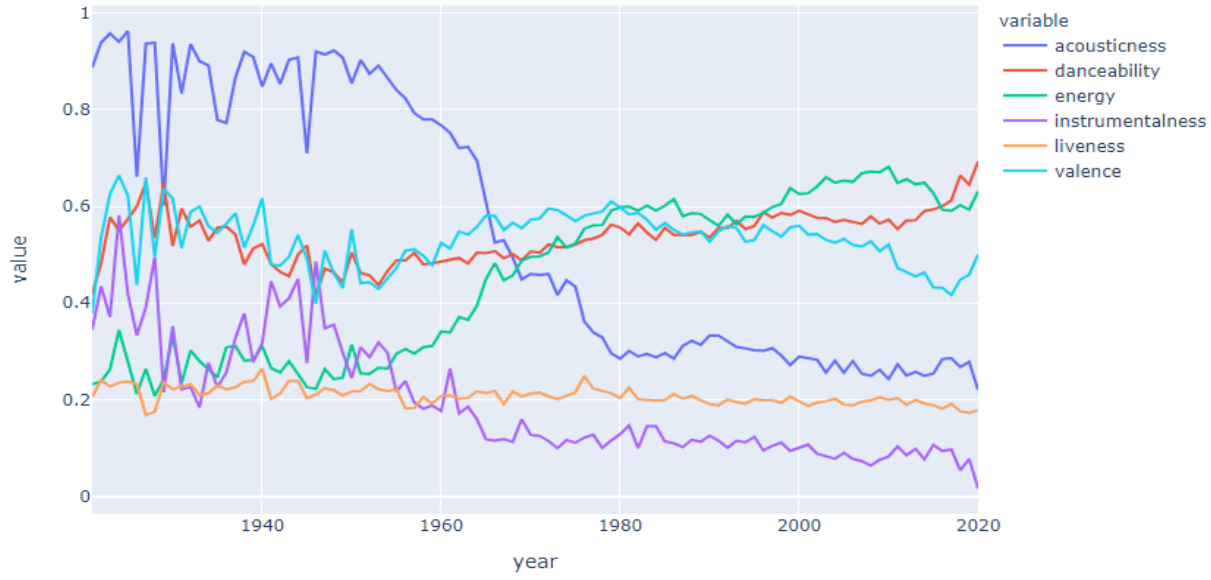

Fig (8): Visualizing trends of different variables using lines.

Characteristics of Different Genres

This dataset contains the audio features of different songs along with the audio features of different genres. We can use this information to compare different genres and understand their unique differences in sound.

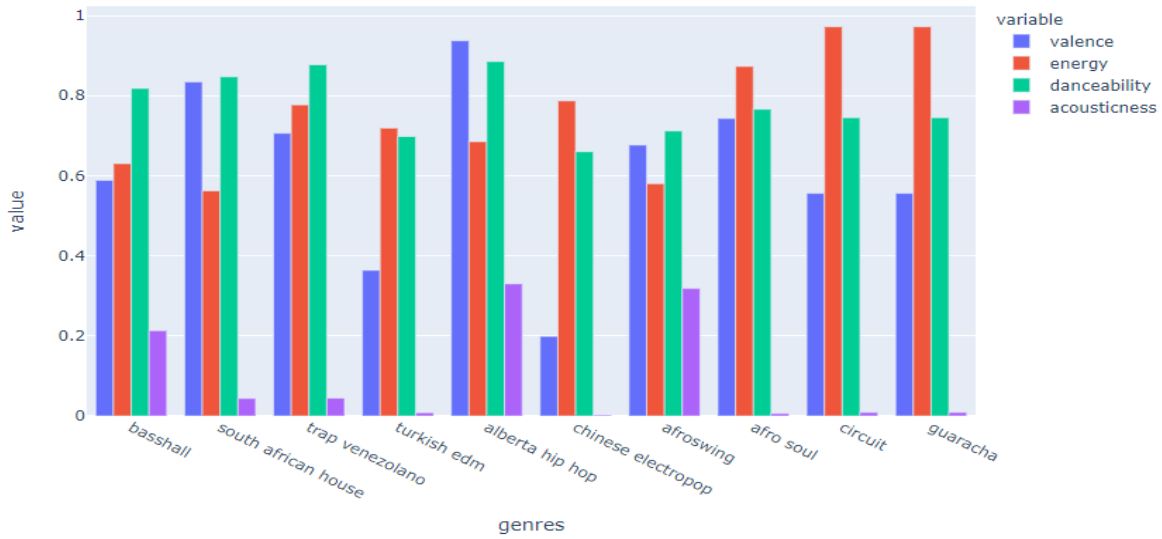

Fig (9): Representing the characteristics of different Genres

4. Methodology

4.1 Existing System

Traditional music recommendation systems frequently depend on basic algorithms that might not fully represent each user's unique tastes. These methods may prioritize suggesting songs that are currently popular or songs that are similar to one another by the same artist or genre.

These restrictions may result in:

Restricted Scope: Genre categories can be random and include a broad spectrum of musical qualities.

Repeated Recommendations: Popularity measures might not accurately capture the uniqueness of a song and lead to unintresting listening sessions.

Proposed AI-based Music Recommendation System

This paper tackles the short comings of current systems by proposing an AI-based music recommendation system. Through the following features, the system uses machine learning and music data to provide

In-depth Music Analysis:
Examines a greater variety of auditory elements, such as danceability, energy, valence, pace, and instrumentals, in order to obtain a more impartial comprehension of a song's attributes.

Identifying Song Similarities:
The system groups songs with similar musical characteristics by using audio

attributes to find patterns and correlations between songs.

Building User Profiles:
Gains insight into a user's taste in music by examining their listening history, which includes playlists they've made, artists they've followed, and songs they've enjoyed.

Generating Personalized Recommendations
Provides recommendations for new songs based on traits from music the user already likes, making the experience more tailored.

Creating Dynamic Playlists:
This feature creates playlists based on the individual requirements and moods of the user, such as a calming wind-down music or an energizing training playlist.

4.2 Proposed System Architecture

4.2.1 Architecture

Data Acquisition Module
Provides recommendations for new songs based on traits from music the user already likes, making the experience more tailored.

Machine Learning Module
This feature creates playlists based on the individual requirements and moods of the user, such as a calming wind-down music or an energizing training playlist.

Recommendation Module:
The recommendation module uses user profiles and music similarity data to provide tailored playlists and song recommendations.

User Interface Module
This offers a user interface via which users can communicate with the system, enter preferences, and get suggestions.

Data Import and Exploration:

A number of CSV files with music, genre, and year-grouped data are imported by the application. Next, it examines the data structure and looks for possible problems, such as missing values, using data.info().

Feature Correlation Analysis:

- To visualize data, the software imports the Yellowbrick library. It describes salient aspects of a song's auditory qualities, such as danceability, energy, and valence.
- To investigate the relationship between these attributes and the desired variable (popularity), it generates a Feature Correlation visualizer. This makes it easier to comprehend how various audio qualities could affect a song's level of popularity.

Data Cleaning and Preprocessing:
In order to deal with missing values, the computer either removes rows that have NaN values or substitutes them with suitable techniques (such mean imputation).

For additional analysis, it specifies the target variable and feature names.

Data Understanding Through Visualization and EDA (Exploratory Data Analysis)

Music Over Time:
By charting attributes like danceability and energy and organizing data by year, the application examines how music has changed over time. This can show patterns in popular music genres over several decades.

Characteristics of Different Genres:
To comprehend how different genres differ in terms of sound characteristics, it analyzes audio variables between them (e.g., greater valence scores for pop music compared to rock).

Clustering Genres with K-Means:
The software groups genres according to their numerical acoustic properties using K-Means clustering. This aids in locating genre clusters.

4.2.2 Machine Learning Algorithms Used

K-Means Clustering
A clustering approach that can be used to find comparable music by grouping songs with similar audio qualities. By clustering songs into groups, helps the system to identify the similarities and patterns in the music data.
This helps in creating user's profiles, personalized music recommendations, enhance user's experiences and music recommendations.

Nearest Neighbors:
A method for locating comparable data points that may be utilized to suggest music that matches a user's taste. This can identify similarities and patterns among users. Users who have listed to similar songs or genres in the past are considered to be "nearest neighbours" in the feature space.

Matrix Factorization:
The fundamental idea behind this matrix factorization is to represent the user item interaction data, typically stored in a user-item matrix, as a product of two lower dimensional matrices, known as latent feature matrices.

User matrix: In this matrix, latent features are represented as columns and users as rows.
A user's preferences across several latent features are represented by each row in the user matrix.

Item matrix: Music tracks are shown as rows in this item matrix, while latent features are shown as columns. The properties of a music track across several latent features are represented by each row of the item matrix.

4.2.3 Model Selection and Evaluation Metrics
It is likely that the system will select the particular machine learning models it

uses based on criteria such as:

Data characteristics:
The type and format of the music data that is available are among the data characteristics.

Task requirements:
Whether to prioritize recommendation generating, user preference learning, or music similarity.

Performance metrics:
The efficiency of the selected models will be evaluated using metrics such as accuracy, precision, recall, and suggestion diversity.

Euclidean Distance: the distance between two points x and y in n-dimensional space is given by

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(xi - yi)^2}$$

Centroid Calculation: Mean of all the data points in its cluster.

$$Cj = \frac{1}{Sj}\sum_{x \in Sj} x$$

Nearest Neighbour: The nearest neighbor search locates the data point in X that is closest to query point q given a set of data points X and a distance measure (usually the Euclidean distance). The above-mentioned Euclidean distance formula is comparable to the method for determining the nearest neighbor.

Matrix Factorization: the user-item matrix R is broken down into two lower-rank matrices, U and V, whose products roughly equal R. Gradient descent and other optimization techniques are used to minimize the reconstruction error in order to approximate the result:

$$R \sim UV^2$$

Objective Function: In matrix factorization, the goal function is usually to minimize the Frobenius norm of the difference between R and its approximation, UV T.

$R - UV^{t}{}_{F}^{2}$ where, F =Frobenius.

5. Experimental Setup

5.1 Hardware and Software Used

The selection of hardware and software is essential for an AI music recommendation and generation system to ensure effective algorithm processing and implementation.

Hardware

CPU and GPU:
The computational demands of machine learning algorithms involve a strong CPU or GPU, particularly during the training and inference stages. Deep learning applications are particularly well-suited for GPUs due to their exceptional performance in parallel processing tasks.

RAM, or memory:
Large datasets must be efficiently stored and managed, which requires enough RAM. Large music datasets might require a lot of memory for feature extraction, pre-processing, and model training.

Storage:
To store models for training, temporarily outcomes, and music datasets, high-capacity storage disks are required. Speedier read/write rates are provided by solid-state drives (SSDs) in comparison to conventional hard disk drives (HDDs), enabling speedier data access.

Software:

Python
Python's large libraries and signal processing and machine learning frameworks make it the main programming language used to create AI music recommendation and generating systems. NumPy, Pandas, and Scikit-learn are libraries that are frequently used for training models and manipulating data.

Machine Learning Libraries:
For the purpose of implementing machine learning and deep learning models, libraries like Tensor Flow, PyTorch, and Keras offers strong tools. These libraries provide pre-configured modules for assessment metrics, optimization techniques, and neural network topologies.

Development Environments:
Popular development environments for AI projects that provide interactive computing and collaborative capabilities are Kaggle, Google Colab, and Jupyter Notebook. These platforms give customers access to GPU-accelerated computing resources, which facilitates the effective training of complicated models.

Database Management Systems (DBMS):
User preferences, recommendation history, and music information can be stored and managed using DBMS platforms such as SQLite, MySQL, or PostgreSQL. These systems offer effective methods for storing and retrieving data, making them suitable for managing substantial datasets.

Visualization Tools:
Data visualization and analysis are done using visualization packages such as Matplotlib and Seaborn. With the use of these tools, users may clearly and understandably view music attributes, model predictions, and recommendation outcomes.

5.2 Parameter Tuning Process

Optimizing the k-means clustering algorithm's parameters to improve the system's accuracy and performance is known as the "parameter tuning process" for the AI music recommendation and generation system. The goal of this procedure is to determine the ideal values for several parameters, including the distance metric and the number of clusters (k).

Number of Clusters (k):

The number of clusters (k) in the k-means method is one of its important parameters. The efficacy and granularity of the clustering process are determined by the ideal value of k. Several methods, including the elbow method, silhouette score, and silhouette analysis, can be used to adjust this value. These methods assist in determining the value of k that optimizes intra-cluster similarity while minimizes inter-cluster similarity by analyzing the clustering results for various values of k.

Distance Metric:

The clustering findings are influenced by the distance metric that is selected to determine the degree of similarity between the data points. Cosine similarity, Manhattan distance, and Euclidean distance are examples of common distance measures. The right distance metric must be chosen based on the goals of the clustering process and the type of music data. In order to fine-tune the distance metric, one must test out several metrics and assess how they affect the performance of clustering.

Initialization Method:

The initialization method, which establishes the initial location of cluster centroids, is an additional parameter in k-means clustering. The clustering algorithm's stability and convergence may be impacted by the initialization technique selected. K-means++ initialization and random initialization are two popular initialization techniques. Comparing the effectiveness of several starting techniques and choosing the one that produces the most reliable and accurate clustering results is known as parameter tweaking.

Convergence Criteria:

The conditions under which the k-means algorithm breaks are specified by the convergence criterion. The process usually repeats until the centroids stabilize or until a predetermined number of iterations is reached. A few examples of parameters to tweak include the maximum number of iterations and the centroid movement threshold while tuning the convergence criteria. This prevents overfitting and early algorithm termination while guaranteeing an efficient convergence of the algorithm.

Evaluation Metrics:

Evaluation measures like the pattern score, are used to gauge the quality of the clustering results throughout the parameter tweaking phase. By offering

quantifiable assessments of clustering performance, these metrics make it possible to compare various parameter combinations in an unbiased manner.

6. Results and Discussion

Clustering Genres with K-means

K-means clustering is used to divide the genres in this dataset into ten clusters based on the numerical audio features of each genres. This t-Distributed Stochastic Neighbor Embedding (t-SNE) is used to reduce dimensionaility of the genre data and visualize it in a two-dimensional space.

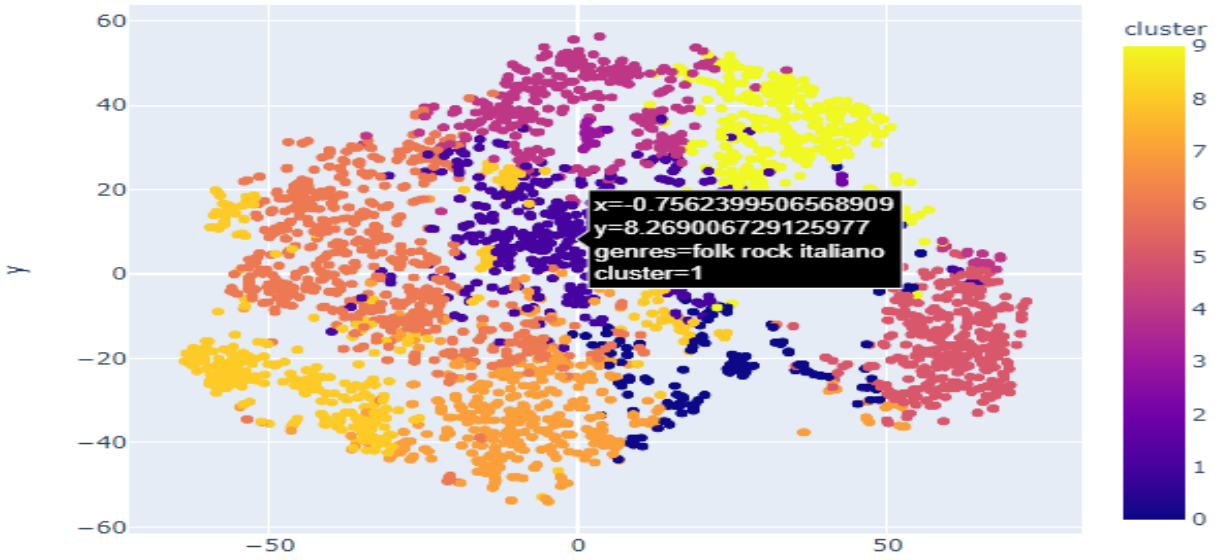

Fig (10): Clustering genres with k-means

Clustering song with k-means

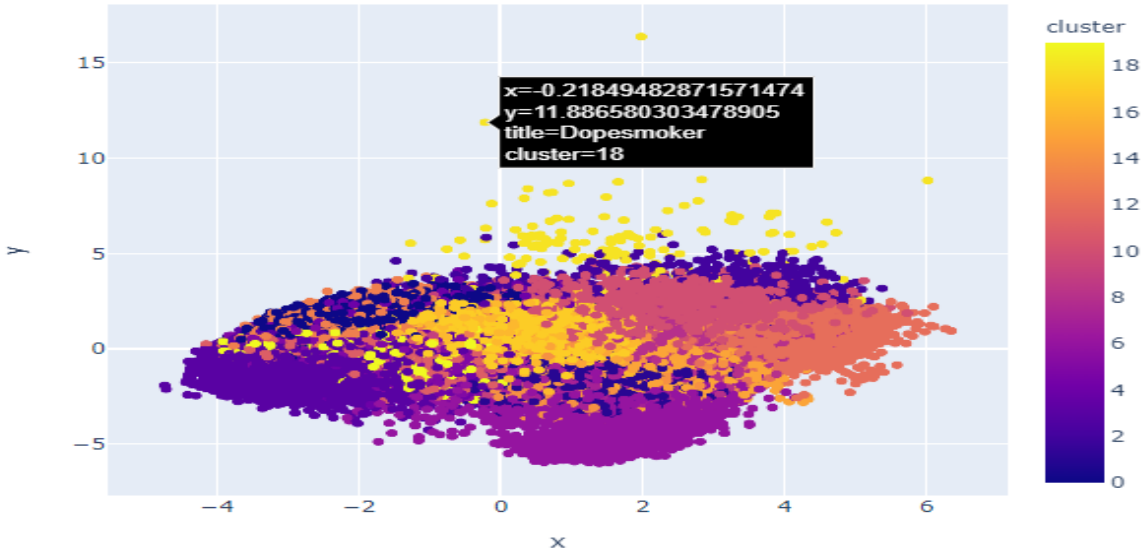

Fig (11): Clustering songs with k-means

This uses PCA to reduce dimensionality and Plotly Express to clusters in a two-dimensional space after K-means clustering is applied to the song data and cluster labels are assigned to each song.

Output:

It is evident from the analysis and visualizations that comparable song types are clustered together, and that comparable genres also contain data points that are positioned close to one another.

Songs within related genres will sound similar and originate from similar eras. Similarly, genres will sound similar. By using the data points of the songs a user has listened to, we may leverage this concept to create a recommendation system that suggests songs based on adjacent data points.

Suggested Songs:
A list of songs is provided by the system, together with information about the artists, albums, and years of release.

Code Excerpt:

A snippet of Python code output can be seen in the image below.
It defines a function named recommend_song, whose input is a list of dictionaries, most likely containing song data.
Based on user preferences, the function most likely generates song recommendations.

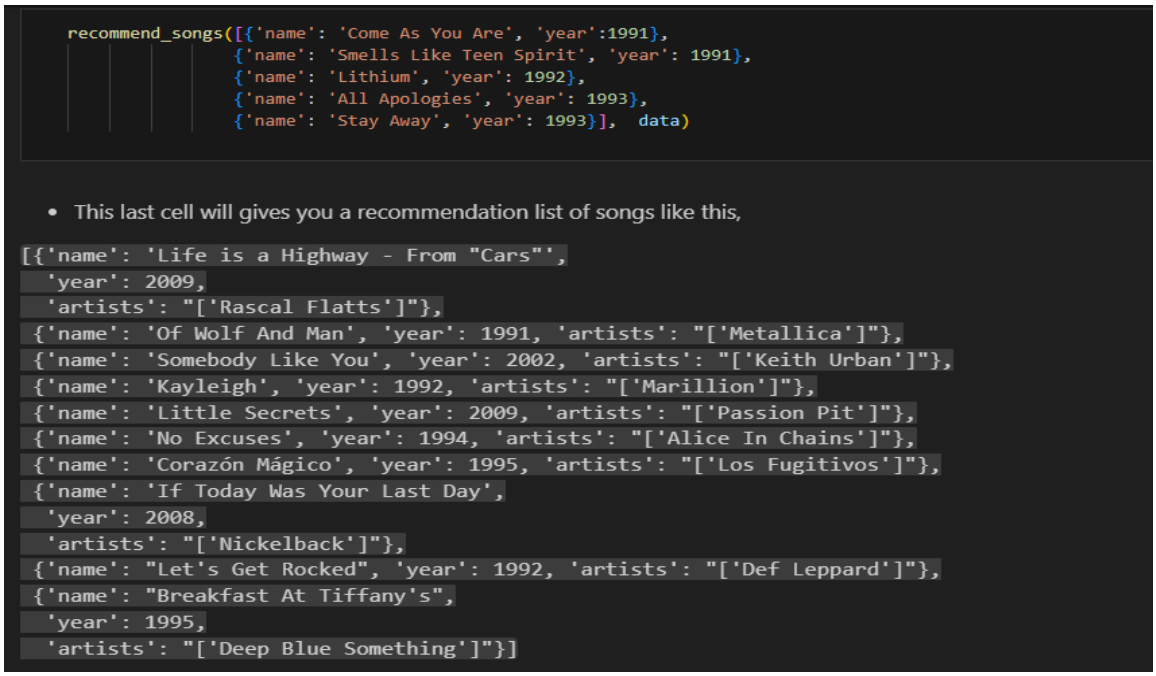

Fig (12): Output showing AI Music Recommendation and Generation.

7. Conclusion and Future Scope

Conclusion

The AI-based system for creating playlists and recommending music seems promising. It uses sensory features and music genres to find similarities between songs and suggest music based on common traits. The system also uses K Means clustering to group songs based on how they sound. However, it might struggle to capture all the unique aspects of music creation, which could affect how well it works. To improve, the system could consider user feedback, use better machine learning techniques, and refine how it recommends music. This could help make the music listening experience more personalized and varied for users in the future.

Future Scope

The AI-based music recommendation and playlist creation model have a bright future. To make it even better, we can add sentiment analysis to suggest music based on how users feel. Using advanced machine learning like GANs and VAEs could help create more diverse music, appealing to more people. It would also help to use real-time data to keep up with current music trends and what users like. Adding social features would let users share and find music together, making the experience more fun and interactive. By doing these things, the model can become a great music platform that not only suggests personalized music but also helps people make music and connect with others.

8. References

- Born, G., Morris, J., Diaz, F., & Anderson, A. (2021). Artificial intelligence, music recommendation, and the curation of culture.
- Afchar, D., Melchiorre, A., Schedl, M., Hennequin, R., Epure, E., & Moussallam, M. (2022). Explainability in music recommender systems. *AI Magazine*, *43*(2), 190-208.
- Li, J., & Li, S. (2011). Music Recommendation Using Content-Based Filtering and Collaborative Filtering with Data Fusion Approach. *International Conference on Advanced Data Mining and Applications.
- Kim, S., & Lee, J. H. (2015). Music Recommendation Based on Emotional Cues. International Conference on Music Information Retrieval.
- Lian, J., & Gou, L. (2013). Combining Collaborative Filtering and Sentiment Analysis for Music Recommendation. IEEE International Conference on Big Data.
- Andjelkovic, I., D. Parra, and J. O'Donovan. 2016. "Moodplay: Interactive Mood-based Music Discovery and Recommendation." In Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization, 275–9. New York: ACM
- Abdollahi, B., and O. Nasraoui. 2016. "Explainable Matrix Factorization for Collaborative Filtering." In Proceedings of the 25th International Conference

Companion on World Wide Web, 5–6

- Goto, M., and R. B. Dannenberg. 2019. "Music Interfaces based on Automatic Music Signal Analysis: New Ways to Create and Listen to Music." IEEE Signal Processing Magazine 36(1): 74–81

- Abdollahpouri, H., Burke, R., & Mobasher, B. (2019). Managing Popularity Bias in Recommender Systems with Personalized Re-ranking. arXiv preprint arXiv:1901.07555.

- Aggarwal, C. C. (2016). Recommender systems (pp. 1-28). Cham: Springer International Publishing.

9. Appendix

Building Music Recommendation System using Spotify Dataset

Hello and welcome to my kernel. In this kernel, I have created Music Recommendation System using Spotify Dataset. To do this, I presented some of the visualization processes to understand data and done some EDA(Exploratory Data Analysis) so we can select features that are relevant to create a Recommendation System.

```
import os
import numpy as np
import pandas as pd

import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.manifold import TSNE
from sklearn.decomposition import PCA
from sklearn.metrics import euclidean_distances
from scipy.spatial.distance import cdist


        import warnings
warnings.filterwarnings("ignore")
```

Read Data

```
        data = pd.read_csv("../input/spotify-dataset/data/data.csv")
  genre_data = pd.read_csv('../input/spotify-dataset/data/data_by_genres.csv')
year_data = pd.read_csv('../input/spotify-dataset/data/data_by_year.csv')
```

```
print(data.info())

print(genre_data.info())

print(year_data.info())

from yellowbrick.target import FeatureCorrelation

feature_names = ['acousticness', 'danceability', 'energy', 'instrumentalness',
       'liveness', 'loudness', 'speechiness', 'tempo',
'valence','duration_ms','explicit','key','mode','year']

X, y = data[feature_names], data['popularity']

# Create a list of the feature names
features = np.array(feature_names)

# Instantiate the visualizer
visualizer = FeatureCorrelation(labels=features)

plt.rcParams['figure.figsize']=(20,20)
visualizer.fit(X, y)     # Fit the data to the visualizer
visualizer.show()
```

We are going to check for all the analysis with the target as **'popularity'**. Before going to do that let's check for the Feature Correlation by considering a few features and for that, I'm going to use the **yellowbrick** package.

Data Understanding by Visualization and EDA

Music over Time

Using the data grouped by year, we can understand how the overall sound of music has changed from 1921 to 2020.

```
def get_decade(year):
    period_start = int(year/10) * 10
    decade = '{}s'.format(period_start)
    return decade

data['decade'] = data['year'].apply(get_decade)

sns.set(rc={'figure.figsize':(11 ,6)})
sns.countplot(data['decade'])


sound_features = ['acousticness', 'danceability', 'energy', 'instrumentalness',
'liveness', 'valence']
fig = px.line(year_data, x='year', y=sound_features)
fig.show()
```

Characteristics of Different Genres

 # This dataset contains the audio features for different songs along with the audio features for different genres. We can use this information to compare different genres and understand their unique differences in sound.

```
        top10_genres = genre_data.nlargest(10, 'popularity')

fig = px.bar(top10_genres, x='genres', y=['valence', 'energy', 'danceability',
'acousticness'], barmode='group')
fig.show()
```

Clustering Genres with K-Means

```
# Here, the simple K-means clustering algorithm is used to divide the genres in
this dataset into ten clusters based on the numerical audio features of each
genres.

from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

cluster_pipeline = Pipeline([('scaler', StandardScaler()), ('kmeans',
KMeans(n_clusters=10, n_jobs=-1))])
X = genre_data.select_dtypes(np.number)
cluster_pipeline.fit(X)
genre_data['cluster'] = cluster_pipeline.predict(X)

# Visualizing the Clusters with t-SNE

from sklearn.manifold import TSNE

tsne_pipeline = Pipeline([('scaler', StandardScaler()), ('tsne',
TSNE(n_components=2, verbose=1))])
genre_embedding = tsne_pipeline.fit_transform(X)
projection = pd.DataFrame(columns=['x', 'y'], data=genre_embedding)
projection['genres'] = genre_data['genres']
projection['cluster'] = genre_data['cluster']

fig = px.scatter(
    projection, x='x', y='y', color='cluster', hover_data=['x', 'y', 'genres'])
fig.show()
```

Clustering Songs with K-Means

```
song_cluster_pipeline = Pipeline([('scaler', StandardScaler()),
                                  ('kmeans', KMeans(n_clusters=20,
                                   verbose=False, n_jobs=4))
                                 ], verbose=False)

X = data.select_dtypes(np.number)
```

```
number_cols = list(X.columns)
song_cluster_pipeline.fit(X)
song_cluster_labels = song_cluster_pipeline.predict(X)
data['cluster_label'] = song_cluster_labels


# Visualizing the Clusters with PCA

from sklearn.decomposition import PCA

pca_pipeline = Pipeline([('scaler', StandardScaler()), ('PCA',
PCA(n_components=2))])
song_embedding = pca_pipeline.fit_transform(X)
projection = pd.DataFrame(columns=['x', 'y'], data=song_embedding)
projection['title'] = data['name']
projection['cluster'] = data['cluster_label']

fig = px.scatter(
    projection, x='x', y='y', color='cluster', hover_data=['x', 'y', 'title'])
fig.show()
```

Build Recommender System

```
# Based on the analysis and visualizations, it’s clear that similar genres tend to
have data points that are located close to each other while similar types of songs
are also clustered together.

# This observation makes perfect sense. Similar genres will sound similar and
will come from similar time periods while the same can be said for songs within
those genres. We can use this idea to build a recommendation system by taking
the data points of the songs a user has listened to and recommending songs
corresponding to nearby data points.

        !pip install spotipy


import spotipy
from spotipy.oauth2 import SpotifyClientCredentials
from collections import defaultdict

sp =
spotipy.Spotify(auth_manager=SpotifyClientCredentials(client_id=os.environ["S
POTIFY_CLIENT_ID"],

client_secret=os.environ["SPOTIFY_CLIENT_SECRET"]))

def find_song(name, year):
    song_data = defaultdict()
    results = sp.search(q= 'track: {} year: {}'.format(name,year), limit=1)
    if results['tracks']['items'] == []:
        return None
```

```
    results = results['tracks']['items'][0]
    track_id = results['id']
    audio_features = sp.audio_features(track_id)[0]

    song_data['name'] = [name]
    song_data['year'] = [year]
    song_data['explicit'] = [int(results['explicit'])]
    song_data['duration_ms'] = [results['duration_ms']]
    song_data['popularity'] = [results['popularity']]

    for key, value in audio_features.items():
        song_data[key] = value

    return pd.DataFrame(song_data)

from collections import defaultdict
from sklearn.metrics import euclidean_distances
from scipy.spatial.distance import cdist
import difflib

number_cols = ['valence', 'year', 'acousticness', 'danceability', 'duration_ms',
'energy', 'explicit',
 'instrumentalness', 'key', 'liveness', 'loudness', 'mode', 'popularity',
'speechiness', 'tempo']


def get_song_data(song, spotify_data):

    try:
        song_data = spotify_data[(spotify_data['name'] == song['name'])
                                 & (spotify_data['year'] == song['year'])].iloc[0]
        return song_data

    except IndexError:
        return find_song(song['name'], song['year'])


def get_mean_vector(song_list, spotify_data):

    song_vectors = []

    for song in song_list:
        song_data = get_song_data(song, spotify_data)
        if song_data is None:
            print('Warning: {} does not exist in Spotify or in
database'.format(song['name']))
            continue
        song_vector = song_data[number_cols].values
        song_vectors.append(song_vector)

    song_matrix = np.array(list(song_vectors))
```

```
    return np.mean(song_matrix, axis=0)


def flatten_dict_list(dict_list):

    flattened_dict = defaultdict()
    for key in dict_list[0].keys():
        flattened_dict[key] = []

    for dictionary in dict_list:
        for key, value in dictionary.items():
            flattened_dict[key].append(value)

    return flattened_dict


def recommend_songs( song_list, spotify_data, n_songs=10):

    metadata_cols = ['name', 'year', 'artists']
    song_dict = flatten_dict_list(song_list)

    song_center = get_mean_vector(song_list, spotify_data)
    scaler = song_cluster_pipeline.steps[0][1]
    scaled_data = scaler.transform(spotify_data[number_cols])
    scaled_song_center = scaler.transform(song_center.reshape(1, -1))
    distances = cdist(scaled_song_center, scaled_data, 'cosine')
    index = list(np.argsort(distances)[:, :n_songs][0])

    rec_songs = spotify_data.iloc[index]
    rec_songs = rec_songs[~rec_songs['name'].isin(song_dict['name'])]
    return rec_songs[metadata_cols].to_dict(orient='records')


recommend_songs([{'name': 'Come As You Are', 'year':1991},
                 {'name': 'Smells Like Teen Spirit', 'year': 1991},
                 {'name': 'Lithium', 'year': 1992},
                 {'name': 'All Apologies', 'year': 1993},
                 {'name': 'Stay Away', 'year': 1993}], data)
```