Music Recommendation System

Milestone: Project Proposal

Group 24

Harshini Daggubati
Sai Vinay Teja Jakku

857-200-9927 (Harshini Daggubati)

617-412-7383 (Sai Vinay Teja Jakku)

daggubati.h@northeastern.edu

jakku.s@northeastern.edu

Percentage of Effort Contributed by Student 1:	**50%**
Percentage of Effort Contributed by Student 2:	**50%**
Signature of Student 1:	**HARSHINI DAGGUBATI**
Signature of Student 2:	**SAI VINAY TEJA JAKKU**
Submission Date:	**05/01/2022**

Data Mining in Engineering

Problem Setting:

Spotify is an application that offers online listening and streaming services in the form of music or podcasts to people around the globe. Spotify has a large collection of songs and podcasts, and users can choose to listen to any song and give the song a rating. Spotify uses these user ratings and other metrics to recommend similar songs to the user.

Problem Definition:

This project aims to do an Exploratory data analysis of the Spotify open data using Python libraries. Further, we will do visualization analysis for this dataset to understand factors like the correlation between different variables. We will use supervised learning methods to predict the popularity of the song and report the prediction accuracy and errors.

We aim to find the answers to the below-proposed business questions:

- Why makes a song popular?
- Do the valence and different factors affect the song popularity?
- How important is artist and genre when it comes to determining the popularity of a song?

Data Sources:

Link of data sources as below:

- https://www.kaggle.com/vatsalmavani/spotify-dataset

Data Description:

The dataset contains 3 datasets called Data, genre_data, and year_data. We will primarily work using these three datasets to determine which factors affect the popularity of a song. Data has in total 19 columns and more than 170000 rows and has columns like Valence, Acousticness, and other variables like danceability and duration. The dataset contains columns like song ID, year, artists, energy, liveness, etc. of the song.

Data Collection and Processing:

The data has been collected from the Kaggle website

Importing the dataset

We import the dataset using the pandas' library's read_csv() function. Three datasets will be used in this project:

1. **Data.csv:** This dataset contains the Spotify song listings detail in brief. It has columnslike a songID, valence, liveness, etc.
2. **genre_data.csv:** This dataset contains the Spotify song listing and its genre. It has columns like genres, tempo, loudness, etc.
3. **year_data.csv:** This dataset contains the Spotify song listings and its release dates. It has columns like year, energy, speechiness, etc.

Below is a snapshot of the code from the notebook to import the dataset.

```
In [6]: #Importing Pyhton libraries
        import pandas as pd
        import numpy as np

        #Importing the dataset in the notebook
        data = pd.read_csv('data.csv')
        genre_data = pd.read_csv('data_by_genres.csv')
        year_data = pd.read_csv('data_by_year.csv')
```

After importing the dataset, we use the head function to view the top rows of each dataset.

```
In [7]: #checking if the dataset is imported properly
        data.head()
```

Out[7]:

	valence	year	acousticness	artists	danceability	duration_ms	energy	explicit	id	instrumentalness	key	liveness	loudnes
0	0.0594	1921	0.982	['Sergei Rachmaninoff', 'James Levine', 'Berli...	0.279	831667	0.211	0	4BJqT0PrAfrxzMOxytFOIz	0.878000	10	0.665	-20.09
1	0.9630	1921	0.732	['Dennis Day']	0.819	180533	0.341	0	7xPhfUan2yNtyFG0cUWkt8	0.000000	7	0.160	-12.44
2	0.0394	1921	0.961	['KHP Kridhamardawa Karaton Ngayogyakarta Hadi...	0.328	500062	0.166	0	1o6I8BglA6ylDMrIELygv1	0.913000	3	0.101	-14.85
3	0.1650	1921	0.967	['Frank Parker']	0.275	210000	0.309	0	3ftBPsC5vPBKxYSee08FDH	0.000028	5	0.381	-9.31
4	0.2530	1921	0.957	['Phil Regan']	0.418	166693	0.193	0	4d6HGyGT8e121BsdKmw9v6	0.000002	3	0.229	-10.09

```
In [8]: #checking if the dataset is imported properly
        genre_data.head()
```

Out[8]:

	mode	genres	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence	popularity	k
0	1	21st century classical	0.979333	0.162883	1.602977e+05	0.071317	0.606834	0.361600	-31.514333	0.040567	75.336500	0.103783	27.833333	
1	1	432hz	0.494780	0.299333	1.048887e+06	0.450678	0.477762	0.131000	-16.854000	0.076817	120.285667	0.221750	52.500000	
2	1	8-bit	0.762000	0.712000	1.151770e+05	0.818000	0.876000	0.126000	-9.180000	0.047000	133.444000	0.975000	48.000000	
3	1	[]	0.651417	0.529093	2.328809e+05	0.419146	0.205309	0.218696	-12.288965	0.107872	112.857352	0.513604	20.859882	
4	1	a cappella	0.676557	0.538961	1.906285e+05	0.316434	0.003003	0.172254	-12.479387	0.082851	112.110362	0.448249	45.820071	

```
In [9]: #checking if the dataset is imported properly
        year_data.head()
```

click to scroll output; double click to hide

Out[9]:

	mode	year	acousticness	danceability	duration_ms	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence	popularity	key
0	1	1921	0.886896	0.418597	260537.166667	0.231815	0.344878	0.205710	-17.048667	0.073662	101.531493	0.379327	0.653333	2
1	1	1922	0.938592	0.482042	165469.746479	0.237815	0.434195	0.240720	-19.275282	0.116655	100.884521	0.535549	0.140845	10
2	1	1923	0.957247	0.577341	177942.362162	0.262406	0.371733	0.227462	-14.129211	0.093949	114.010730	0.625492	5.389189	0
3	1	1924	0.940200	0.549894	191046.707627	0.344347	0.581701	0.235219	-14.231343	0.092089	120.689572	0.663725	0.661017	10
4	1	1925	0.962607	0.573863	184986.924460	0.278594	0.418297	0.237668	-14.146414	0.111918	115.521921	0.621929	2.604317	5

Data Processing:

In this step, we will use the describe () function to get an idea about the mean, median, etc of the numeric variables in the dataset. We will also use the info () function to see if all the columns have the correct data type.

```
In [10]: data.describe()
```

Out[10]:

	valence	year	acousticness	danceability	duration_ms	energy	explicit	instrumentalness	key	livene
count	170653.000000	170653.000000	170653.000000	170653.000000	1.706530e+05	170653.000000	170653.000000	170653.000000	170653.000000	170653.0000
mean	0.528587	1976.787241	0.502115	0.537396	2.309483e+05	0.482389	0.084575	0.167010	5.199844	0.2058
std	0.263171	25.917853	0.376032	0.176138	1.261184e+05	0.267646	0.278249	0.313475	3.515094	0.1748
min	0.000000	1921.000000	0.000000	0.000000	5.108000e+03	0.000000	0.000000	0.000000	0.000000	0.0000
25%	0.317000	1956.000000	0.102000	0.415000	1.698270e+05	0.255000	0.000000	0.000000	2.000000	0.0988
50%	0.540000	1977.000000	0.516000	0.548000	2.074670e+05	0.471000	0.000000	0.000216	5.000000	0.1360
75%	0.747000	1999.000000	0.893000	0.668000	2.624000e+05	0.703000	0.000000	0.102000	8.000000	0.2610
max	1.000000	2020.000000	0.996000	0.988000	5.403500e+06	1.000000	1.000000	1.000000	11.000000	1.0000

```
In [14]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 170653 entries, 0 to 170652
Data columns (total 19 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   valence           170653 non-null  float64
 1   year              170653 non-null  int64  
 2   acousticness      170653 non-null  float64
 3   artists           170653 non-null  object 
 4   danceability      170653 non-null  float64
 5   duration_ms       170653 non-null  int64  
 6   energy            170653 non-null  float64
 7   explicit          170653 non-null  int64  
 8   id                170653 non-null  object 
 9   instrumentalness  170653 non-null  float64
 10  key               170653 non-null  int64  
 11  liveness          170653 non-null  float64
 12  loudness          170653 non-null  float64
 13  mode              170653 non-null  int64  
 14  name              170653 non-null  object 
 15  popularity        170653 non-null  int64  
 16  release_date      170653 non-null  object 
 17  speechiness       170653 non-null  float64
 18  tempo             170653 non-null  float64
dtypes: float64(9), int64(6), object(4)
memory usage: 24.7+ MB
```

Changed duration of song from milliseconds to minutes for ease of visualization and renamed the column to ‘duration_min’

```
In [35]: data['duration_ms'] = data['duration_ms'].apply(lambda x:x/60000).round(2)
         data.rename(columns={'duration_ms':'duration_min'},inplace = True)
         data.head()
```

Out[35]:

	valence	year	acousticness	artists	danceability	duration_min	energy	explicit	id	instrumentalness	key	liveness	loudnes
0	0.0594	1921	0.982	['Sergei Rachmaninoff', 'James Levine', 'Berli...	0.279	13.86	0.211	0	4BJqT0PrAfrxzMOxytFOIz	0.878000	10	0.665	-20.0
1	0.9630	1921	0.732	['Dennis Day']	0.819	3.01	0.341	0	7xPhfUan2yNtyFG0cUWkt8	0.000000	7	0.160	-12.4
2	0.0394	1921	0.961	['KHP Kridhamardawa Karaton Ngayogyakarta Hadi...	0.328	8.33	0.166	0	1o6I8BglA6ylDMrIELygv1	0.913000	3	0.101	-14.8
3	0.1650	1921	0.967	['Frank Parker']	0.275	3.50	0.309	0	3ftBPsC5vPBKxYSee08FDH	0.000028	5	0.381	-9.3
4	0.2530	1921	0.957	['Phil Regan']	0.418	2.78	0.193	0	4d6HGyGT8e121BsdKmw9v6	0.000002	3	0.229	-10.0

Outcome Variable:

In this project, we aim to predict the popularity of Spotify songs and analyze how different factors affect the ratings of the song. Hence, the popularity variable will be our outcome variable. By checking the variables available with us, through domain knowledge, we can understand that the variables like id, mode, release date, popularity etc will not be used for popularity prediction. Hence, we have dropped these variables from the dataset. Our outcome variable has also been dropped from the dataset and it will be our target variable and we won't use it to build the prediction model.

```
In [37]: data = data.drop(columns=['id','popularity','release_date','mode'])
         data.head()
```

Out[37]:

	valence	year	acousticness	artists	danceability	duration_min	energy	explicit	instrumentalness	key	liveness	loudness	name	speechiness	
0	0.0594	1921	0.982	['Sergei Rachmaninoff', 'James Levine', 'Berli...	0.279	13.86	0.211	0	0.878000	10	0.665	-20.096	Piano Concerto No. 3 in D Minor, Op. 30: III. ...	0.0366	
1	0.9630	1921	0.732	['Dennis Day']	0.819	3.01	0.341	0	0.000000	7	0.160	-12.441	Clancy Lowered the Boom	0.4150	
2	0.0394	1921	0.961	['KHP Kridhamardawa Karaton Ngayogyakarta Hadi...	0.328	8.33	0.166	0	0.913000	3	0.101	-14.850	Gati Bali	0.0339	1
3	0.1650	1921	0.967	['Frank Parker']	0.275	3.50	0.309	0	0.000028	5	0.381	-9.316	Danny Boy	0.0354	1
4	0.2530	1921	0.957	['Phil Regan']	0.418	2.78	0.193	0	0.000002	3	0.229	-10.096	When Irish Eyes Are Smiling	0.0380	1

The final data frame has been stored in the 'data' dataframe. And there are in total 170k+ rows. We can also see that there are no null values in the dataset.

```
In [41]: data.isnull().sum()
```

```
Out[41]: valence             0
         year                0
         acousticness        0
         artists             0
         danceability        0
         duration_min        0
         energy              0
         explicit            0
         instrumentalness    0
         key                 0
         liveness            0
         loudness            0
         name                0
         speechiness         0
         tempo               0
         dtype: int64
```

Conclusion:

Hence, we have completed data collection and initial processing of the data

Data Exploration and Visualization

Loading necessary libraries

```
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")
```

Reading Data

```
data = pd.read_csv("data.csv")
genre_data = pd.read_csv('data_by_genres.csv')
year_data = pd.read_csv('data_by_year.csv')
```

Correlation between the features

We can see strong correlation between Popularity & Year which makes sense, because a popularity of a song is highly dependent on when it's released and how long ago it was released

Loudness & energy - high correlation between these two variables is seen because its common to perceive loud songs as being acoustic

Acousticness & energy - similar arguments can be made about these two variables as well

```
rs = np.random.RandomState(0)
df = pd.DataFrame(rs.rand(10, 10))
corr = data.corr()
corr.style.background_gradient(cmap='coolwarm', axis = None).set_precision(3)
```

	valence	year	acousticness	danceability	duration_ms	energy	explicit	instrumentalness	key	liveness	loudness	mode	popularity	spe
valence	1.000	-0.028	-0.184	0.559	-0.192	0.354	-0.019	-0.199	0.028	0.004	0.314	0.016	0.014	
year	-0.028	1.000	-0.614	0.189	0.080	0.530	0.221	-0.272	0.008	-0.057	0.488	-0.032	0.862	
acousticness	-0.184	-0.614	1.000	-0.267	-0.076	-0.749	-0.246	0.330	-0.021	-0.024	-0.562	0.047	-0.573	
danceability	0.559	0.189	-0.267	1.000	-0.140	0.222	0.242	-0.278	0.024	-0.100	0.285	-0.046	0.200	
duration_ms	-0.192	0.080	-0.076	-0.140	1.000	0.042	-0.049	0.085	-0.004	0.047	-0.003	-0.046	0.060	
energy	0.354	0.530	-0.749	0.222	0.042	1.000	0.133	-0.281	0.028	0.126	0.782	-0.039	0.485	
explicit	-0.019	0.221	-0.246	0.242	-0.049	0.133	1.000	-0.141	0.005	0.040	0.140	-0.079	0.192	
instrumentalness	-0.199	-0.272	0.330	-0.278	0.085	-0.281	-0.141	1.000	-0.015	-0.047	-0.409	-0.037	-0.297	
key	0.028	0.008	-0.021	0.024	-0.004	0.028	0.005	-0.015	1.000	0.000	0.017	-0.116	0.008	
liveness	0.004	-0.057	-0.024	-0.100	0.047	0.126	0.040	-0.047	0.000	1.000	0.056	0.003	-0.076	
loudness	0.314	0.488	-0.562	0.285	-0.003	0.782	0.140	-0.409	0.017	0.056	1.000	-0.011	0.457	
mode	0.016	-0.032	0.047	-0.046	-0.046	-0.039	-0.079	-0.037	-0.116	0.003	-0.011	1.000	-0.029	
popularity	0.014	0.862	-0.573	0.200	0.060	0.485	0.192	-0.297	0.008	-0.076	0.457	-0.029	1.000	
speechiness	0.046	-0.168	-0.044	0.235	-0.085	-0.071	0.414	-0.122	0.024	0.135	-0.139	-0.058	-0.172	
tempo	0.172	0.141	-0.207	0.002	-0.025	0.251	0.012	-0.105	0.003	0.008	0.210	0.012	0.133	

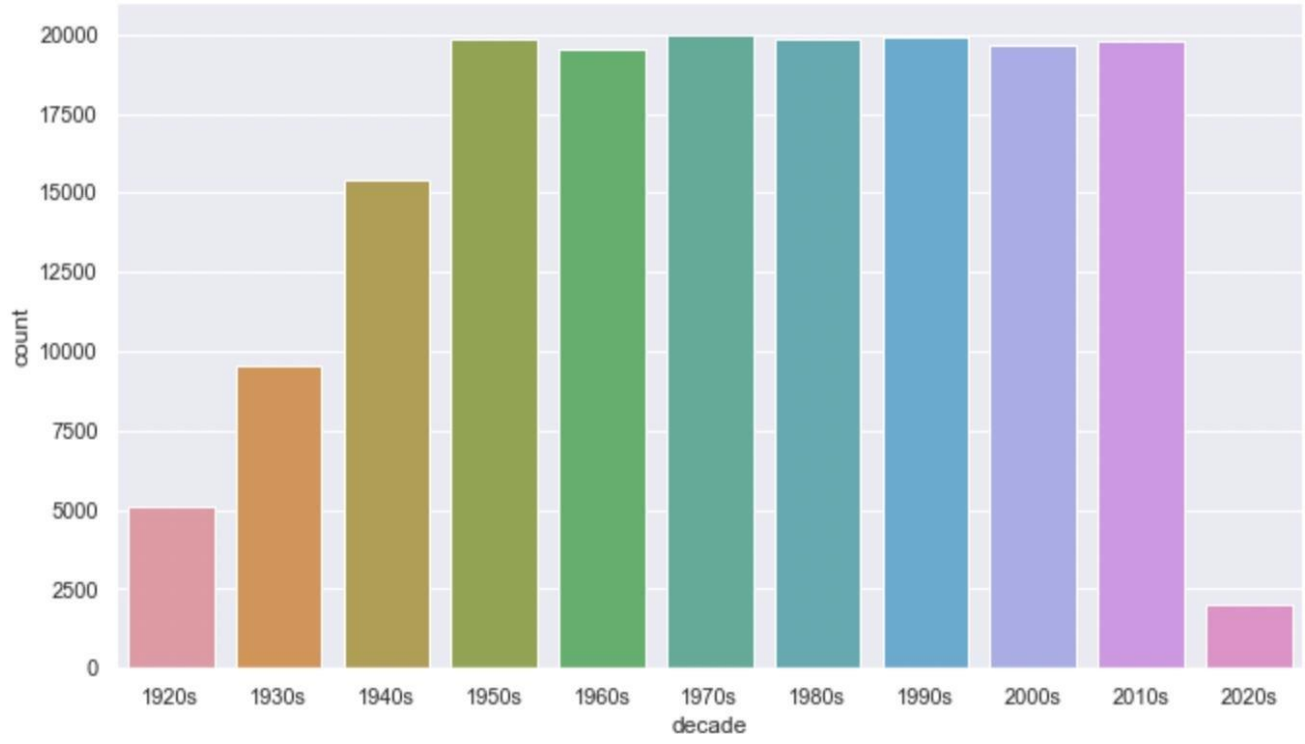
Music Over Time

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

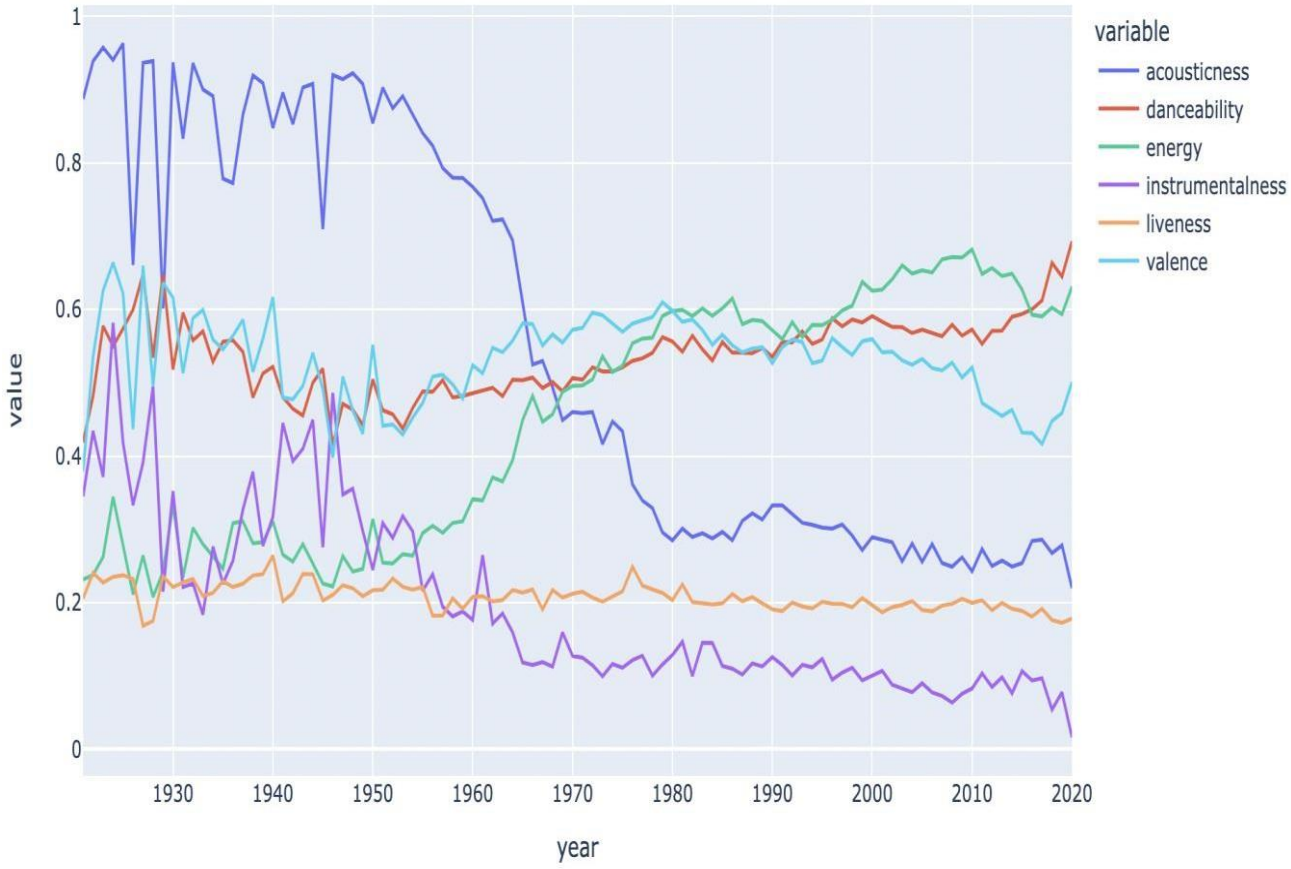
From this bar graph we can see that music industry has started evolving since 1950's and it could be attributed to the invention of tape recorder which happened at the end of 1940's

```
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'])
```

Wrote a function to calculate decade column to group years to show bar graph

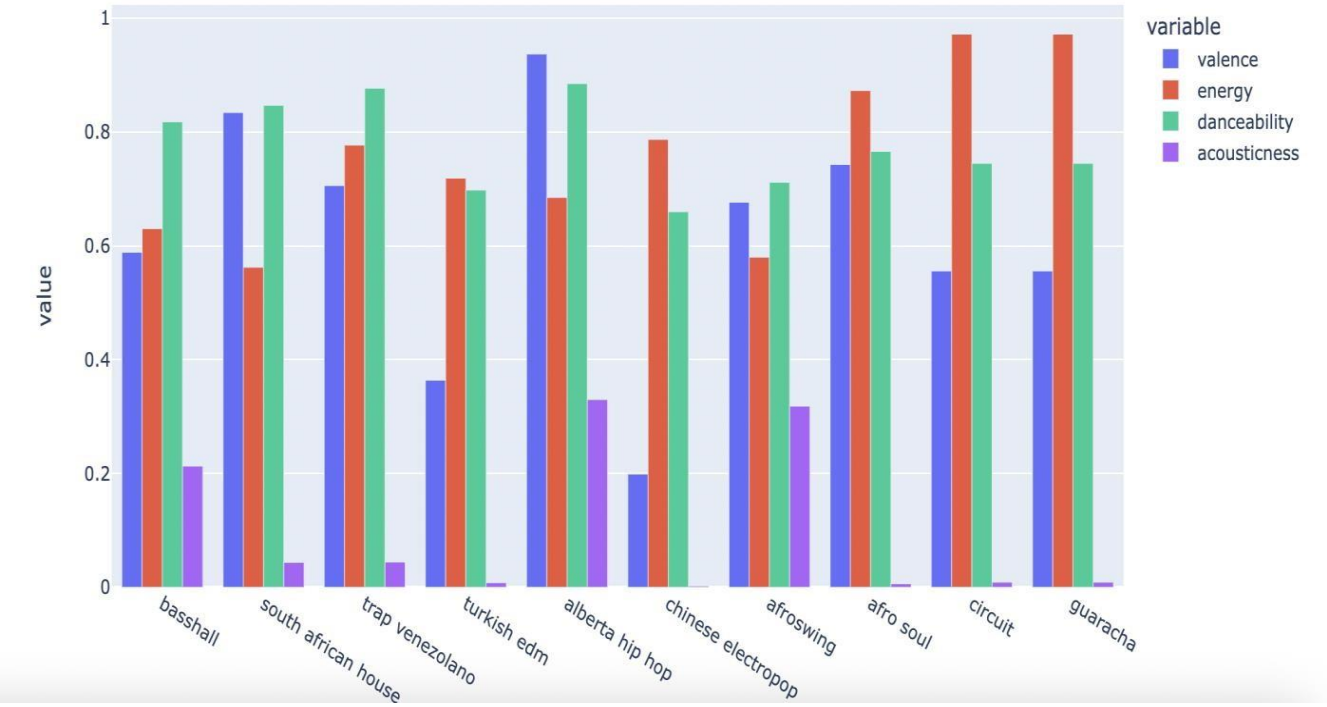



```
sound_features = ['acousticness', 'danceability', 'energy', 'instrumentalness', 'liveness', 'valence']
fig = px.line(year_data, x='year', y=sound_features)
fig.show()
```



From the line graph we can see the trendline increasing for danceability and energy of the songs
And we can observe a sharp decline of trendline for acousticness and intrumentalness
This indicates there is a growing number of ‘catchy’ songs with less focus on instrumentals and acousticness of the song

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

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



Grouped bar chart for top ten genres and its features

This concludes our exploratory data analysis and by using the visualizations we get an idea of different factors which make a song popular at a given time.

Model Exploration & Selection:

For the review dataset, we have to first do sentiment analysis and based on the sentiment score we will be building a collaborative filtering-based recommendation system on the product dataset.
In the sentiment analysis, we try to assign the review to a particular class like 0/1 stating a positive sentiment or negative sentiment based on the summary, voting on reviews (helpfulness) and review text of the customer. As this becomes a binary classification problem, we will be using classification algorithms such as Logistic Regression, Naïve Bayes, Random Forest, XGBoost Classifier and LSTM and comparing them to select the best model.

We convert the text data into a tokenized vector and then use this preprocessed data as an input to our model.

Logistic Regression:

Logistic regression is S-Shaped and bounded function; generally used for binary classification of data. Logistic regression outputs conditional probability value however we find predicted value in regression. Logistic Regression will not predict the exact category your observation should be in but gives you a probability of each observation falls into the category '1'.

Advantages:
1. It is easier to implement, interpret and can be easily extended to multiple classes if needed.
2. Very fast in classifying unknown records.
3. It interprets model coefficients as an indicator of feature importance.

Disadvantages:
1. It requires average or no multicollinearity between independent variables.
2. It assumes that there is linearity between dependent and independent variables.
3. If the number of observations are less than number of features, logistic regressions leads to overfitting the data.

Naïve Bayes Classifier:

Naïve bayes classifier is a probabilistic ML model used to classify data. It is based on Bayes theorem. The major assumption in naïve bayes is that the predictors are independent of each other i.e., the presence of one feature does not affect other. Hence it is called naïve. Another assumption is that all predictors are given equal importance in the output.

Naïve Bayes are mostly used in sentiment analysis, spam filtering and recommendation systems etc.

Advantages:
1. It is simpler and faster compared to exact bayes and easy to implement.
2. It is highly scalable as it scales linearly with number of predictors.
3. It is relatively robust to noisy data.

Disadvantages:
1. The predictors are to be independent of each other. In most of real life examples, the predictors are dependent and this effects the performance of the model.

2. It requires large number of records to obtain reliable parameter estimates.
3. Compared to exact bayes, naïve bayes retains the order of the propensity order but fails to generate the accurate propensity.

Random Forest:

Random forest is an extension of a simple decision tree, the only difference being this algorithm provides the combined result of many such trees, hence the word 'Forest'. A single decision tree looks at all the features by itself to classify the observation. But each tree in the Random Forest model will only look at a randomly selected subset of the complete feature set to conclude, hence the word 'Random'.
What improves the performance of a Random Forest model against a traditional decision tree model is that, by randomly selecting subsets of features, some trees of the forest can isolate more important features while increasing the overall accuracy of the result.

Advantages:
1. It is based on bagging algorithm uses ensemble learning technique.
2. No feature scaling required (standardization & normalization).
3. It is comparatively less impacted by noise and can automatically handle missing values.

Disadvantages:
1. It creates a lot of trees and hence require more computational power and resources for larger datasets.
2. Require more time to train as compared to decision trees.
3. Algorithm computations may go far more complex compared to other algorithms.

XG Boost Classifier:

Boosting is a technique that combines a set of weak learners and delivers improved prediction accuracy. XGBoost is an optimized gradient boosting algorithm known for its speed and unparalleled performance. The library is parallelized, meaning the algorithm can run on a cluster of GPUs or a network of computers. It internally has parameters for cross-validation, regularization, user-defined objective functions, missing values, tree parameters etc.

Advantages:
1. It works well with small data, data with subgroups, big data and complicated data.
2. Less feature engineering is required and less prone to overfitting.
3. When using Scikit learn library, nthread hyper-parameter is used for parallel processing if unspecified, the algorithm will detect automatically the number of CPU cores to be used.

Disadvantages:
1. It doesn't work well with sparse data and dispersed data. It also doesn't perform well with unstructured data.
2. It is very sensitive to outliers and is hardly scalable.
3. If parameters are not tuned properly, overfitting is possible.

Long Short-term Memory:

LSTM is an artificial neural network architecture used in the field of deep learning. It is an updated version of RNN to overcome the vanishing gradient problem. Unlike standard feedforward neural

networks, LSTM has feedback connections. It never keeps the entire data like standard recurrent neural network, LSTM keeps short-term memory of data.

We can summarize the advantages and disadvantages of LSTM cells in 4 main points:

Advantages:

1. They are able to model long-term sequence dependencies.
2. They are more robust to the problem of short memory than 'Vanilla' RNNs since the definition of the internal memory is changed from:

$$h_t = f(Ux_t + Wh_{t-1}) \rightarrow h_t = tanh(s_{t-1}.f_t + g_t.i_t).o_t$$

Disadvantages:

1. They increase the computing complexity compared to the RNN with the introduction of more parameters to learn.
2. The memory required is higher than the one of 'Vanilla' RNNs due to the presence of several memory cells.

For Recommendation system we will be using KNN to perform item based collaborative filtering to recommend top 3 similar items.

K-Nearest Neighbors:

A k-nearest-neighbor algorithm, often abbreviated k-nn, is an approach to data classification that estimates how likely a data point is to be a member of one group or the other depending on what group the data points nearest to it are in.

The k-nearest-neighbor is an example of a "lazy learner" algorithm, meaning that it does not build a model using the training set until a query of the data set is performed.

Advantages:

1. KNN is called Lazy Learner (Instance based learning). It does not learn anything in the training period. It does not derive any discriminative function from the training data. In other words, there is no training period for it. It stores the training dataset and learns from it only at the time of making real time predictions. This makes the KNN algorithm much faster than other algorithms that require training.
2. Since the KNN algorithm requires no training before making predictions, new data can be added seamlessly which will not impact the accuracy of the algorithm.

KNN is very easy to implement. There are only two parameters required to implement KNN i.e., the value of K and the distance function (eg. Euclidean or Manhattan etc.)

Disadvantages:

1. In large datasets, the cost of calculating the distance between the new point and each existing points is huge which degrades the performance of the algorithm.
2. The KNN algorithm doesn't work well with high dimensional data because with large number of dimensions, it becomes difficult for the algorithm to calculate the distance in each dimension.
3. We need to do feature scaling (standardization and normalization) before applying KNN algorithm to any dataset. If we don't do so, KNN may generate wrong predictions.

KNN Item based collaborative filtering:

In collaborative filtering, the items are based on the user's history of choices. This system does not require a large amount of product features to operate. An embedding or feature vector describes each item and User, and it sinks both the items and the users in a similar location. It creates enclosures for items and users on its own.

Advantages:

1. We don't need domain knowledge because the embeddings are automatically learned.
2. The model can help users discover new interests. In isolation, the ML system may not know the user is interested in a given item, but the model might still recommend it becausesimilar users are interested in that item.
3. To some extent, the system needs only the feedback matrix to train a matrix factorization model. In particular, the system doesn't need contextual features. In practice, this can be used as one of multiple candidate generators.

Disadvantages:

1. The prediction of the model for a given (user, item) pair is the dot product of the corresponding embeddings. So, if an item is not seen during training, the system can't create an embedding for it and can't query the model with this item. This issue is often called the cold-start problem.
2. Side features are any features beyond the query or item ID. For movie recommendations,the side features might include country or age. Including available side features improvesthe quality of the model. Although it may not be easy to include side features in WALS, ageneralization of WALS makes this possible.

Model Exploration & Selection:

For the review dataset, we have to first do sentiment analysis and based on the sentiment score we will be building a collaborative filtering-based recommendation system on the product dataset.
In the sentiment analysis, we try to assign the review to a particular class like 0/1 stating a positive sentiment or negative sentiment based on the summary, voting on reviews (helpfulness) and review text of the customer. As this becomes a binary classification problem, we will be using classification algorithms such as Logistic Regression, Naïve Bayes, Random Forest, XGBoost Classifier and LSTM and comparing them to select the best model.

We convert the text data into a tokenized vector and then use this preprocessed data as an input to our model.

Logistic Regression:

Logistic regression is S-Shaped and bounded function; generally used for binary classification of data. Logistic regression outputs conditional probability value however we find predicted value in regression. Logistic Regression will not predict the exact category your observation should be in but gives you a probability of each observation falls into the category '1'.

Advantages:
1. It is easier to implement, interpret and can be easily extended to multiple classes if needed.
2. Very fast in classifying unknown records.
3. It interprets model coefficients as an indicator of feature importance.

Disadvantages:
1. It requires average or no multicollinearity between independent variables.
2. It assumes that there is linearity between dependent and independent variables.
3. If the number of observations are less than number of features, logistic regressions leads to overfitting the data.

Naïve Bayes Classifier:

Naïve bayes classifier is a probabilistic ML model used to classify data. It is based on Bayes theorem. The major assumption in naïve bayes is that the predictors are independent of each other i.e., the presence of one feature does not affect other. Hence it is called naïve. Another assumption is that all predictors are given equal importance in the output.

Naïve Bayes are mostly used in sentiment analysis, spam filtering and recommendation systems etc.

Advantages:
1. It is simpler and faster compared to exact bayes and easy to implement.
2. It is highly scalable as it scales linearly with number of predictors.
3. It is relatively robust to noisy data.

Disadvantages:
1. The predictors are to be independent of each other. In most of real life examples, the predictors are dependent and this effects the performance of the model.
2. It requires large number of records to obtain reliable parameter estimates.

3. Compared to exact bayes, naïve bayes retains the order of the propensity order but fails to generate the accurate propensity.

Random Forest:

Random forest is an extension of a simple decision tree, the only difference being this algorithm provides the combined result of many such trees, hence the word 'Forest'. A single decision tree looks at all the features by itself to classify the observation. But each tree in the Random Forest model will only look at a randomly selected subset of the complete feature set to conclude, hence the word 'Random'.
What improves the performance of a Random Forest model against a traditional decision tree model is that, by randomly selecting subsets of features, some trees of the forest can isolate more important features while increasing the overall accuracy of the result.

Advantages:
1. It is based on bagging algorithm uses ensemble learning technique.
2. No feature scaling required (standardization & normalization).
3. It is comparatively less impacted by noise and can automatically handle missing values.

Disadvantages:
1. It creates a lot of trees and hence require more computational power and resources for larger datasets.
2. Require more time to train as compared to decision trees.
3. Algorithm computations may go far more complex compared to other algorithms.

XG Boost Classifier:

Boosting is a technique that combines a set of weak learners and delivers improved prediction accuracy. XGBoost is an optimized gradient boosting algorithm known for its speed and unparalleled performance. The library is parallelized, meaning the algorithm can run on a cluster of GPUs or a network of computers. It internally has parameters for cross-validation, regularization, user-defined objective functions, missing values, tree parameters etc.

Advantages:
1. It works well with small data, data with subgroups, big data and complicated data.
2. Less feature engineering is required and less prone to overfitting.
3. When using Scikit learn library, nthread hyper-parameter is used for parallel processing if unspecified, the algorithm will detect automatically the number of CPU cores to be used.

Disadvantages:
1. It doesn't work well with sparse data and dispersed data. It also doesn't perform well with unstructured data.
2. It is very sensitive to outliers and is hardly scalable.
3. If parameters are not tuned properly, overfitting is possible.

Long Short-term Memory:

LSTM is an artificial neural network architecture used in the field of deep learning. It is an updated version of RNN to overcome the vanishing gradient problem. Unlike standard feedforward neural networks, LSTM has feedback connections. It never keeps the entire data like standard recurrent

neural network, LSTM keeps short-term memory of data.

We can summarize the advantages and disadvantages of LSTM cells in 4 main points:

Advantages:

1. They are able to model long-term sequence dependencies.
2. They are more robust to the problem of short memory than 'Vanilla' RNNs since the definition of the internal memory is changed from:

$$h_t = f(Ux_t + Wh_{t-1}) \rightarrow h_t = tanh(s_{t-1}.f_t + g_t.i_t).o_t$$

Disadvantages:

1. They increase the computing complexity compared to the RNN with the introduction of more parameters to learn.
2. The memory required is higher than the one of 'Vanilla' RNNs due to the presence of several memory cells.

For Recommendation system we will be using KNN to perform item based collaborative filtering to recommend top 3 similar items.

K-Nearest Neighbors:

A k-nearest-neighbor algorithm, often abbreviated k-nn, is an approach to data classification that estimates how likely a data point is to be a member of one group or the other depending on what group the data points nearest to it are in.

The k-nearest-neighbor is an example of a "lazy learner" algorithm, meaning that it does not build a model using the training set until a query of the data set is performed.

Advantages:

1. KNN is called Lazy Learner (Instance based learning). It does not learn anything in the training period. It does not derive any discriminative function from the training data. In other words, there is no training period for it. It stores the training dataset and learns from it only at the time of making real time predictions. This makes the KNN algorithm much faster than other algorithms that require training.
2. Since the KNN algorithm requires no training before making predictions, new data can be added seamlessly which will not impact the accuracy of the algorithm.

KNN is very easy to implement. There are only two parameters required to implement KNN i.e., the value of K and the distance function (eg. Euclidean or Manhattan etc.)

Disadvantages:

1. In large datasets, the cost of calculating the distance between the new point and each existing points is huge which degrades the performance of the algorithm.
2. The KNN algorithm doesn't work well with high dimensional data because with large number of dimensions, it becomes difficult for the algorithm to calculate the distance in each dimension.
3. We need to do feature scaling (standardization and normalization) before applying KNN algorithm to any dataset. If we don't do so, KNN may generate wrong predictions.

KNN Item based collaborative filtering:

In collaborative filtering, the items are based on the user's history of choices. This system does not require a large amount of product features to operate. An embedding or feature vector describes each item and User, and it sinks both the items and the users in a similar location. It creates enclosures for items and users on its own.

Advantages:

1. We don't need domain knowledge because the embeddings are automatically learned.
2. The model can help users discover new interests. In isolation, the ML system may not know the user is interested in a given item, but the model might still recommend it becausesimilar users are interested in that item.
3. To some extent, the system needs only the feedback matrix to train a matrix factorizationmodel. In particular, the system doesn't need contextual features. In practice, this can be used as one of multiple candidate generators.

Disadvantages:

1. The prediction of the model for a given (user, item) pair is the dot product of the corresponding embeddings. So, if an item is not seen during training, the system can't create an embedding for it and can't query the model with this item. This issue is often called the cold-start problem.
2. Side features are any features beyond the query or item ID. For movie recommendations,the side features might include country or age. Including available side features improvesthe quality of the model. Although it may not be easy to include side features in WALS, ageneralization of WALS makes this possible.

Model Implementation:

Using the preprocessed review dataset, we have only selected songs that have total reviewers greater than 100. On top of this data, computed the average rating of each song and merged all the reviews of each song id into a single list and added it back to the dataframe. All the duplicate rows were deleted based on song id. The review data was then vectorized using count vectorizer and the data was then split into train and test data. The training data was then fed to a KNN model with parameters k set to 10 and algorithm = ball_tree.

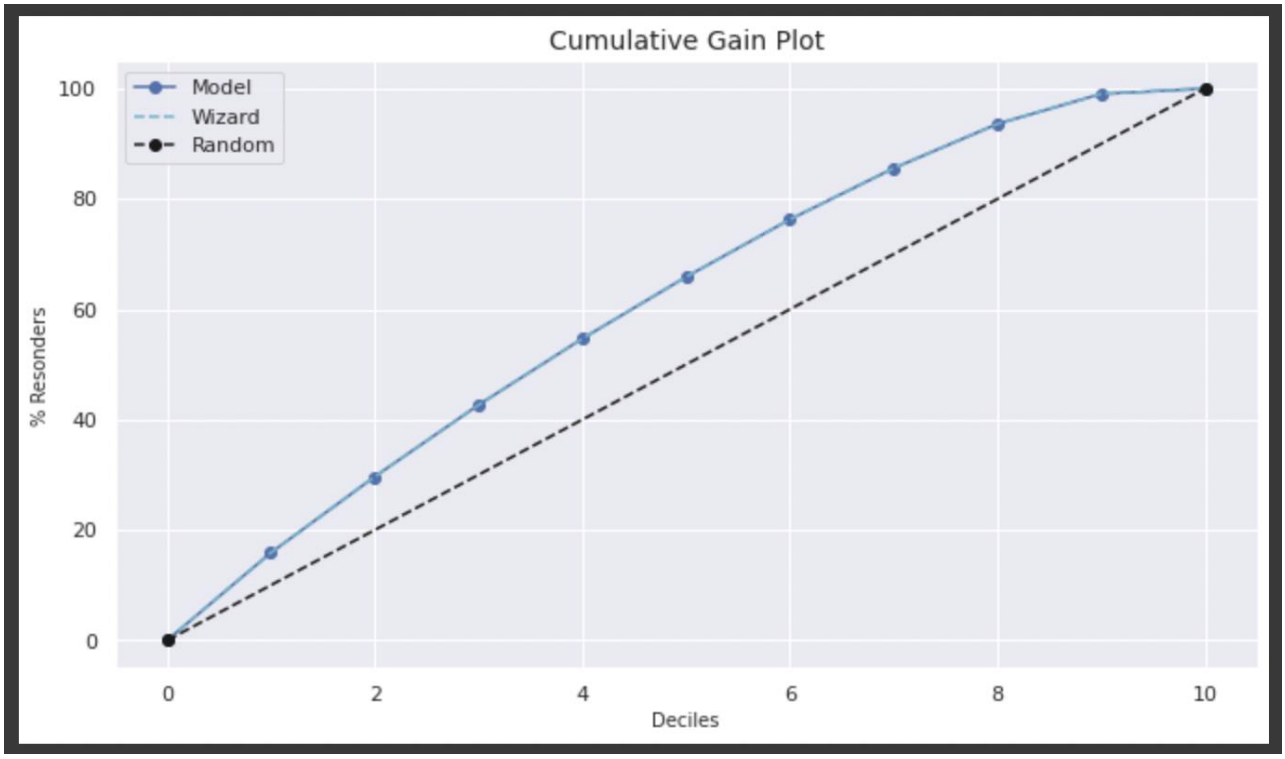
```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 2973 samples in 0.010s...
[t-SNE] Computed neighbors for 2973 samples in 0.209s...
[t-SNE] Computed conditional probabilities for sample 1000 / 2973
[t-SNE] Computed conditional probabilities for sample 2000 / 2973
[t-SNE] Computed conditional probabilities for sample 2973 / 2973
[t-SNE] Mean sigma: 0.777516
[t-SNE] KL divergence after 250 iterations with early exaggeration: 76.114395
[t-SNE] KL divergence after 1000 iterations: 1.392500
```

Performance Evaluation

1) Linear Regression Model

For Linear Regression, the graph fits well with the prediction, and the r2 is prefect 1 with both rmse and mae scores low. This may be the best model for our dataset.

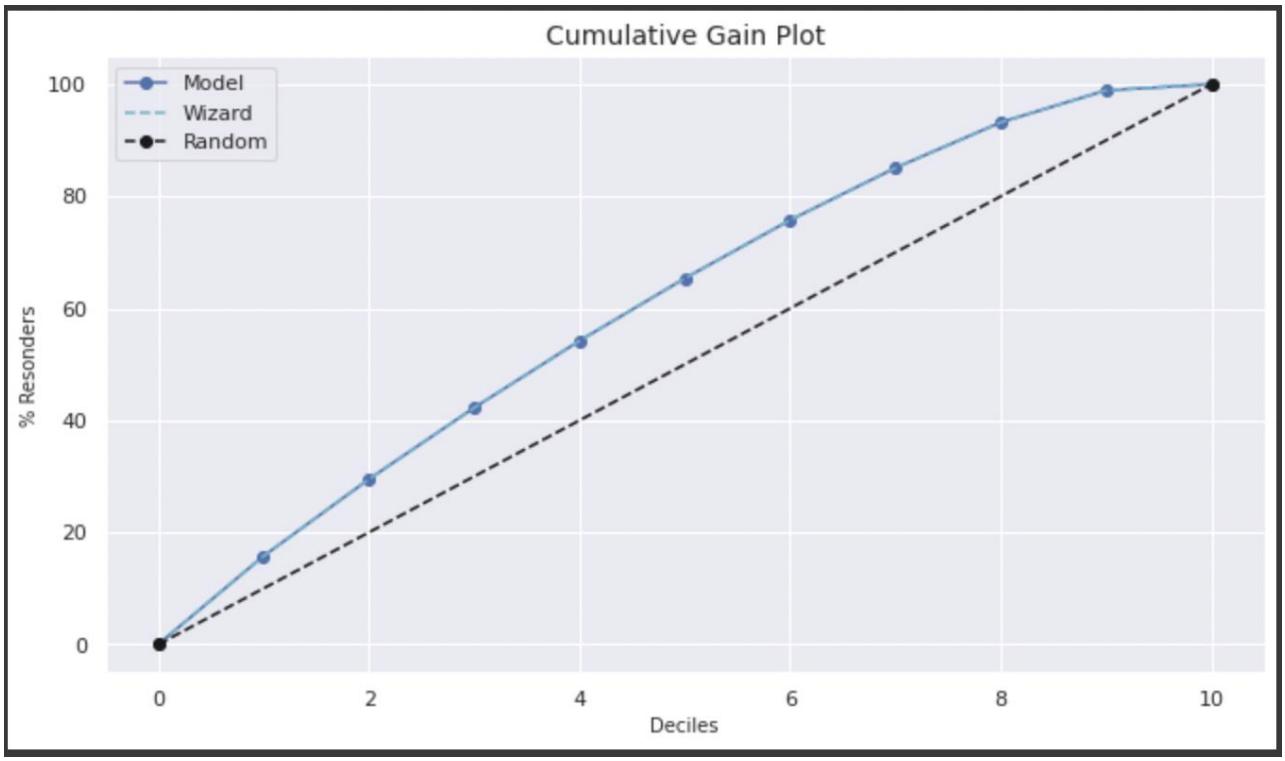
```
Mean absolute error is :  8.045477396354241e-15
RMSE:  1.0483044159639842e-14
MSE is  1.098942148529 59e-28
R2 score is 1.0
```



2) Polynomial Regression Model

For Polynomial Regression model, we can see higher rmse and mae values than that of Linear regression and high mean absolute error. For the prediction model, we will comparethe scores with a graph to see which prediction model has lowest rmse.

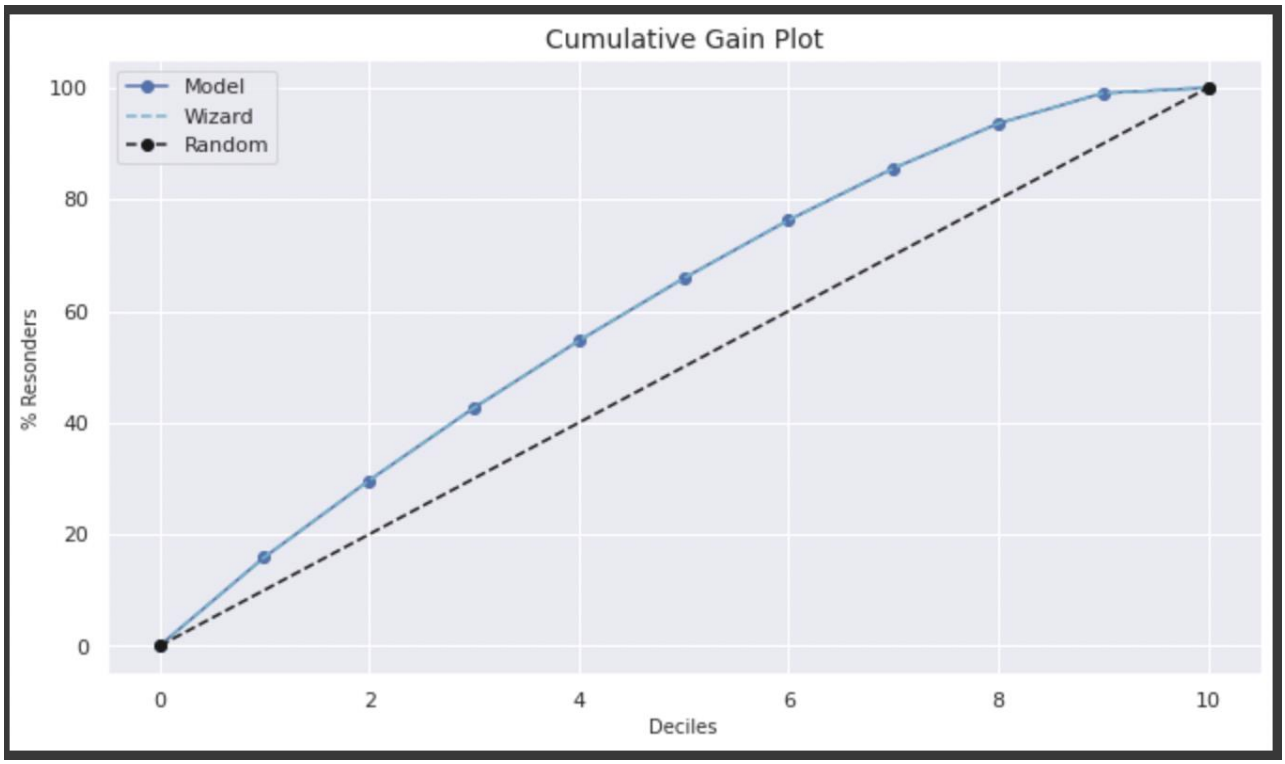
```
Mean absolute error is :  5.7153116831806646e-05
RMSE:  8.222007161783l9e-05
MSE is  6.760140176841406e-09
R2 score is 0.9999999999750321
```



3) Lasso Regression Model

For Lasso Regression Model, after normalization, we can see very low scores of rmse andmae, but still higher than the above two models.

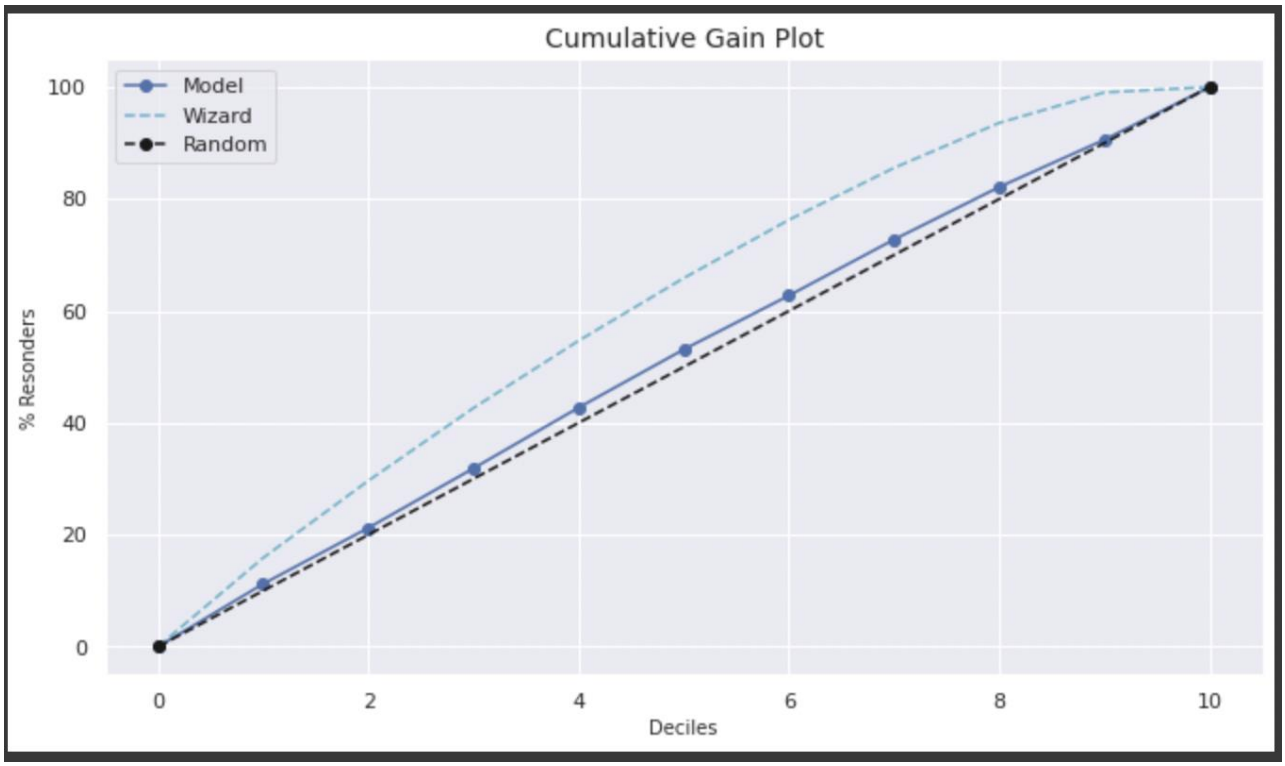
```
Mean absolute error is :  0.7841299425755205
RMSE:  1.0015376218892358
MSE is  1.0030776080595458
R2 score is 0.9959608007291691
```



4) KNN Regression Model

For KNN Regression Model, we determined the optimal no of K using rmse scores and evaluated the model on this K. We got high rmse and mae scores compared to the above models.

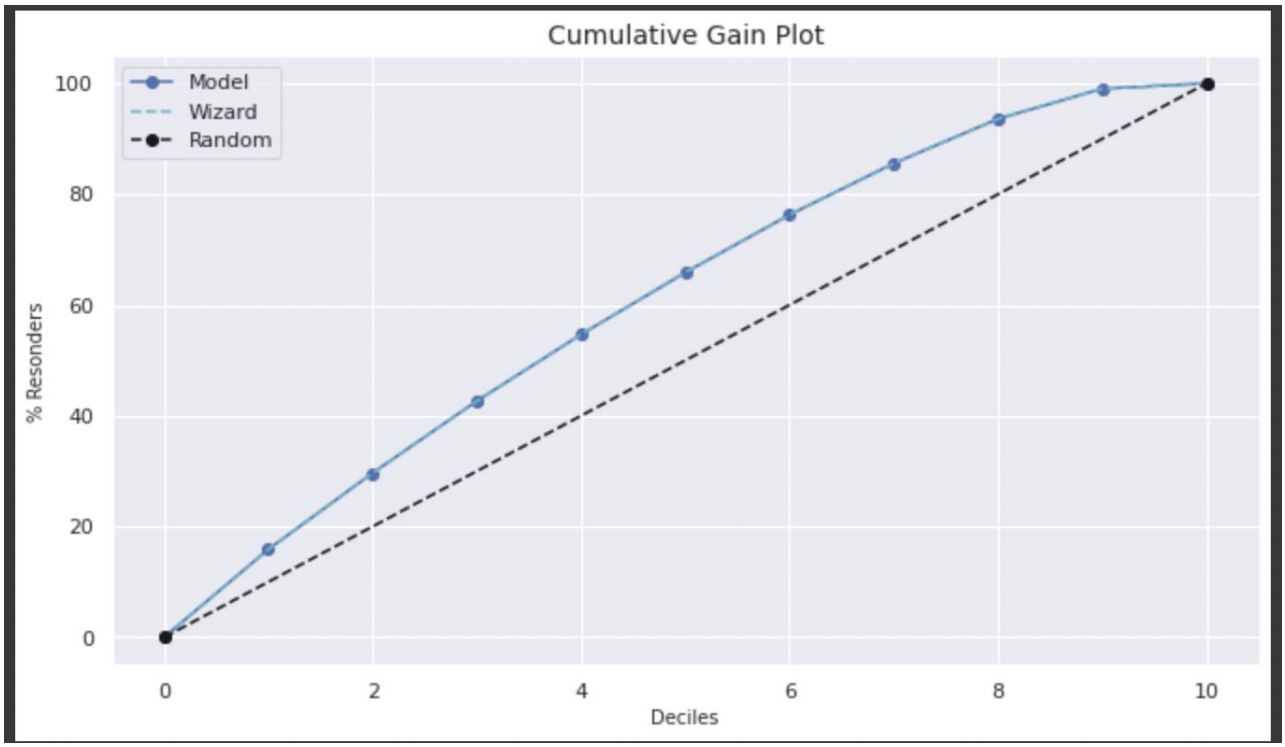
```
Mean absolute error is :  13.236338776859949
RMSE:  17.22283955115392
MSE is  296.6262022047918
R2 score is -4.748406022859824
```



5) Random Forest Regression Model

For Random Forest Regression Model, we can see very low values of rmse and mae, but notas low as Linear and Polynomial regression models. This could be also the result of overfitting which is common with the model.

```
Mean absolute error is :  0.0258009368454102 52
RMSE:  0.06841506347039769
MSE is  0.0046806209096585 44
R2 score is 0.9999833271441884
```



Conclusion:

After evaluating performances of all the models of our dataset, we conclude that Linear regression model fits our data well and predicts popularity of songs with very less errors. In this project we have seen how various parameters effect the results of the model and how to choose a model which fits our dataset.

For future work on the project, we propose to use this prediction of the rating of the song to build a song recommendation system using the user's listening history, association rules and collaborative filtering models.