SPOTIFY SONG MOOD ANALYSIS USING CLASSIFICATION AND CLUSTERING

Presented By : GPyTer

Machine Learning | 2024

ABSTRACT

We have devised an innovative approach to predict a track's mood based on audio features. We have mainly classified a song into one of four moods – calm, happy, energetic, and sad. For EDA, we have constructed Heatmaps correlating the features, Radar plots, and KDE plots for better visualization of our data. We attempted to do feature selection using Principal Component Analysis. We have implemented Classification as well as Clustering methods to model our data. The Classification models include the Gaussian Naive Bayes classifier, Decision Tree, Random Forest Classifier, and K-Nearest Neighbours. As for clustering, we have employed the K-Means algorithm. After this, we compared the results obtained from both paradigms and attempted to draw conclusions on our dataset

OVERVIEW

PROBLEM

Our primary objective is to try and use mood as a basis of classification of songs, which can enhance the quality of song recommendations for the user.

We use the results of the project to compare and contrast classification and clustering approaches to the problem.

Supervised

We approach it as a Classification problem where we have 4 mood labels and we try to classify each song as one of those labels.

Unsupervised

We approach it as a Clustering problem to just group songs together on the basis of the audio features in our dataset.

DATA SCRAPING

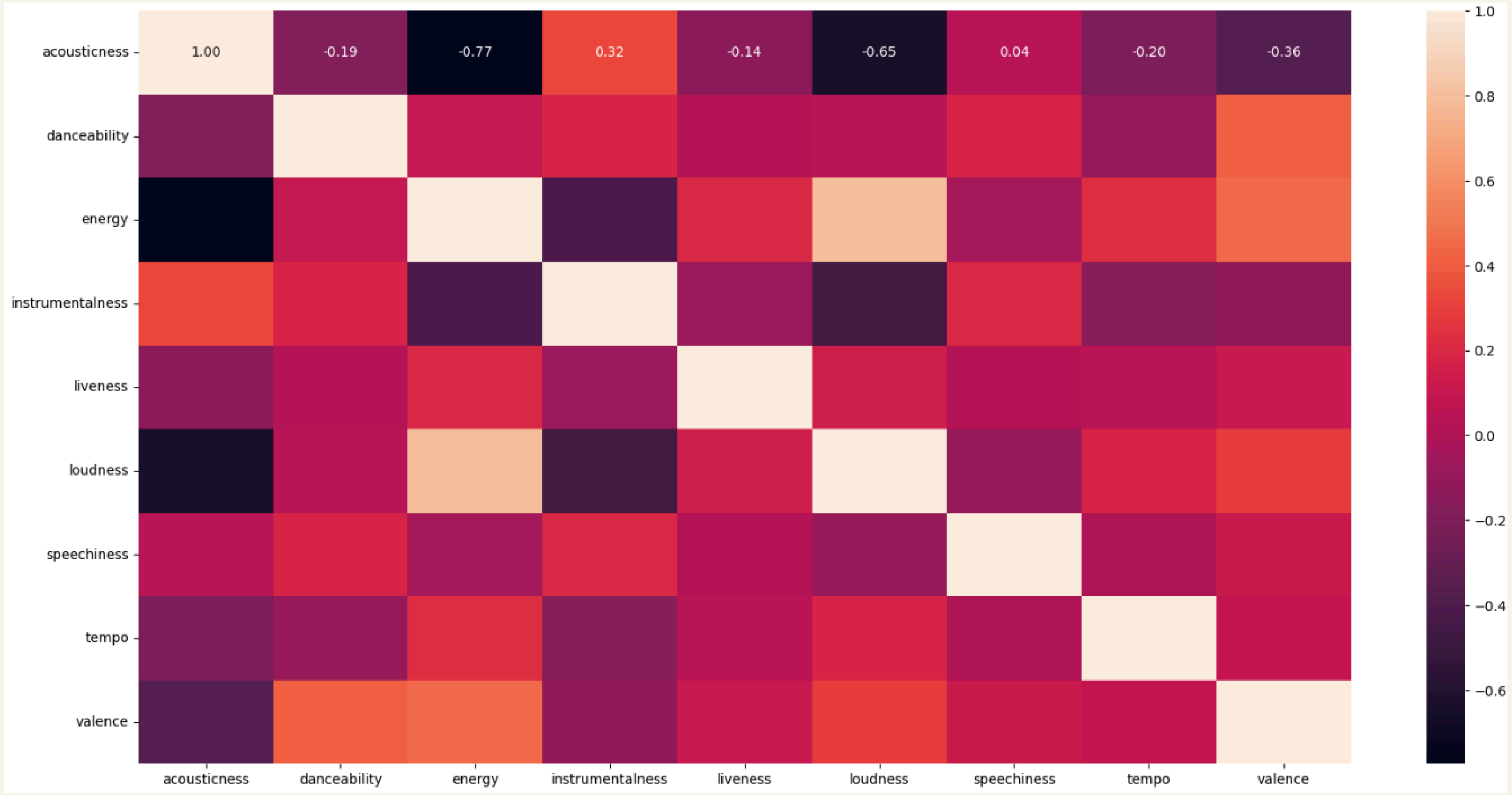
- We had initially planned to make our dataset using the spotipy python library which fetches the data from several playlists labeled as ‘Calm’, ‘Happy’, ‘Sad’, and ‘Energetic.
- The standard playlists made by Spotify which seemed to be accurate were fairly small (usually 50-100 songs) which did not provide for proper accurate training of the model.
- Large user-made playlists had a lot of variance, where songs overlapped as well as could be incorrectly labeled based on the biased preferences of the user who created the playlist.
- This creates a need for manual labeling of data, which is not feasible for large datasets for the given timeline of the project.

EXPLORATORY DATA ANALYSIS

We have used a labeled dataset that we found on GitHub that contained multiple audio feature values and the corresponding mood of each song.

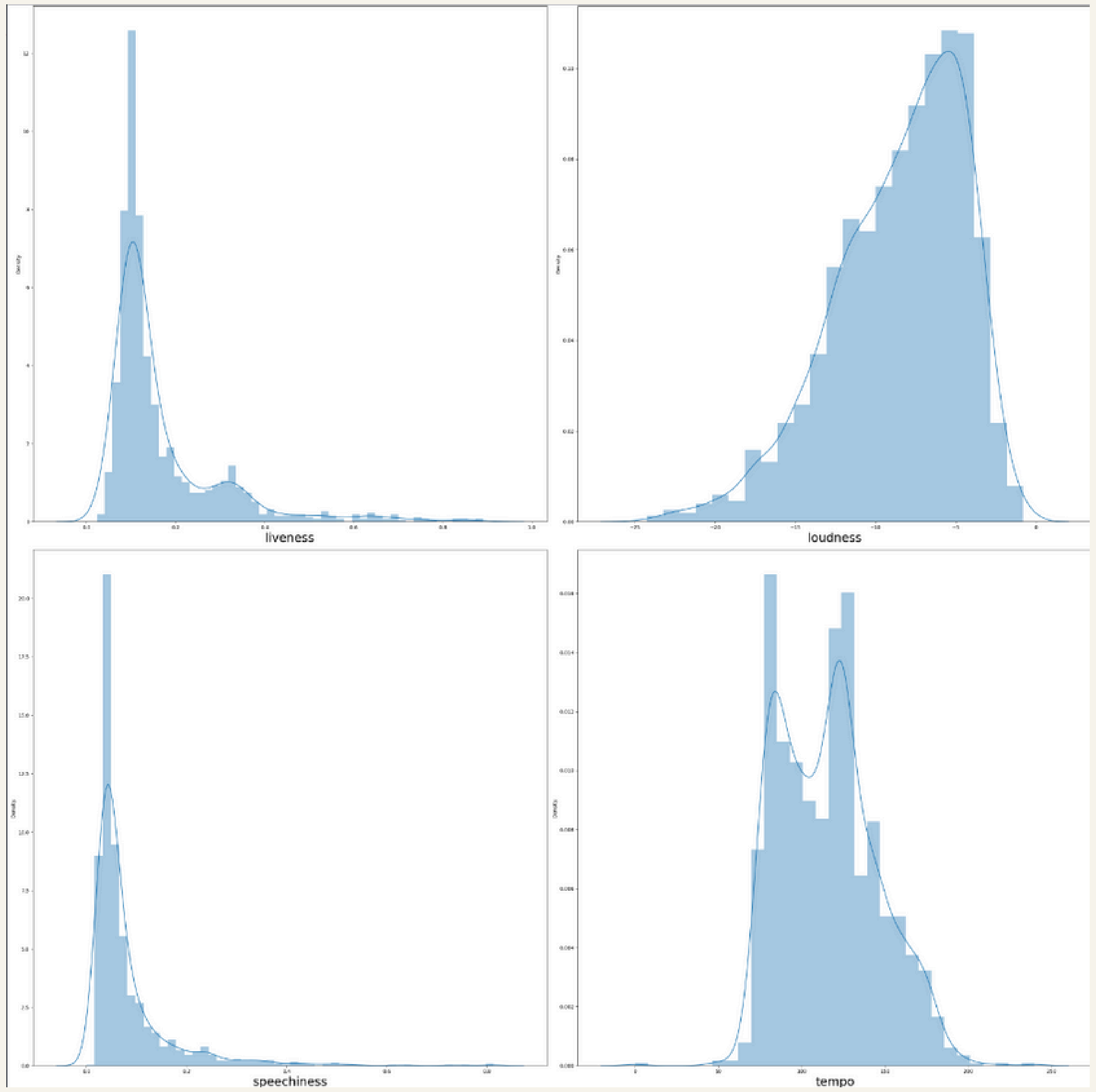
The data that we have used for training and analysis has roughly 1500 rows.

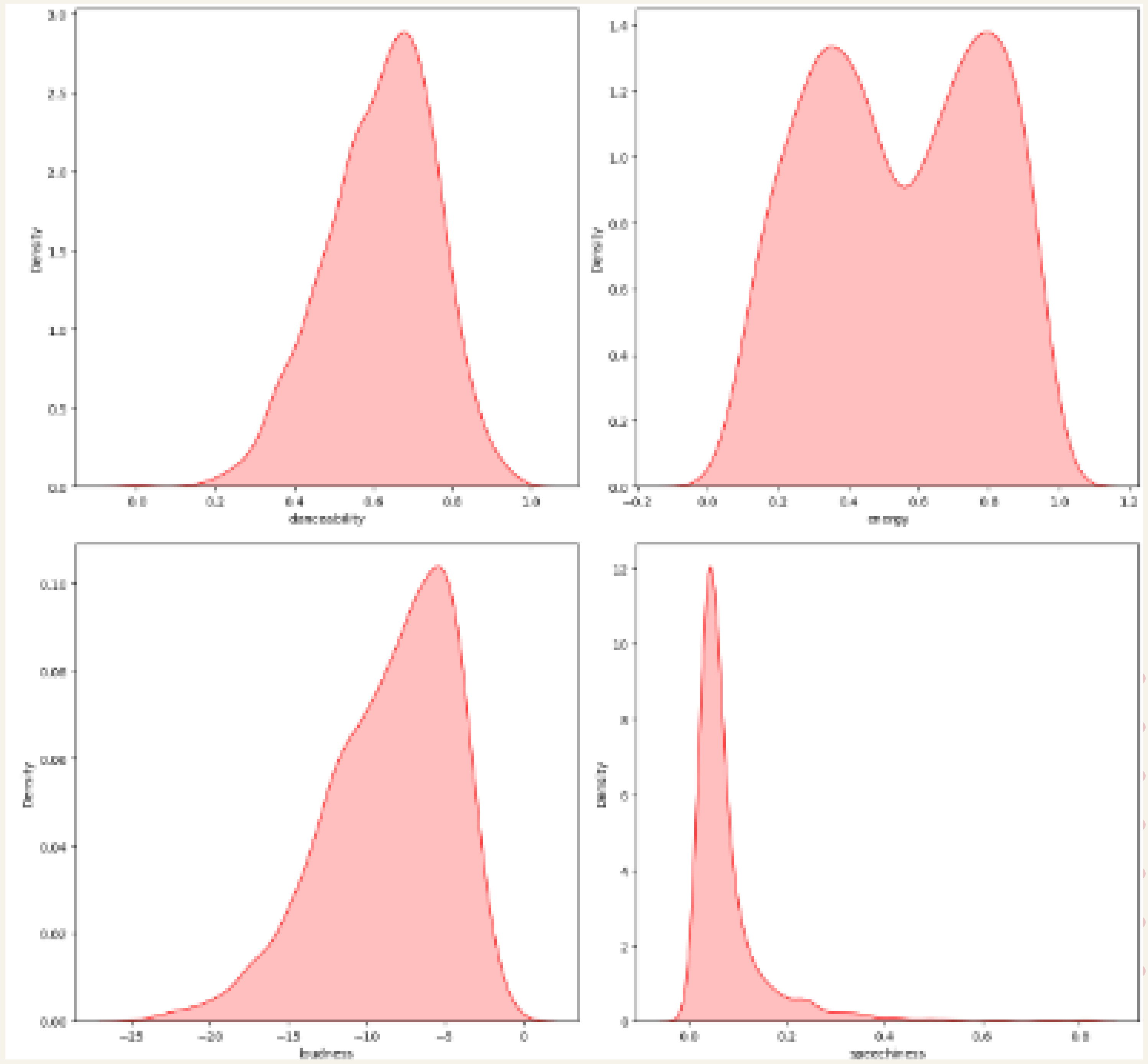
UNDERSTANDING THE DATA



Different features are not correlated to each other. Hence they all provide unique information to our classification

UNDERSTANDING THE DATA



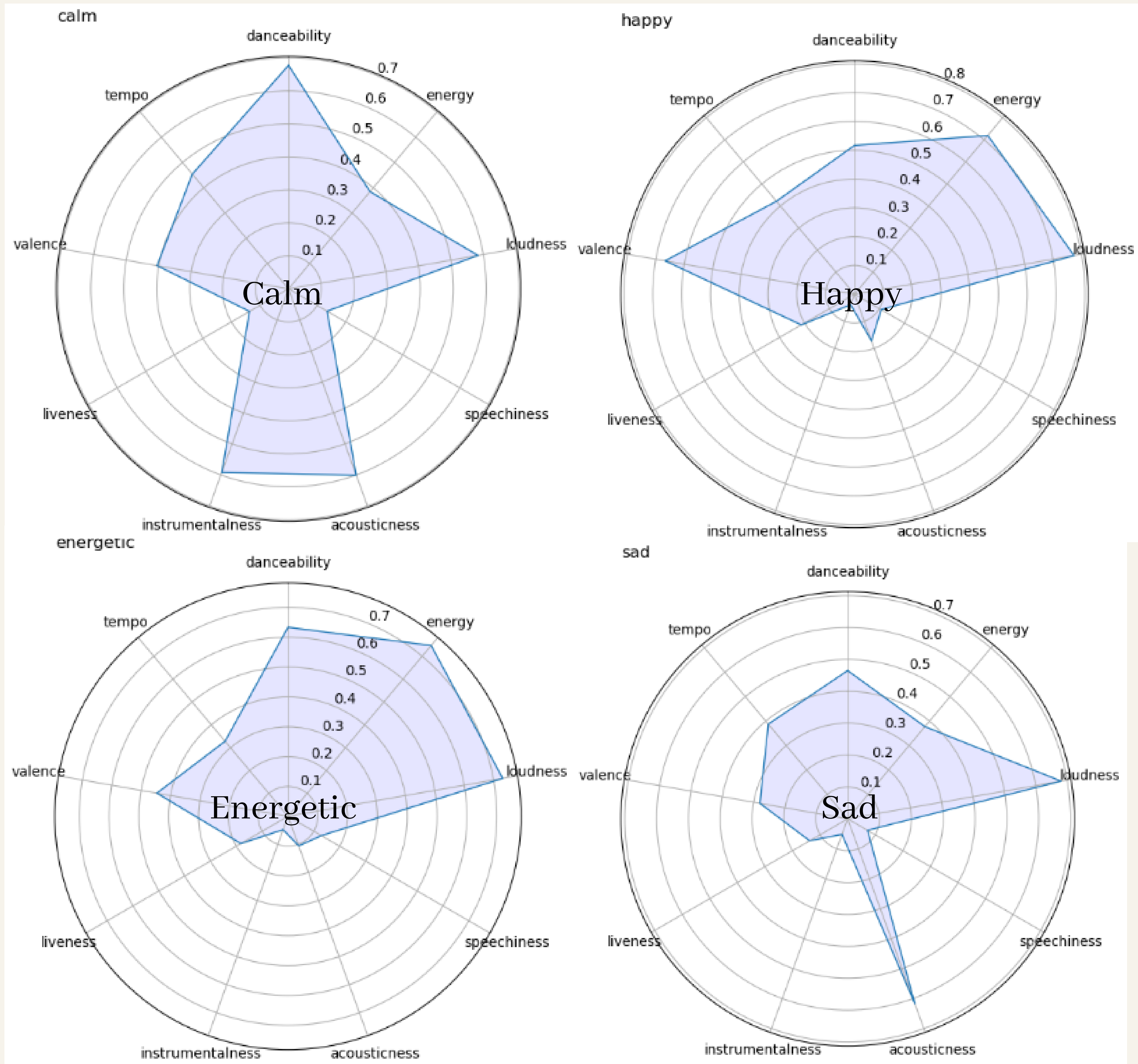


The data follows Gaussian class conditional distribution across most features.

UNDERSTANDING THE DATA

- We segregated the data based on classes. Then for each class, we first normalized each feature using min-max normalization.

- Then, we took the mean and median values of all features and plotted a radar chart to visualize what average values all features take for each class and how they are different for all the classes. The radar charts based on mean and median were fairly the same for every class.

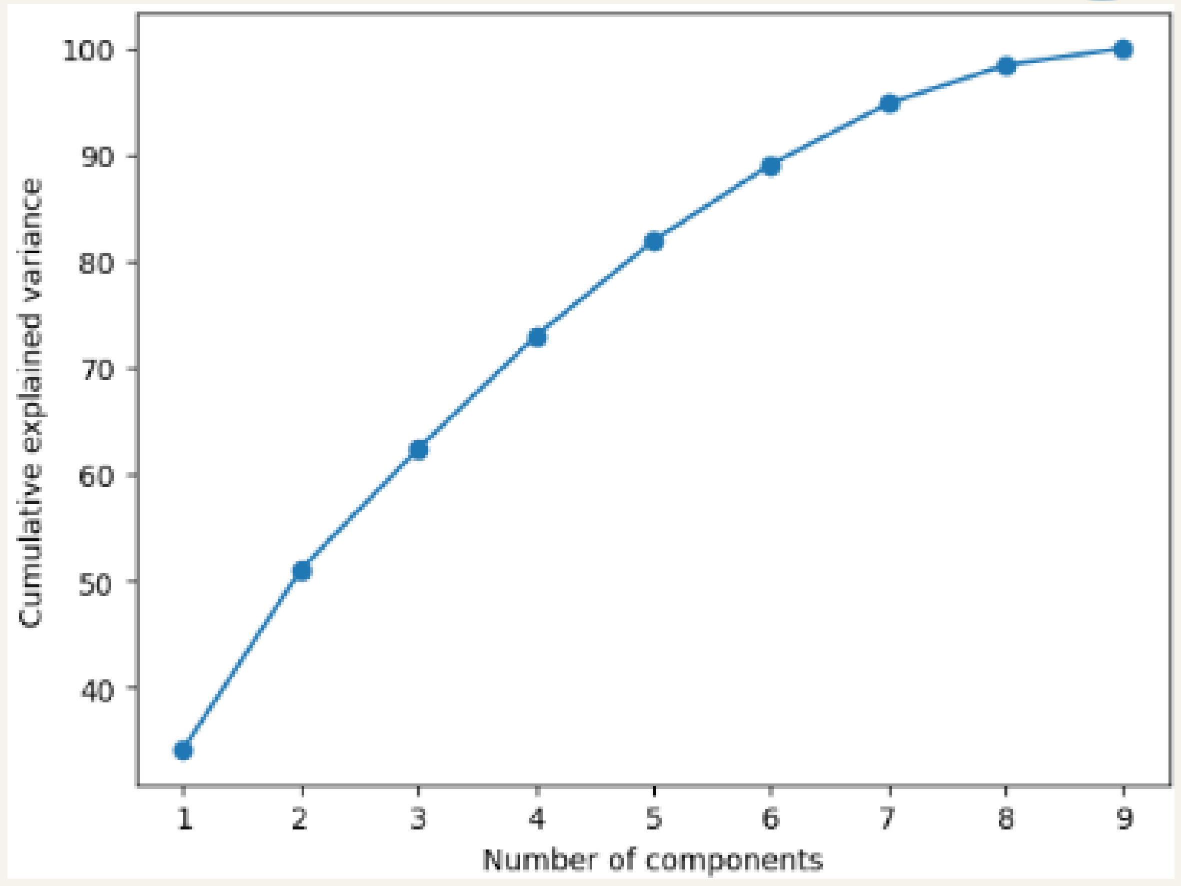


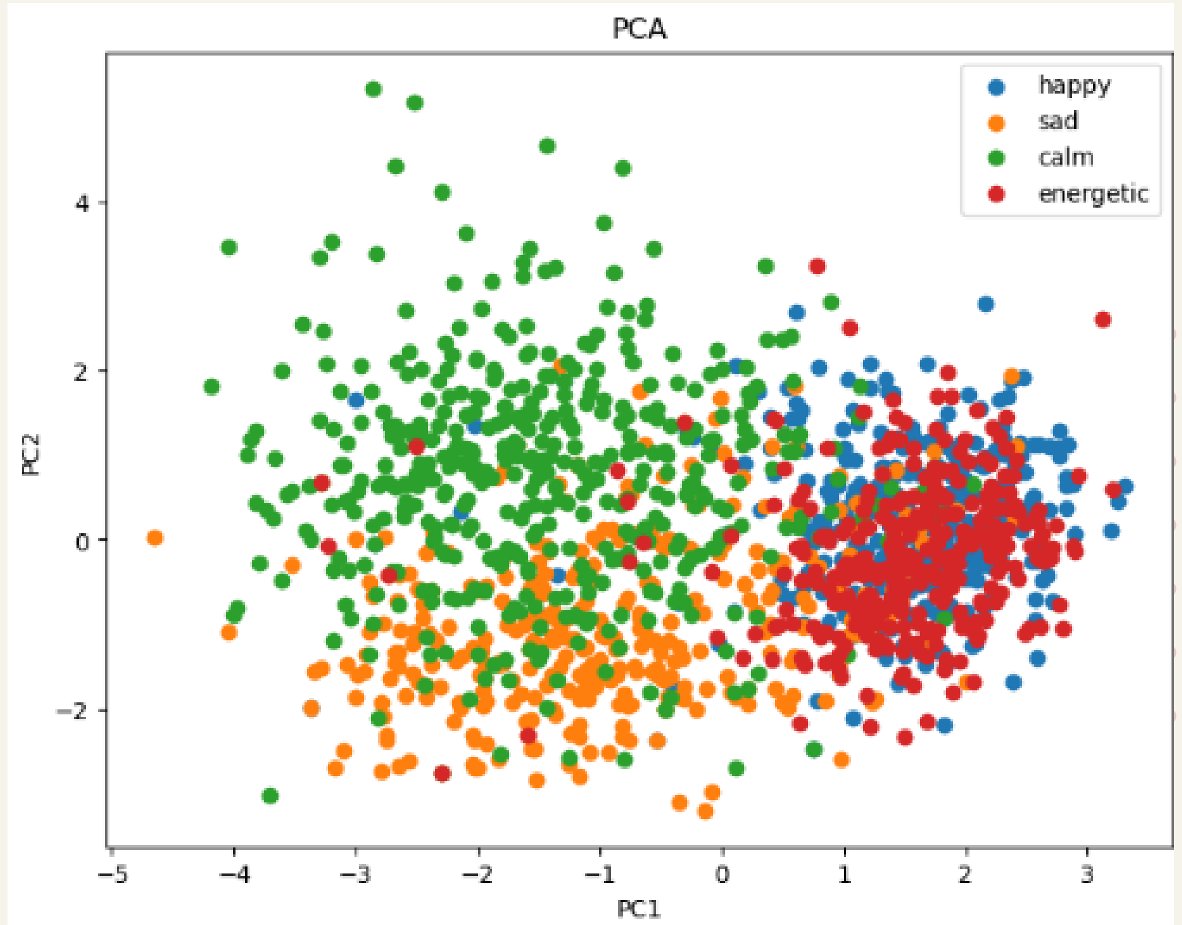
UNDERSTANDING THE DATA

PRINCIPAL COMPONENTS ANALYSIS

- If we consider at least 85-90% variance explained as a threshold, we found that it would require a projection onto a 6-dimensional space. However, this was not a considerable gain (reducing only 2-3 features) as compared to the possible information loss and reducing the accuracy of classification.

- Happy and Energetic datapoints are highly overlapping while Sad and Calm points are slightly overlapping.





GAUSSIAN NAIVE BAYES CLASSIFIER

GAUSSIAN NAIVE BAYES

WHY?

We can expect Gaussian Naive Bayes to work particularly well on our dataset because of the following reasons :

1. **<u>Data points exhibit Normal Distribution</u>: Our model capitalizes on the underlying Gaussian assumption, utilizing the inherent distribution of our song features to enhance classification accuracy.**
2. **<u>Features are not correlated</u>: Since the features of our dataset all provide unique information to our classification, and do not depend on each other, our model's assumption of conditional independence among predictors fits well, and allows the model to effectively classify despite the diverse range of features in our dataset, assigning equal importance to each characteristic in the final mood classification.**
3. **<u>Robustness and Scalability</u>: With so many song characteristics (features) available, GNB performs well in scalability, effortlessly handling the highdimensional nature of our dataset.**

GAUSSIAN NAIVE BAYES

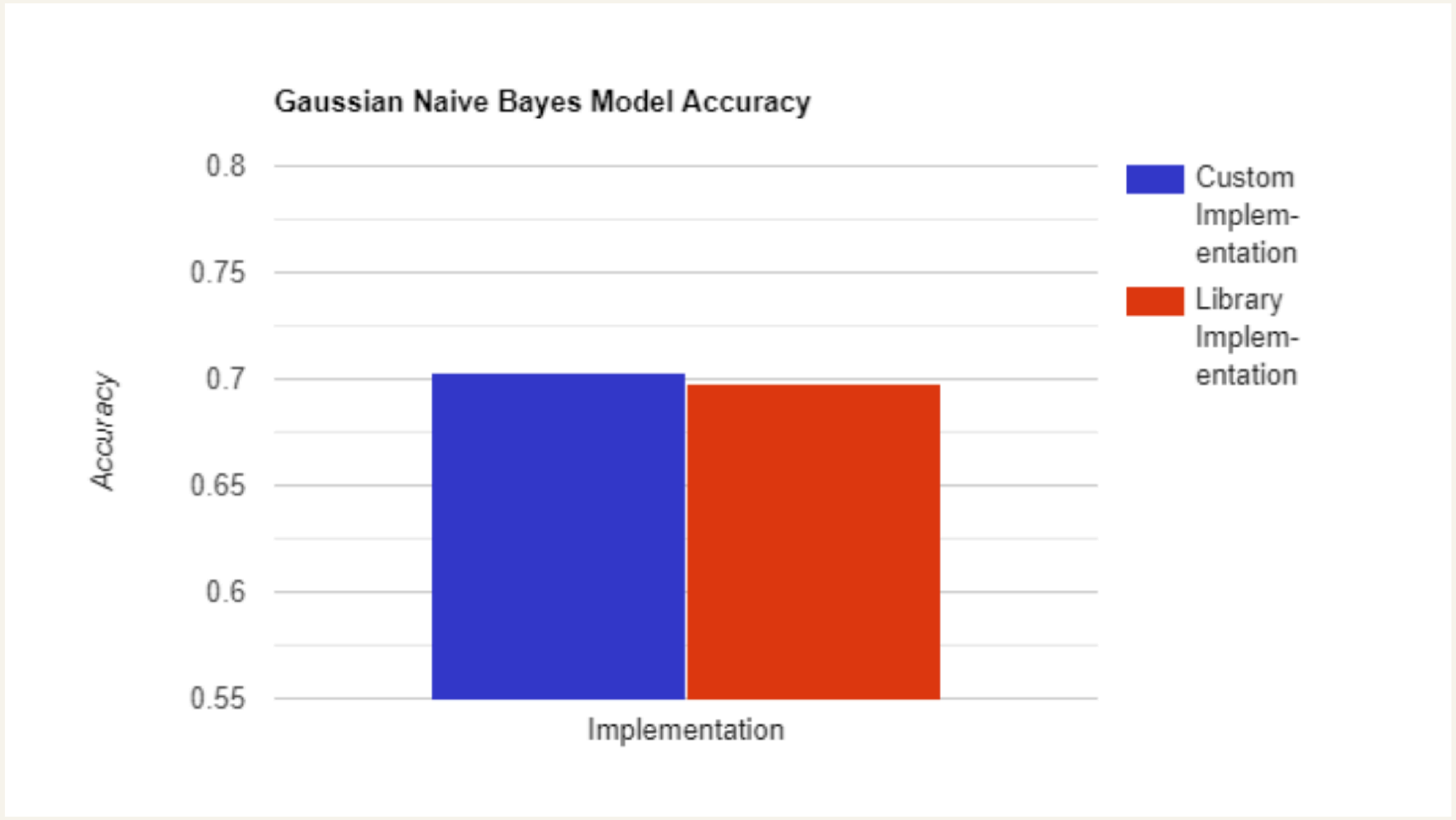
IMPLEMENTATION

1. Separate By Class : Parses through the entire dataset and separates rows according to their respective class.
2. Summarize Dataset : Computes the mean and standard deviation for every feature separately.
3. Summarize Data By Class : We use the above two functions to separate the data by class, and compute statistics for each feature, for each class.
4. Gaussian Probability Density Function : Computes the probability that a given data point lies in a particular class by multiplying the probability for each of the 9 features.
5. Class Probabilities : Multiply the above value with the posterior probability of that class to find the probability of that data point lying in the class.

GAUSSIAN NAIVE BAYES

PERFORMANCE

- We evaluated the performance of our model using K-fold Cross Validation.



Comparing accuracy of our implementation with the sklearn library implementation

DECISION TREE RANDOM FOREST CLASSIFIER

WHY RANDOM FOREST?

- <u>Ensemble learning benefits</u>: With a dataset of around 1500 rows, Random Forest's bagging of decision trees with Bootstrap helps prevent overfitting and enhances model robustness by considering multiple subsets of data.
- <u>Robustness to noise and outliers</u>: By aggregating results from multiple decision trees, Random Forest is less sensitive to outliers, which is crucial in fields like music where outliers are common and may affect mood classification.
- <u>Handling non-linear relationships</u>: With 9 features and 4 mood classes, Random Forest is effective at capturing non-linear relationships, enhancing classification accuracy in scenarios where such relationships are likely present

IMPLEMENTATION

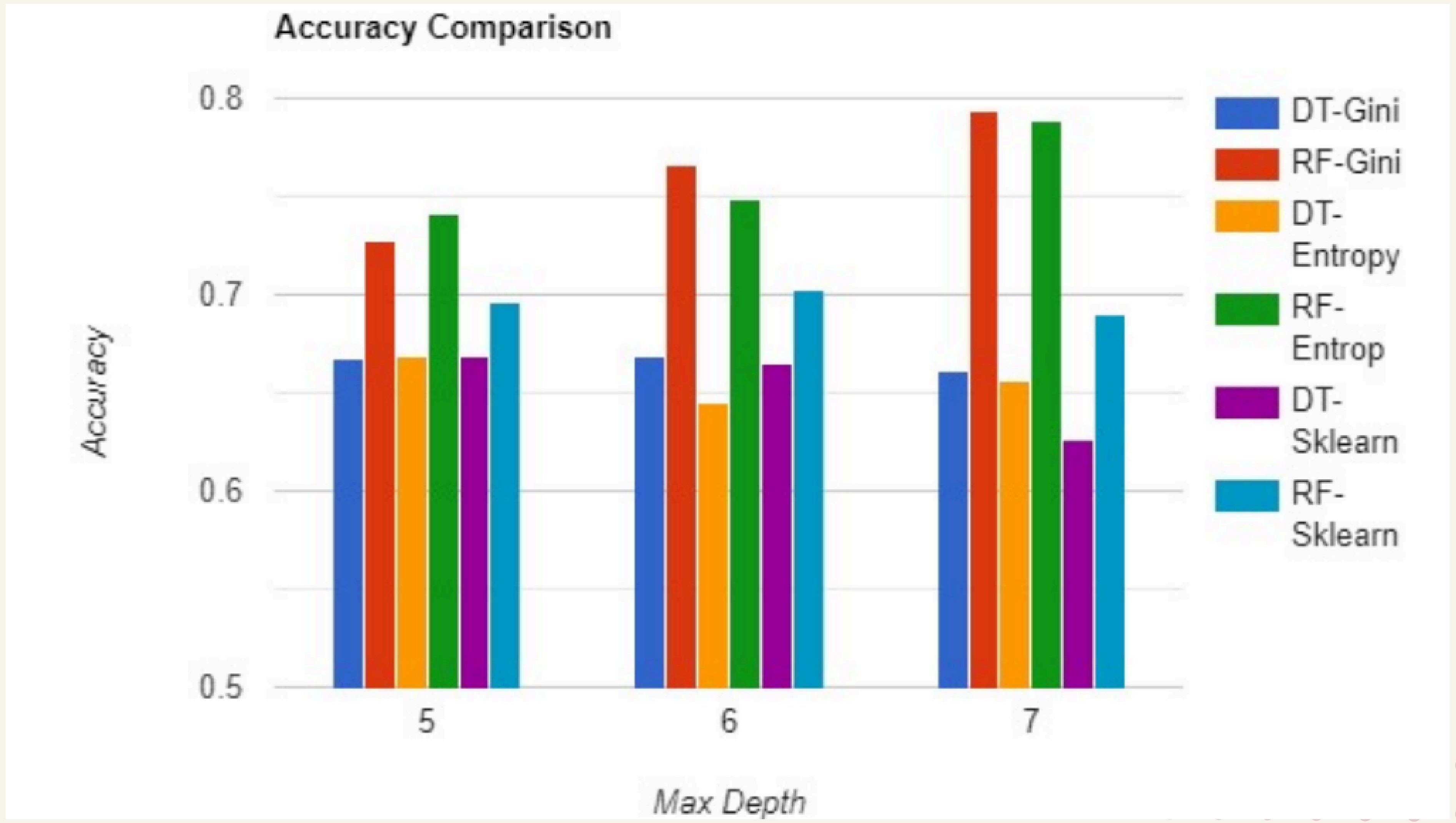
- Implemented decision tree from scratch using 2 different impurity measures for determining the optimal split at any node.
- Gini impurity is a measure of how often a randomly chosen element from the dataset would be incorrectly classified if it were labelled according to the distribution of class labels in the node.
- Entropy measures the level of disorder or uncertainty in a dataset's class distribution
- By resampling subsets of training data with replacement, diverse bootstrap samples are generated, aiding in mitigating overfitting and improving generalization to unseen data. Each decision tree in the Random Forest class is trained on a bootstrap sample.
- The outputs generated by different decision trees is aggregated using voting method.

PERFORMANCE

- For our data, Gini impurity turned out to give slightly better results than entropy, however, the difference is minimal only.

- Decision Tree yields less accuracy than Random Forest which is expected since it might be overfitting and capturing the noise in the training data.

- For random forest, our implementation is performing better than the library implementation



K-NEAREST NEIGHBOURS CLASSIFIER

K-NEAREST NEIGHBOURS

The K-nearest neighbors algorithm is a popular machine learning algorithm that is used for both classification and regression tasks. It works by finding the K-nearest data points to a given data point that we want to classify or predict the value of.

Why KNN?

1. <u>KNN for Few Features</u>: Chose KNN due to the limited number of features, as it performs well with fewer dimensions.
2. <u>Non-Parametric Nature</u>: KNN is non-parametric, meaning it doesn't assume a specific data distribution, which is advantageous for diverse and complex song attributes.
3. <u>Prediction based on Similarity</u>: Predictions are made based on the similarity of instances in the feature space, aligning with the idea that songs with similar features likely evoke similar moods.

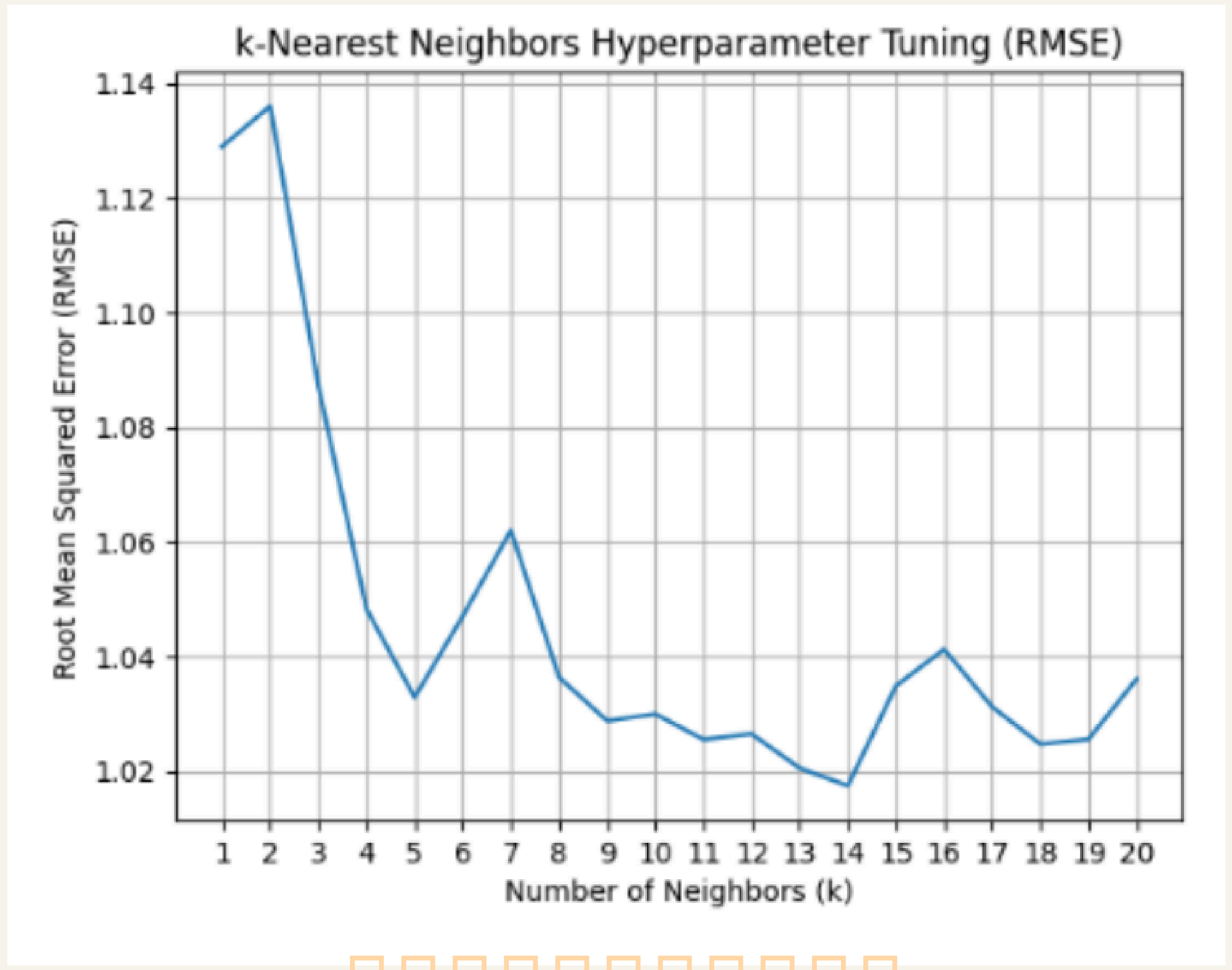
IMPLEMENTATION

1. Calculate Distances: Measure the distance between the data point to classify and all other data points in the dataset.
2. Select Nearest Neighbors: Choose the K data points closest to the one being classified.
3. Determine Classes: Identify the classes of the selected nearest neighbors.
4. Assign Class: Assign the most frequent class among the nearest neighbors to the data point being classified.

1. Why Euclidean Distance?: Euclidean distance is preferable for features with floating-point numbers, while Hamming distance works well for categorical features.
2. Parameter Tuning: . The value of K is a parameter that needs to be determined based on the minimum Root Mean Squared Error (RMSE) for the respective K. RMSE measures the differences between the predicted values and actual values.

PARAMETER TUNING

We did Parameter tuning for k = 1 to 21,and we obtained the following plot for Number of neighbors vs RMSE.

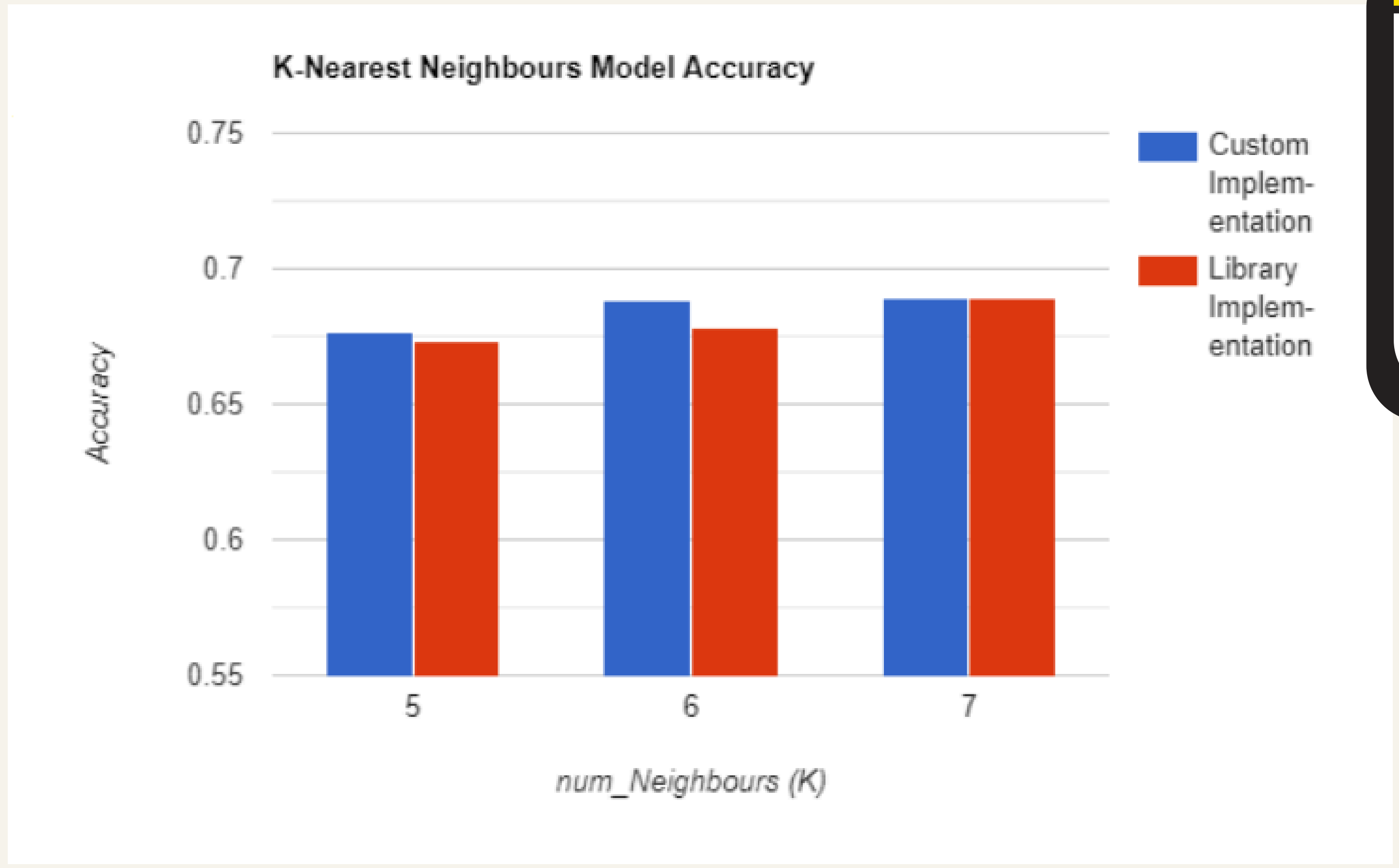


Global minima at k = 14
Subsequently, for k = 14
Mean Accuracy: 69.932%

- RMSE decreases as the number of neighbors (k) increases initially, and then it starts to increase again after reaching a minimum.
- This trend suggests that with a smaller value of k, the model may suffer from overfitting, leading to higher RMSE due to capturing noise in the data.
- On the other hand, with a larger value of k, the model may become too generalized, resulting in higher bias and also higher RMSE

PERFORMANCE

- We performed a evaluation of K-Nearest Neighbors (KNN) algorithm using 5-fold cross validation.
- We tested the KNN algorithm on different values of k that is num neighbors, namely 5, 6, and 7.



k = 5: 67.703%
k = 6: 68.851%
k = 7: 68.919%

We can clearly see, mean accuracy for k = 14 which is 69.932% is much better than k = 5, 6, and Thus parameter tuning is important for finding the best value of k thus best mean accuracy that we can obtain.

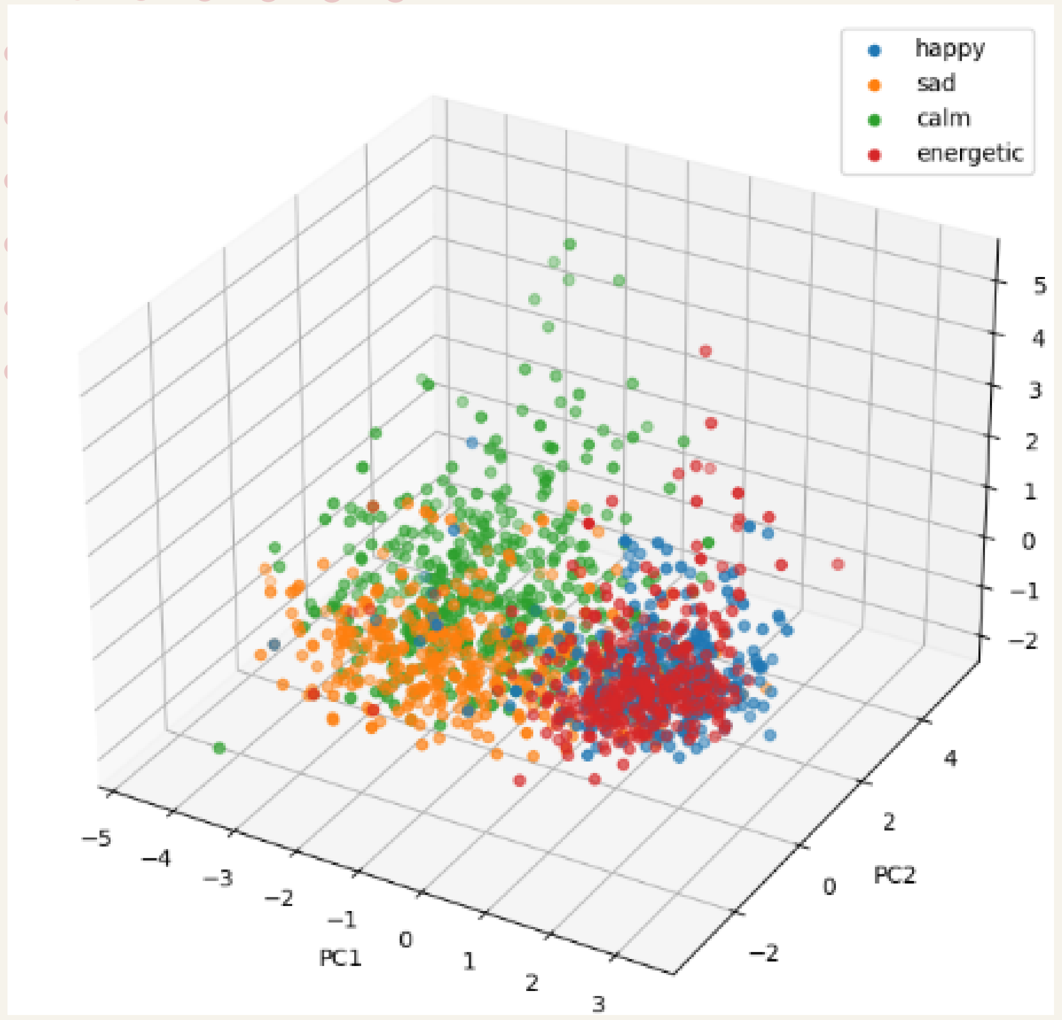
COMPARING PERFORMANCE OF CLASSIFICATION ALGORITHMS

- After comparing the accuracy achieved by various classification methods, we found that Random Forest performs best for this task on our dataset, followed by Naive Bayes, KNN and then finally Decision Tree.

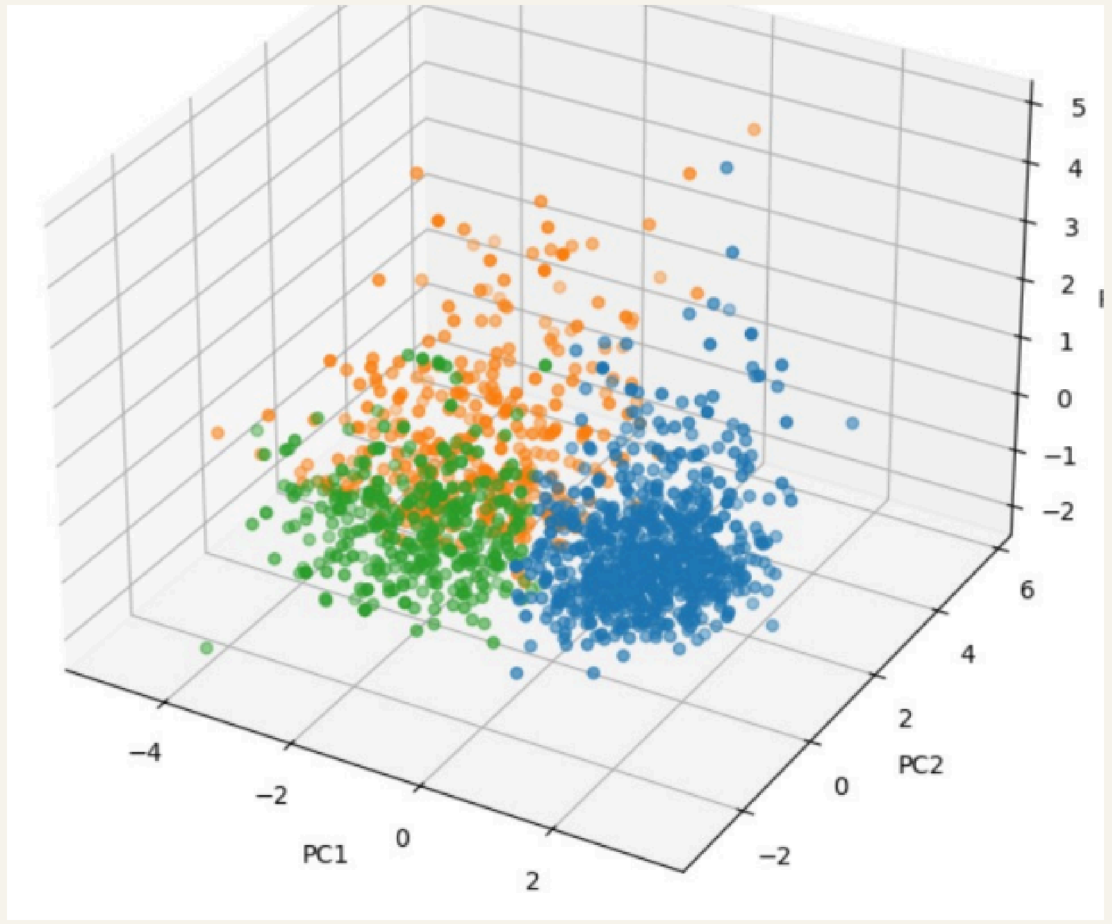
- They are compared using accuracy as the metric.

- Accuracy of the methods :
  - Random Forest : 77.76%
  - Naive Bayes : 70.34%
  - K-Nearest Neighbours : 69.93%
  - Decision Tree : 68.58%

K-MEANS CLUSTERING



Classes vs Clusters



WHY CLUSTERING? WHY K-MEANS?

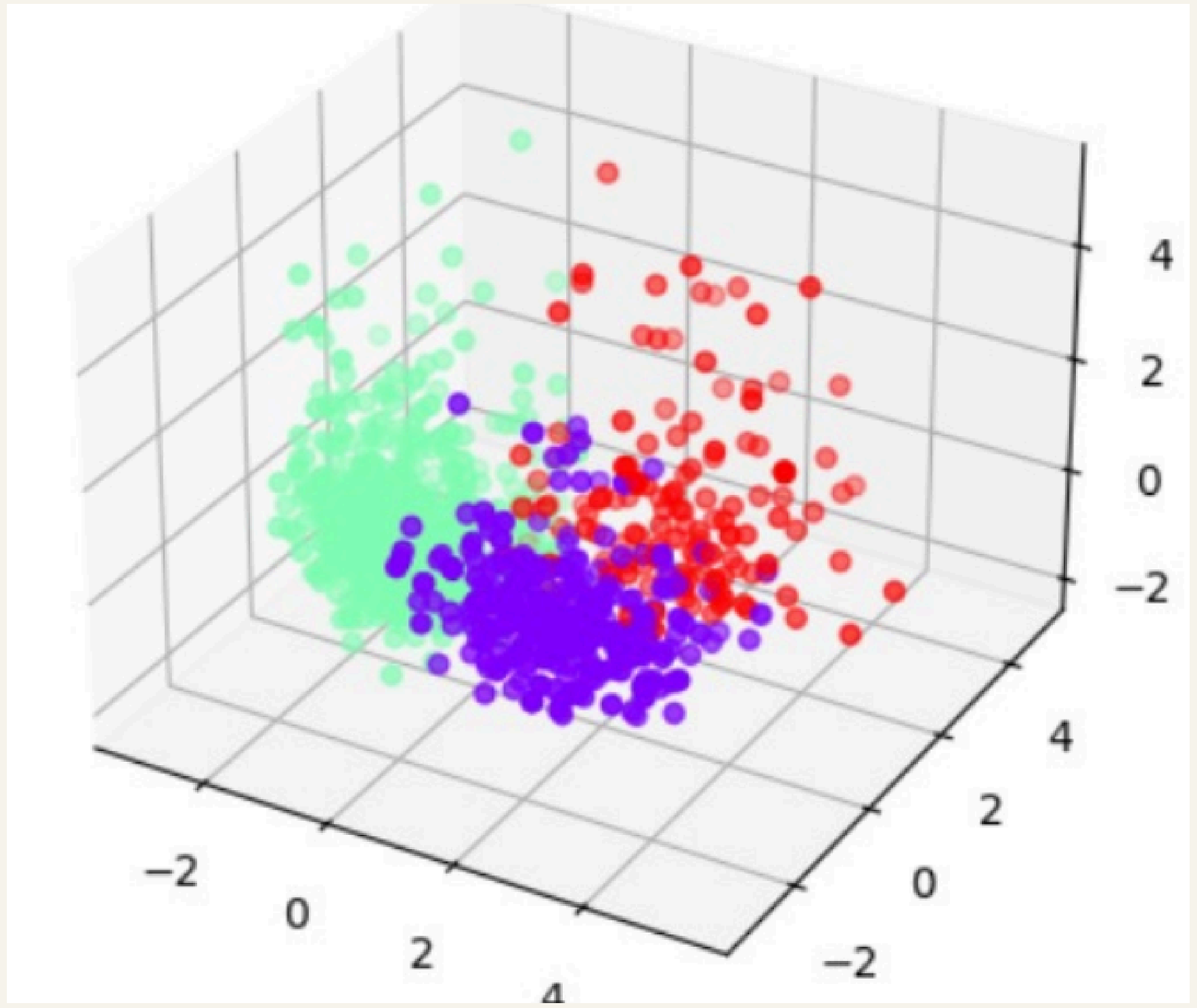
- We found out that there is a lot of overlap of songs between different mood labels which makes sense as well because music is a very subjective field

- In this case, the problem can be formulated as grouping similar songs(with respect to the audio features we have) together so that if a user likes a song, he/she can then access similar songs

- K-means is an easy algorithm to implement which is guaranteed to converge. It is also pretty fast when compared with other clustering algorithms.

IMPLEMENTATION

- Initially, K centroids are randomly selected from the dataset. Then, in an iterative process, each data point is assigned to the nearest centroid based on Euclidean distance.
- After all points are assigned, the centroids are recalculated as the mean of the points assigned to them.
- This assignment and centroid update process repeats until convergence, typically defined by minimal movement of centroids or reaching a maximum number of iterations.



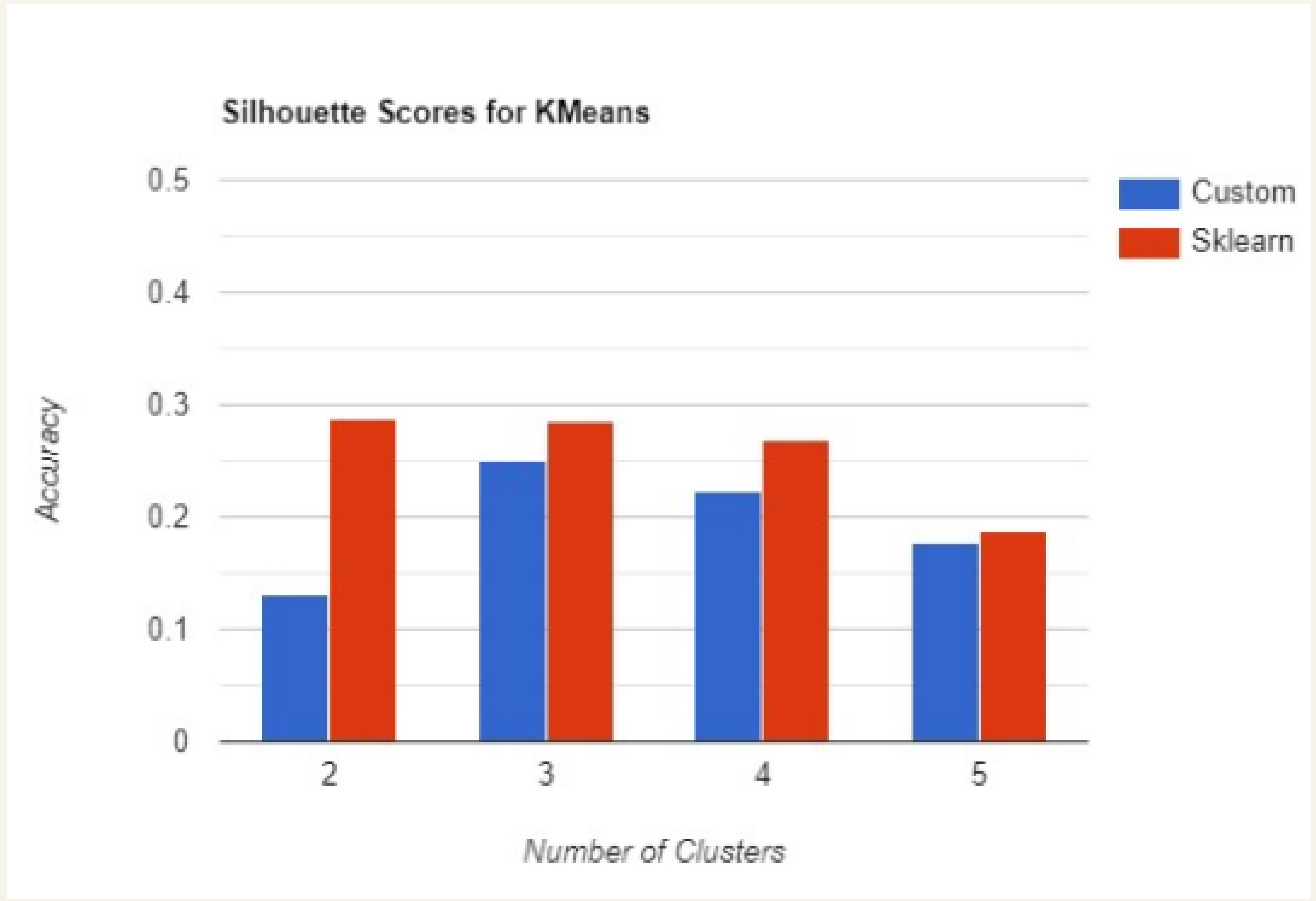
Sklearn implemented clusters

MODEL EVALUATION AND PARAMETER TUNING

- We have used the Silhouette score to evaluate the quality of clusters generated.

- We observed that the score increased from k=2 to k=3 and then decreased on increasing k further.

- This validates our analysis as to how happy and energetic songs are very similar while sad and calm songs tend to form their separate clusters while some sad and calm songs are grouped in the last cluster.



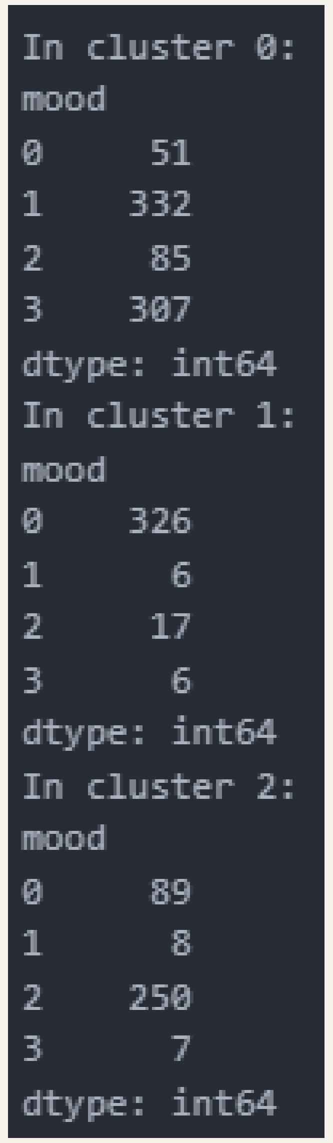
.

CLUSTERING VS CLASSIFICATION

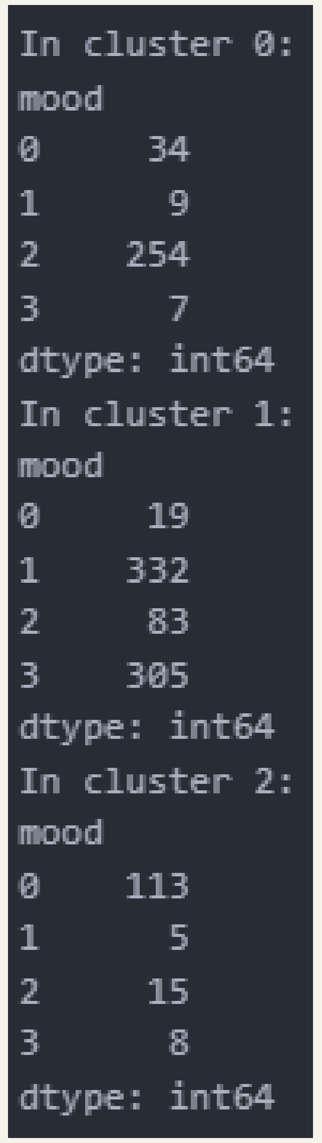
- Some of the calm songs were clustered separately, while the others overlapped with the cluster majorly consisting of sad songs. Meanwhile happy & energetic songs clustered together.

- This is in coherence with our understanding of music where the characteristics of happy and energetic songs are- Loud, lively, and danceable, while the calm and sad songs have similar characteristics- slow, acoustic, and sedate.

- It highlights the importance of unsupervised learning for this use case which mostly consists of song recommendation and classification based on the mood.



```
In cluster 0:
mood
0     51
1    332
2     85
3    307
dtype: int64
In cluster 1:
mood
0    326
1      6
2     17
3      6
dtype: int64
In cluster 2:
mood
0     89
1      8
2    250
3      7
dtype: int64
```



```
In cluster 0:
mood
0     34
1      9
2    254
3      7
dtype: int64
In cluster 1:
mood
0     19
1    332
2     83
3    305
dtype: int64
In cluster 2:
mood
0    113
1      5
2     15
3      8
dtype: int64
```

Custom K-means vs Sklearn

CHALLENGES FACED

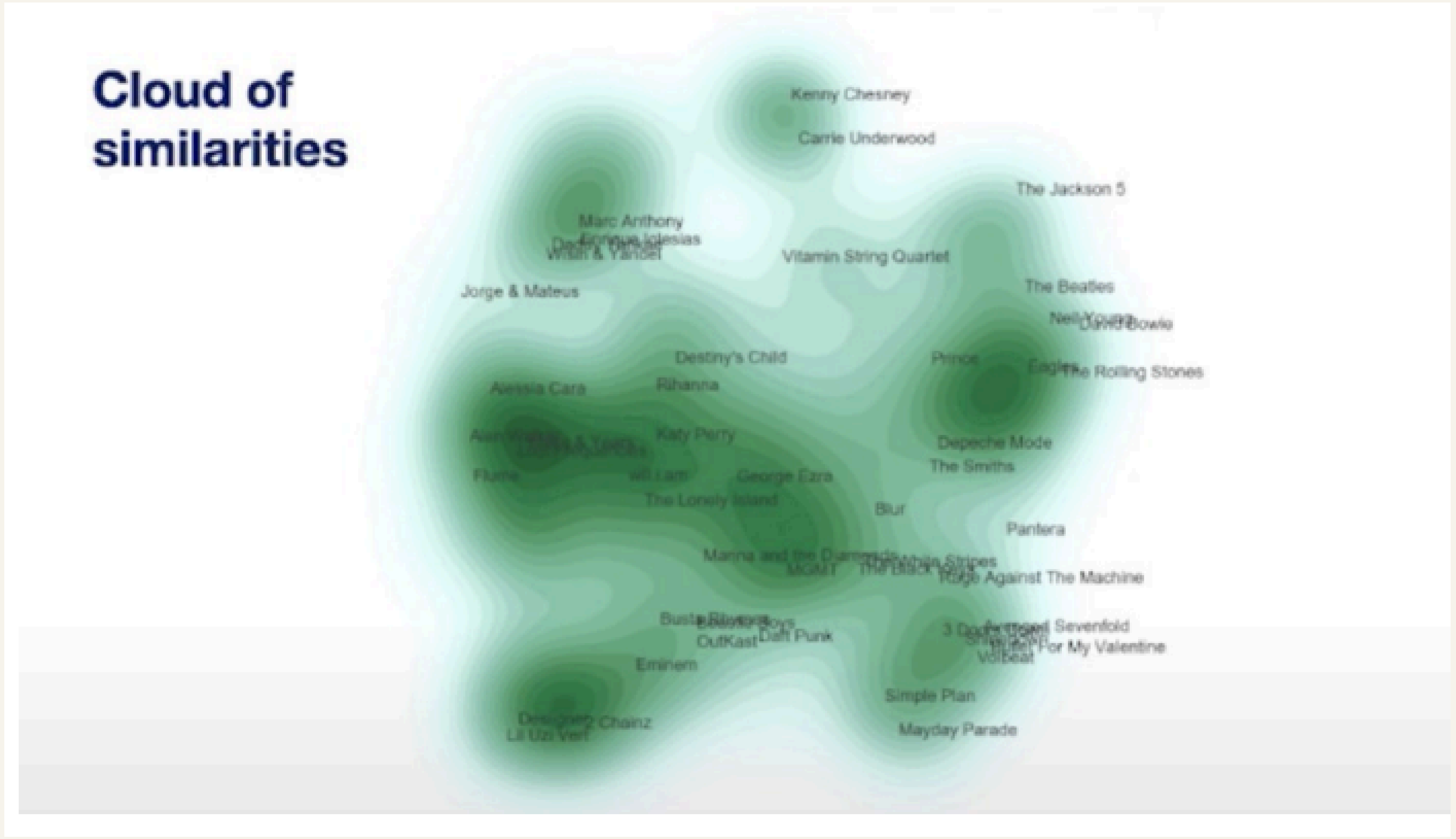
- Data Scraping : The Spotify API restricts the number of songs you can extract at once, making the process very time consuming.
- Biased playlists : Even after gathering the data, the overlapping of songs and people's preferences in making playlists resulted in a irregular dataset.
- Computational challenges : Testing the effectiveness of our implemented models, like random forest and K-Means, demanded significant computational time.
- Parameter Tuning : The K-Nearest Neighbor (KNN) model took a considerable amount of time for parameter tuning.
- Out-of-bounds error : During the training of the random forest model, we encountered 'out of bounds' errors on the leaves, and resolving the issue took a considerable amount of time.

CONCLUSION

Some of the calm songs were clustered separately, while the others overlapped with the cluster majorly consisting of sad songs.

Similarity of audio features to determine the mood is a better parameter of recommendation, rather than labelling songs into different moods.

Clustering on multiple parameters performs better for the objective than any classification algorithm



Spotify Algorithm based on similarities

REFERENCES

1 Patra, B. G., Bandyopadhyay, S., and Das, D. Unsupervised Approach to Hindi Music Mood Classification, 2013.

2 Wallach, J., Corr, B., and Moschitto, M. Mood Classification of Spotify Songs, 2021.

3 Nuzzolo, M. Music Mood Classification, Electrical and Computer Engineering Design Handbook, 2015.

4 GitHub Dataset

CS 361 - Machine Learning | 2024

THANK YOU

Presented By : GPyTer