Project Proposal: Evaluating Trends through Clustering

Khawar Murad Ahmed
Shreeman Gautam

ACM Reference Format:
Khawar Murad Ahmed and Shreeman Gautam. 2022. Project Proposal: Evaluating Trends through Clustering. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The question that we would like to study has slightly changed. The main questions we'd like to study are the following: How does music differ as time passes? What were the different musical trends across generations? For that, we are using a dataset that we found on kaggle.

The dataset that we plan on using is a list of around 160,000 songs from Spotify that was found in Kaggle. It includes columns such as the name of the song, artist, the date in which it was released etc., amounting to 19 dimensions. Out of these 19 dimensions, 13 are numerical in nature. Of the 13 dimensions, explicit and mode were two categorical features (0 or 1), and the rest were numerical, which are acousticness (range 0 to 1), danceability (range 0 to 1), duration in milliseconds (typically ranging 200k to 300k), energy(range 0 to 1), instrumentalness(range 0 to 1), key(from 0 to 11 starting C as 0), liveness (range 0 to 1), loudness (float ranging from -60 to 0), speechiness (range 0 to 1), tempo (float typically ranging from 50-150), and valence (range 0 to 1). During pre-processing, while the dataset was from 1920-2020, we decided that our dataset would only include songs from 1960-2020 just to make it more convenient to work with (roughly speaking, while we are more familiar with music from the last 20 years, 1960s was manageable while the 1920s was not).

For the techniques that we used, we utilized clustering and regression. More specifically, we decided to go with k-means clustering and linear regression. We did clustering in order to learn more about the kind of data that we're dealing with, and since we can't go through each song on its own, clustering helped us get a sense of what the algorithm thought was similar and what was different. So, we used clustering to make the date splits that we did. So the way we organized our sets were based on emperical results that we got from clustering. We also tried dimentionality reduction, but that did not give good results as we will see later. Overall, we saw decent results but not as well we had hoped and will discuss the reasons for that towards the end of the paper.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 TECHNIQUES AND RESULTS: CLUSTERING

As part of K-means clustering, we ran experiments to see how data would cluster for different number of clusters. We have varied the number of clusters from n = 3 to n = 6. K-means clustering was done using scikit-learn's K-means module. Even though, K-means is random, we have observed that from n = 3 to n = 6, approximately the same clusters are produced albeit in a different order with every run of the code. So for instance, for n = 6, we get 6 clusters where:

- In run = 1, we see that cluster-0 has 53459 songs, cluster-1 has 4505 songs, cluster-2 has 365 songs, cluster-3 has 40565 songs, cluster-4 has 22743 songs and cluster-5 has 19 songs.
- In run = 2, we see that cluster-0 has 40395 songs, cluster-1 has 22806 songs, cluster-2 has 4503 songs, cluster-3 has 362 songs, cluster-4 has 53572 songs and cluster-5 has 18 songs.

A closer look at the 2 runs tells us that:

- cluster-0 of run 1 is approximately the same as cluster-4 of run 2.
- cluster-1 of run 1 is approximately the same as cluster-2 of run 2.
- cluster-2 of run 1 is approximately the same as cluster-3 of run 2.
- cluster-3 of run 1 is approximately the same as cluster-0 of run 2.
- cluster-4 of run 1 is approximately the same as cluster-1 of run 2.
- cluster-5 of run 1 is approximately the same as cluster-5 of run 2.

We did 10 runs each of the code and each run produced 6 clusters(in different orders) with approximately the same 6 values(same thing holds for n = 3 to n = 5 clusters). This proves to us that despite, the randomness of K-means, approximately the same 6 clusters are produced albeit in different orders.

We then put our clusters into histograms. As an example, for n = 3, we get 3 clusters. For each cluster, we made a histogram. On the x-axis of the histogram, we specify 6 intervals where each interval represents roughly a decade[1960-1969, 1970-1979, 1980-1989, 1990-1999, 2000-2009, 2010-2020]. On, the y-axis, we count the number of songs for that decade. We did the same thing for the other n values as well.

From n = 3 to n = 6, we get 3+4+5+6=18 histograms but we will show 3 histograms from n = 6 clusters because:

- We observe that these 3 histograms contain the majority of songs(from a total of 121656 songs in the pre-processed data, these 3 histograms contain 22741+53474+40550 = 116765 songs while the other 3 histograms contains 121656-116765 = 4891 songs)
- n = 3 to n = 5 also contain 3 histograms each, where the 3 histograms end up containing the majority of the songs

- Each of the 3 histograms indicate a clustering based on a generation. Generally, a generation is defined as 25 years. Generally, we see that songs from 1960-1979(20 years) ended up being a majority in the 1st cluster, songs from 1980-1999(20 years) ended up being a majority in the 2nd cluster and songs from 2000-2020(21 years) ended up being a majority in the 3rd cluster.

The last bullet point goes against our hypothesis because we thought that one decade would dominate one cluster(what we are saying is that we expected cluster-0 to have a majority of songs from the 1960-1969, cluster-1 to have a majority of songs from 1970-1979, ... cluster-5 to have a majority of songs from 2010-2020). Even though our former hypothesis was proven false, we can now demonstrably say that generally, songs from a generation end up being a majority in one cluster.

Here are the histograms[Figures 1-4]:

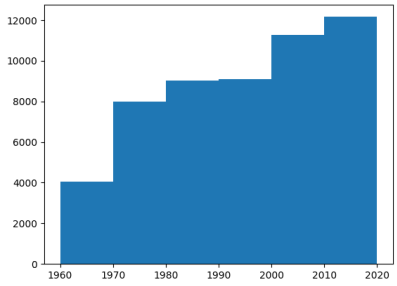

Figure 1: Cluster-0

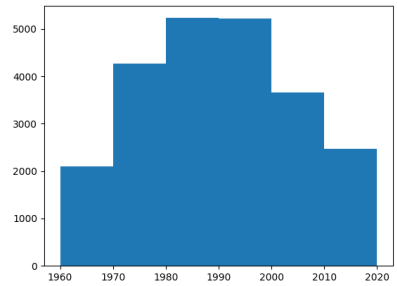

Figure 2: Cluster-1

Now, we have to emphasize that these are general trends. During our experiments from n = 3 to n = 6, we noticed that songs from 1960-1969 and 2010-2020 would end up in a cluster together. You can see that in figure 3 where the number of songs from 2010-2020 is the second highest population in the histogram.

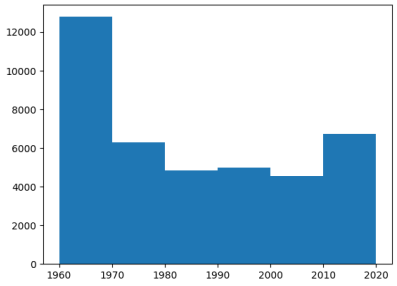

Figure 3: Cluster-2

Time Period	Mean Average Error
Training: 1960-69, 2010-19 Test: 2020	19.40
Training: 1960-79 Test: 1980	9.22
Training: 1980-99 Test: 2000	8.81
Training: 2000 - 19 Test: 2020	17.86

Table 1: Mean Average Error for each Period

3 TECHNIQUES AND RESULTS: REGRESSION

After seeing the clusters, we hypothesized that maybe adding the songs from 1960s to the 2010s and doing regression would give a good popularity score for the 2020 songs. While the performance was not completely bad when compared to just the general dataset, at the same time, it also made the performance of the linear regression worse. We define error as the average of absolute value of observed popularity score minus predicted popularity score given by regression for the relevant dataset. As shown by the table, the error for linear regression was 19.40 when 1960s and 2010s were used for training and 17.56 when 2000-19 was used for training. On the other hand, when PCA was applied, we saw that the performance was really bad, with the error being 64.21 for the dataset when 1960s and 2010s were used, and 63.12 when 2000-19 was used. When we consider that popularity can only be between 0 and 100 inclusive, we see that this kind of error is really bad. So for that, we think that two dimensions is very less to do regression. For the sake of experimentation, we tried out different dimensions(3, 5, 8, 11, 12) to see if for any k, we can see improved performances over the current best model that we have. Our results showed us that doing dimensionality reduction was good because the errors were higher(around 60-70 generally). Consequently, we decided to stick to 13 dimensions and show the cdf graphs that we got from them. Just to learn more about models, we decided to run regression on other time periods too to learn more. As seen by the Cumulative Distribution Functions (CDF) (Figures 6-12), most songs that were predicted lie right around the mean of their respective distributions. Overall, while our models do not perform the greatest, they still do

relatively okay given the unpredictable nature of the music industry. In order to check the influence the artists have, we also tried to filter by big artist names and unknown artist names, but found no actual differences. To check the difference between how much the artists affected the popularity rating, we arbitrarily selected popular and not very well known artists, and the songs selected tended to be rated highly in terms of popularity. As Table 2 shows, we found no actual problems. We also found that the mean average tends to be lower on other time periods.

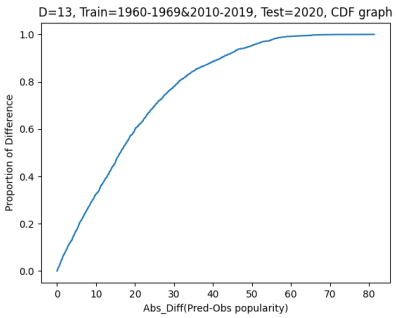

Figure 4: CDF chart, for test error(tested on 2020) on model trained on 1960-1969 2010-2019

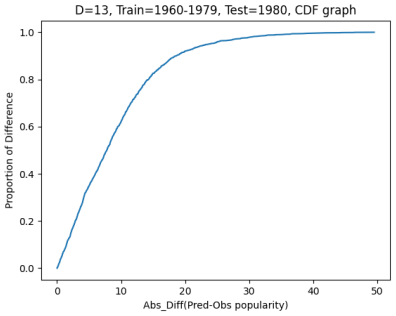

Figure 5: CDF chart, for test error on model(tested on 1980) trained on 1960-1979

4 DISCUSSION

There is a lot that goes into making a song a success, and oftentimes, it is not entirely how the songs sound. Essentially, there is no magic formula that makes a song go popular, even though certain trends might be found. That is why, even though our model does not perform exceptionally well, it doesn't perform particularly terrible either. We thought that an artist name might have been a big factor that we did not take account for, but found that there were no big differences. I think the main conclusion that we reach from this project is that our model, while mathematically sound, is devoid of any cultural context and factors that go into making the song popular. We think that there are many factors to that, but think that smaller time periods (so instead of 20 years, take 10 or 5)

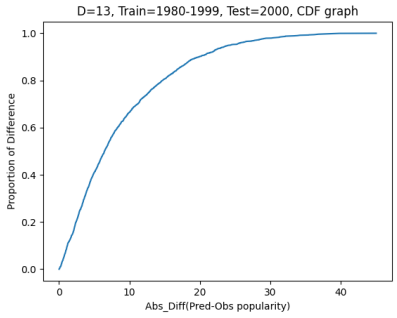

Figure 6: CDF chart, for test error on model(tested on 2000) trained on 1980-1999

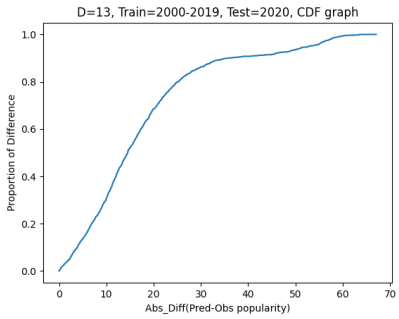

Figure 7: CDF chart, for test error on model(tested on 2020) trained on 2000-2019

Popular/Unknown	Mean Average Error
Popular	23.15
Unknown	24.04

Table 2: Trained on the years 2000-2019, Tested on 2020 with separation on whether the Artists were popular or relatively unknown.

would make our model perform a lot better. The reason our model seems to perform well on older time period is because the range of popularity values is lower, and so the resulting Mean Average Error is also lower. This is backed up by our findings which show us that the 2020 test dataset popularity ranges from 0 to 100, the 1980 test datset popularity ranges from 19 to 82 and the 2000 test datset popularity ranges from 32 to 83. In addition, we also would have liked to have an exact criteria through which popularity was measured in the original dataset, which would have given us a lot more context.