# Need new music? Let data do that for you!

by Corbin Couger

8/30/2023

Course: DSC680

Bellevue University

## Introduction

(proposal, Milestone 1)

Have you recently or in the past been finding yourself craving new tunes to listen to? Like something new, but also similar to the music you like? With so many songs being added to streaming services daily, it is a difficult task to find new artists and songs that are you or your buisness's taste! Even for businesses, music can be a big part of the shopping experience as the vibe of the music must be consistent and draw in more customers. I personally have been in plenty of stores/restaurants where the music made me want to leave and go elsewhere. Enjoying the music of a business will drive a customer to enjoy being at the business longer. So, whether it is your current personal music needing a refresh, or you are a business seeking some automated help in crafting the perfect playlist, look no further!

This project's goal is to create a recommender system using Clustering methods on a spotify song dataset. The end product is an application where a user will input a song (the song you're vibing with the most right now) and then it will output some songs that are statistically similar to the input. This makes the process of finding new music for yourself or business a breeze and it also will (hopefully) do it with precision.

For a project like this, I would typically want to use an API for capturing and maintaining an up-to-date and consistent dataflow, but I am opting to use this really cool dataset I found on Kaggle which is an extraction of a Spotify API. The link for the dataset is located here (https://www.kaggle.com/datasets/mrmorj/dataset-of-songs-in-spotify?select=genres_v2.csv (https://www.kaggle.com/datasets/mrmorj/dataset-of-songs-in-spotify?select=genres_v2.csv)) and it is what I will be using to craft this recommender system.

#### The Data

In [8]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from kneed import KneeLocator
```

```
C:\Users\corbi\anaconda3\lib\site-packages\numpy\_distributor_init.py:
30: UserWarning: loaded more than 1 DLL from .libs:
C:\Users\corbi\anaconda3\lib\site-packages\numpy\.libs\libopenblas.XWY
DX2IKJW2NMTWSFYNGFUWKQU3LYTCZ.gfortran-win_amd64.dll
C:\Users\corbi\anaconda3\lib\site-packages\numpy\.libs\libopenblas64__
v0.3.21-gcc_10_3_0.dll
  warnings.warn("loaded more than 1 DLL from .libs:"
```

In [107]:
```
genres = pd.read_csv('genres_v2.csv')
genres.head(3)
```

```
C:\Users\corbi\AppData\Local\Temp\ipykernel_22040\4256403819.py:1: Dty
peWarning: Columns (19) have mixed types. Specify dtype option on impo
rt or set low_memory=False.
  genres = pd.read_csv('genres_v2.csv')
```

Out[107]:

| | danceability | energy | key | loudness | mode | speechiness | acousticness | instrumentaln |
|---|---|---|---|---|---|---|---|---|
| **0** | 0.831 | 0.814 | 2 | -7.364 | 1 | 0.4200 | 0.0598 | 0.013 |
| **1** | 0.719 | 0.493 | 8 | -7.230 | 1 | 0.0794 | 0.4010 | 0.000 |
| **2** | 0.850 | 0.893 | 5 | -4.783 | 1 | 0.0623 | 0.0138 | 0.000 |

3 rows × 22 columns

In [108]:
```
genres.dtypes
```

Out[108]:
```
danceability        float64
energy              float64
key                   int64
loudness            float64
mode                  int64
speechiness         float64
acousticness        float64
instrumentalness    float64
liveness            float64
valence             float64
tempo               float64
type                 object
id                   object
uri                  object
track_href           object
analysis_url         object
duration_ms           int64
time_signature        int64
genre                object
song_name            object
Unnamed: 0          float64
title                object
dtype: object
```

Initially, looking at the dataset I'm looking at the first three rows and the data types for each variable can tell me a lot. Each of these columns have a rating or value that gives each song its characteristics.

In [109]:
```
print(genres.shape[0], 'rows', genres.shape[1], 'columns')
```

```
42305 rows 22 columns
```

The data contains 42k songs and about 20 different characteristics for each song. A few columns will need to be removed and cleaned up before I can get into any testing, but that will happen in a later section.

### Methods

The source of this data is awesome and I'm very pleased it is a sample dataset from an API pull. I know this data can be used to make a cool, data driven, recommender system. The method I will mainly be using is a clustering technique called K-means clustering. This is a powerful, mathematically driven method that will help 'cluster' different songs together that are similar. This method typically produces results that are not easy to catch when viewing data, K-means finds hidden patterns and groups within the data that can be impactful for you or your business.

K-means clustering will help solve the personal or business problem when it comes to finding the right music and using this method to drive a recommender system makes it all more simple.

#### Ethical Considerations

When considering my outline and vision for this product, I do want to address any ethical considerations that come to mind. I know this recommender system might not be something that could introduce those concerns, but it I do want to address some. My main focus of the project will be to make accurate recommendations. I wouldn't want a family friendly restaurant to get recommended explicit music to play. I want the use and satisfaction of the system to be positive. On a not-so-ethical consideration, the music should be checked by the business to not interfere with any copyright laws. I'm not an expert with copyrighting and this would be something I would make sure to prompt a business with. Besides these things, I do not (initially) sense any other ethical considerations for this project. If more come up along the way I will address them then.

#### Challenges/Issues

Other than the issues I mentioned above, this will be my first time attempting to creat an application out of my python code. Taking it out of an IDE environment will most likely be my biggest challenge. Another challenge will be using the outputs from my K-means clustering and working them into the recommender system. I have not had much experience creating recommender systems, but I do have experience with clustering, so that should hopefully go smooth.

#### Next Steps

Now that I have given a quick background on the business problem I'm going to solve, as well as the; methods, ethical considerations, and challenges described, this project is ready

## Preliminary Analysis

(week 2, milestone 2)

In this beginning stage of the analysis, I want to make sure the data is suitable for Cluster Analysis. This first requires checking and cleaning the data.

#### Data Cleaning

I will first remove the columns that are either 'unnamed' or redundant. These will include the 'Unname : 0', 'title', and 'type' columns.

```
In [110]: genres = genres.drop(columns=['Unnamed: 0', 'title', 'type'])
```

Now with those colums removed I will check for NULL (NA) values

In [112]:
```
genres.dropna().shape
```

Out[112]:
```
(21519, 19)
```

In [113]:
```
genres.shape
```

Out[113]:
```
(42305, 19)
```

In [114]:
```
genres.isnull().any()
```

Out[114]:
```
danceability        False
energy              False
key                 False
loudness            False
mode                False
speechiness         False
acousticness        False
instrumentalness    False
liveness            False
valence             False
tempo               False
id                  False
uri                 False
track_href          False
analysis_url        False
duration_ms         False
time_signature      False
genre               False
song_name            True
dtype: bool
```

Looking at this situation with the missing values, there seems to be about half of the song titles being blank. This isn't great news because all the other data is there for these missing values, but the title being a main data point for the recommendation, I want it to be present. So, I will make the decision to remove the missing values.

In [115]:
```
genres = genres.dropna()
```

Now that the missing values are removed, I would typically at this point check for outliers. Since K-means clustering is such a powerful method, there outliers get automatically filtered into what is called noise. With that being said, there is one last thing to do in preliminary analysis, converting all non-numerical variables to numeric. This is because traditional K-means clustering needs numerical variables to mathematically cluster the data.

With further thought on this next step, I know I will be using a built-in Pandas trick (get_dummies) which turns my object/categorical variables into dummy numeric variables. There is one thing however, doing this creates numerous new columns based on the different categories. Since columns such as 'id' (which has letters in it) and 'url' are unique to each row, they technically aren't categories. So, I will create a new numeric dataframe just containing the numeric columns and actual categorical variable(s). Then at the end, I will connect the cluster results back to the original data.

In [116]:
```
genres_n = genres.drop(columns = ['id', 'uri', 'track_href', 'analysis_
```

In [117]:
```
genres_n.dtypes
```

Out[117]:
```
danceability        float64
energy              float64
key                   int64
loudness            float64
mode                  int64
speechiness         float64
acousticness        float64
instrumentalness    float64
liveness            float64
valence             float64
tempo               float64
duration_ms           int64
time_signature        int64
genre                object
dtype: object
```

All there is to do now is convert the 'genre' column to a dummy

In [118]:
```
genres_n = pd.get_dummies(genres_n)
```

In [120]:
```
genres_n.head()
```

Out[120]:

| ness | valence | ... | duration_ms | time_signature | genre_Dark Trap | genre_Emo | genre_Hiphop | g |
|---|---|---|---|---|---|---|---|---|
| 0556 | 0.3890 | ... | 124539 | 4 | 1 | 0 | 0 | |
| 1180 | 0.1240 | ... | 224427 | 4 | 1 | 0 | 0 | |
| 3720 | 0.0391 | ... | 98821 | 4 | 1 | 0 | 0 | |
| 1140 | 0.1750 | ... | 123661 | 3 | 1 | 0 | 0 | |
| 1660 | 0.5910 | ... | 123298 | 4 | 1 | 0 | 0 | |

Something I'm noticing is there are very limited genres. This is a slight concern as it reduces the number of businesses/people I can reach, but thankfully they are also some very popular genres of music that many songs today are classified as. Finally, I'm going to standardize this data so that the results I get from the cluster analysis, make sense mathematically. Basically, this makes all the numerical variables in my dataset standardized to the form where the results are interpretable.

```
In [121]: scaled_genres = StandardScaler().fit_transform(genres_n)
```

With all these steps finished, I am now ready to create a clustering model.

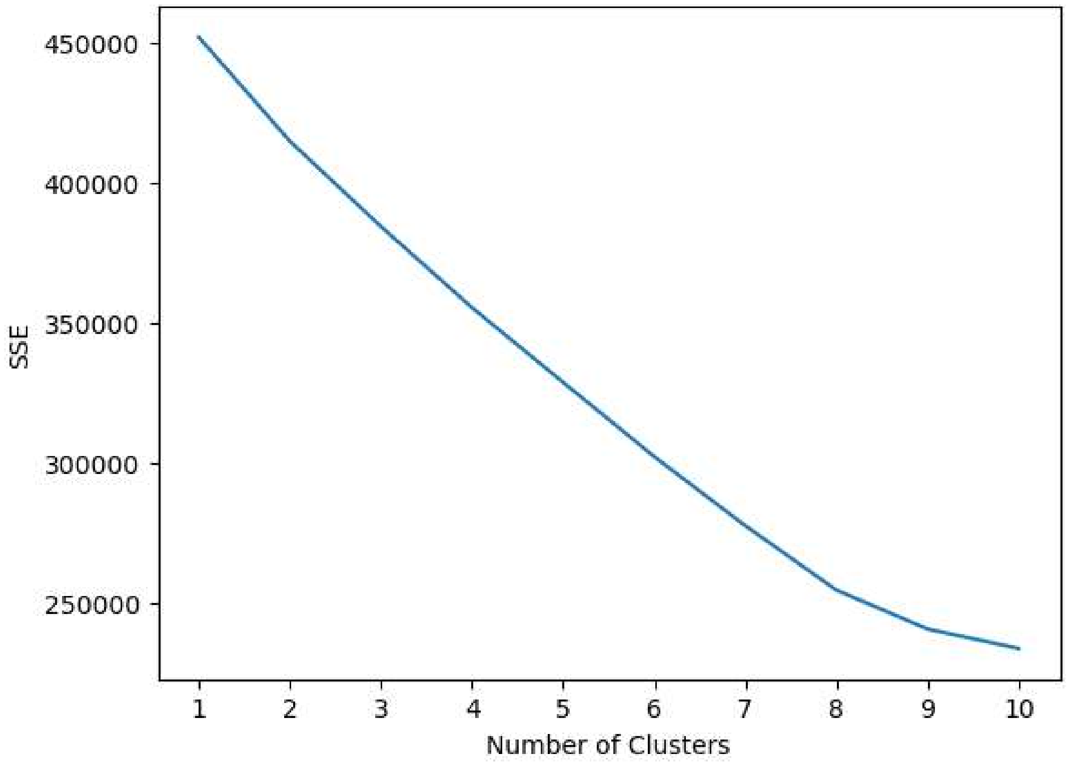
## K-means Clustering

With preliminary analysis done, I can now begin to create the model. To put in simple terms and give background on this type of analysis, K-Means Clustering is an analytical clustering method. Clustering pretty much uses statistics to group data points together. In the case of K-means, the statistics use a specified number of clusters (k). Each k gets a mathematical distance, sort of like an atmosphere, and which ever datapoints fall into that distance are apart of that cluster. I put this super simply and it is much more powerful and complex than this description, but that is pretty much what is going on here. So, I need to first find that specified k.

#### Optimal Number of Clusters

To ensure that I create a solid model, I need to find what the number (k) of clusters do I need to optimize this model. I will do this by creating a plot of the SSE for each K.

In [122]:
```
# this setup below is pretty universal code that helps calculate the SS
kmeans_kwargs = {
"init": "random",
"n_init": 10,
"random_state": 1,
}

sse = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, **kmeans_kwargs)
    kmeans.fit(scaled_genres)
    sse.append(kmeans.inertia_)

#visualize results
plt.plot(range(1, 11), sse)
plt.xticks(range(1, 11))
plt.xlabel("Number of Clusters")
plt.ylabel("SSE")
plt.show()
```



Usually I'm looking for a knee in the SSE plot, but it really is hard to see one, my best guess would be k equaling 3 or 8. I will use a helpful tool that helps locate the knee.

In [123]:
```
kl = KneeLocator(range(1, 11), sse, curve="convex", direction="decreasi
kl.elbow
```

Out[123]:
```
7
```

So, turns out I was a little off, and the right k value was in between 3 and 8. I will use k = 7 for my cluster analysis.

```
In [124]: kmeans = KMeans(init="random", n_clusters=7, n_init=10, random_state=1)
          kmeans.fit(scaled_genres)
```

```
Out[124]: KMeans(init='random', n_clusters=7, random_state=1)
```

Creating the model is actually the easiest part, now I will check how the clusters turned out and attach them to the data.

```
In [125]: genres['cluster'] = kmeans.labels_
```

```
In [126]: genres.groupby(['cluster'])['cluster'].count()
```

```
Out[126]: cluster
          0    1954
          1    1848
          2    4602
          3    5864
          4    2099
          5    1675
          6    3477
          Name: cluster, dtype: int64
```

```
In [127]: #creating a new .csv datafile to store the results
          genres.to_csv('songclusterresults.csv', encoding='utf-8', index=False)
```

## Analysis Conclusion

Looking at the cluster split, I'm pretty pleased. It seems that each cluster has about 2000 different songs within them. This should give the user/business plenty of accuarte options and not run out of music for some time!

I will now begin on the building of the recommender system using this new data set. For this I want to keep it simple, whatever song the user imputs, I will display 100 random songs from that same cluster. This should hopefully be simple enough since I've already done the complexity of filtering each song into a cluster.

The data I want to present back to the user is the Song Name, URL, Energy, and Loudness just to give the user/business some slight information on the song.

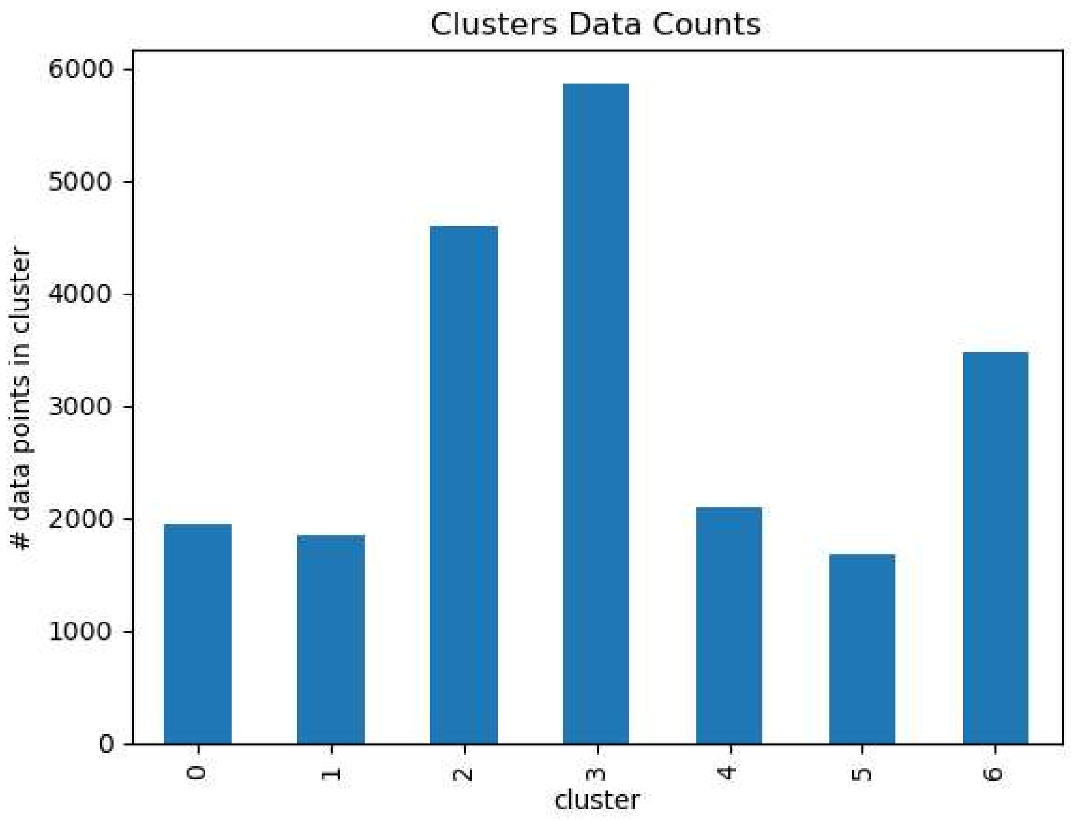
In [37]:
```
results = pd.read_csv('songclusterresults.csv')
# I will also clean up the song title to lowercase and stripped of lead
results['song_name'] = results['song_name'].str.lower()
results['song_name'] = results['song_name'].str.strip()
results.head()
```

Out[37]:

| id | uri | track_href | |
|---|---|---|---|
| Kx | spotify:track:2Vc6NJ9PW9gD9q343XFRKx | https://api.spotify.com/v1/tracks/2Vc6NJ9PW9gD... | h |
| 6p | spotify:track:7pgJBLVz5VmnL7uGHmRj6p | https://api.spotify.com/v1/tracks/7pgJBLVz5Vmn... | h |
| hy | spotify:track:0vSWgAlfpye0WCGeNmuNhy | https://api.spotify.com/v1/tracks/0vSWgAlfpye0... | h |
| nu | spotify:track:0VSXnJqQkwuH2ei1nOQ1nu | https://api.spotify.com/v1/tracks/0VSXnJqQkwuH... | h |
| S3 | spotify:track:4jCeguq9rMTlbMmPHuO7S3 | https://api.spotify.com/v1/tracks/4jCeguq9rMTl... | h |

I will see how the clusters are looking first.

In [98]:
```python
results.groupby(['cluster'])['cluster'].count().plot(kind = 'bar')
plt.title('Clusters Data Counts')
plt.ylabel('# data points in cluster')
plt.show()
```



## Recommender System

Now that I have run my Cluster Analysis, I can take my results and create a recommendation system with them. I'm going to keep it really simple since the K-Means did a lot of the work previously. Like I mentioned 100 raondom songs from each cluster will be presented back to the user with specific data points on the song.

After loading in the data, the code below will be the code that would actually make it into the application.

In [69]:
```python
def user_song():
    song = input('Enter a song you would like to find recommendations fo
    song = song.strip().lower()
    #print(song)
    r = results.loc[results['song_name'] == song]
    #print(r)
    return r
```

In [70]:
```
while True:
    song_data = user_song()
    if song_data.empty is True:
        print('Your song is unfortunately not in the data, or you misspe
        continue
    elif song_data.empty is False:
        #print(song_data)
        break
```

```
Enter a song you would like to find recommendations for: not a song ti
tle
Your song is unfortunately not in the data, or you misspelled. Try aga
in!
Enter a song you would like to find recommendations for: pathology
```

In [74]:
```
song_cluster = song_data['cluster'][1]
```

```
2
```

In [76]:
```
recommendations_data = results.loc[results['cluster'] == song_cluster]
```

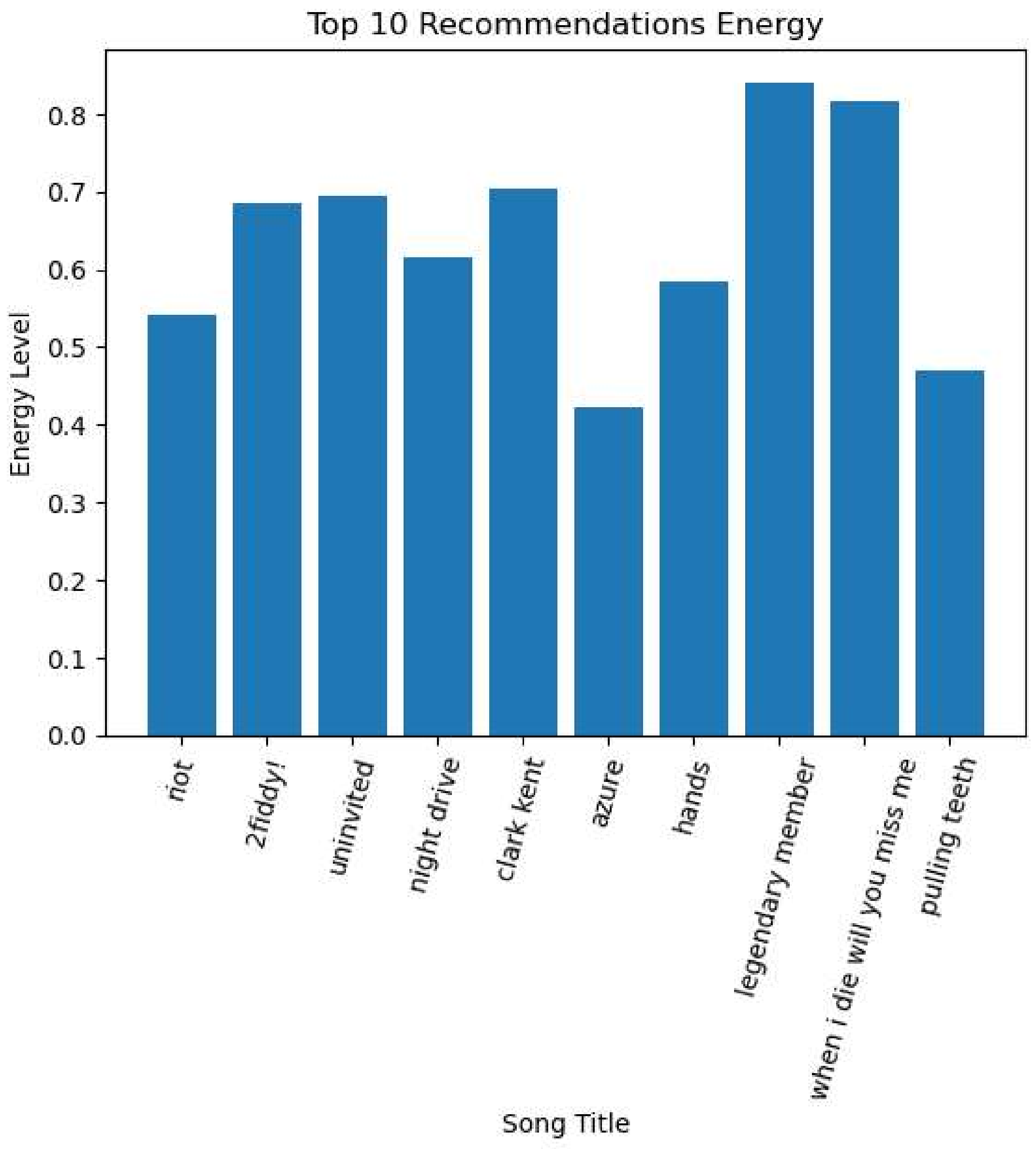
In [100]:
```
recomm = recommendations_data[['song_name', 'genre', 'uri', 'energy', '
```

In [101]:
```
print('Your song recommendations are:')
for index, song in recomm.iterrows():
    print('\n',
          'Song Title:', song['song_name'], '\n',
          'Genre:', song['genre'], '\n',
          'Song URL:', song['uri'], '\n',
          'Song Energy:', song['energy'], '\n',
          'Song Loudness:', song['loudness'], '\n',
          '________________________________________________________________
```

```
Your song recommendations are:

 Song Title: riot
 Genre: Dark Trap
 Song URL: spotify:track:4aOOExMBUyxKnEYb39SrTg
 Song Energy: 0.542
 Song Loudness: -8.027000000000001
 ________________________________________________________________
__

 Song Title: 2fiddy!
 Genre: Dark Trap
 Song URL: spotify:track:76Dpwu17wxBigWdFVutAvm
 Song Energy: 0.685
 Song Loudness: -5.8020000000000005
 ________________________________________________________________
__

 Song Title: uninvited
```

Just to take a look at one variable example to show the similarities between some of these recommendations, I'll plot ten of the song's energy.

In [108]:
```
plt.bar(recomm['song_name'].head(10), recomm['energy'].head(10))
plt.title('Top 10 Recommendations Energy')
plt.xlabel('Song Title')
plt.ylabel('Energy Level')
plt.xticks(rotation = 75)
plt.show()
```



## Recommender System Conclusion

Overall, this simple, yet effective, song recommendation system is great for any person or business looking to expand their music taste. The math and data driving this system works well and is scalable. My next steps, to get this into production, would be to take the last bits of code, create a new python file with them, and create an application from that. For the purpose of this report, I will keep everything here.

### Assumptions

Though my recommender system came out successful, there were some assumptions I took into account. I assumed that the data I obtained was accurately and ethically collected. Like I mentioned before, I would idealy like to use an API on this application and that can easily be implemented in the future. This would remove this assumption. I also assumed that the K-means clustering method was the best for this analysis. With other powerful clustering methods out there, it would be good to test each one to see which provides that best statistics. For the purpose of this simple system, K-means works great. But, it would be good to check other methods as well.

### Limitations

The one limitaion I came across which was mentioned in my preliminary analysis, was the limited genres in the data I used. There were only 8 genres in the entire dataset used for clustering. They also were similar in taste of music. This really does limit the current system to users or businesses that like that style of music. Though there were more than 20,000 songs in the analysis, it is still limited. I will discuss the solutions to this later on in my implementations section.

### Ethical Assessment After Creating the System

In my opening introduction, I mentioned an ethical consideration I had for this project. This was ensuring the system would results in appropriate songs related to the input song. Now this isn't something that the data can provide and will be on the user or business to double check when they are adding this song to their playlist. i.e. checking for explicit content and if they want that or not. Another concern was copyrights to music and my conclustion would be for the business to ensure they are following the laws for copyright and music.

## Implementation Plan

With the recommendation system created, implementing it will be fairly simple.

1. Create new .py file that uses the code from the 'Recommender System' section
2. Either run the .py file in Command Prompt

or

3. Create an application (simple: cxfreeze package) or (complex: creating application and code HTML to make a user interface)

The initial path I would take would be to run it in the command prompt. However, for further and better uses of this project, I would initially connect to an API, pull as many songs as it would allow, and then run then automatically run the prelminary analysis, k-means, and recommendation system all at once. This would not require much code change. This would however require a strong API and connection.

Overall, the simple implementation of this system works great for those users or businesses that are looking for a quick and accurate guide to expanding their music playlists. I've achieved the goals for this project and am happy with the outcome. Any person or business trying to expand their personal or business's music discography can do it with ease by utilizing this recommender system.

## Appendix

Dataset URL: https://www.kaggle.com/datasets/mrmorj/dataset-of-songs-in-spotify?select=genres_v2.csv (https://www.kaggle.com/datasets/mrmorj/dataset-of-songs-in-spotify?select=genres_v2.csv)