SPOTIFY SONG PREFERENCE RECOMMENDATION

By : Kanupriya Jain

1 Introduction

In this project, I am exploring a Spotify user's liked and disliked songs dataset, aiming to develop a predictive model using logistic regression to determine song likability based on various features. The process involves initial data preprocessing, handling missing values. Exploratory Data Analysis (EDA) includes visualizations like histograms and count plots, alongside a correlation matrix to identify influential features. K Means Clustering groups songs, and logistic regression is implemented for probability prediction and cost function for evaluation. The project uses pandas, NumPy, and matplotlib for implementation and visualization. The project aims to reveal key factors influencing song likability for Spotify users.

2 Methodology

1. **Dataset Acquisition :**
   Obtain the dataset representing a Spotify user's liked and disliked songs stored in a CSV file. Ensure the dataset includes features such as liveliness, acoustics, and energy.
2. **Data Preprocessing :**
   I will handle Missing Values. Address any missing values in the dataset, employing strategies like imputation or removal.
3. **Data Preparation :**
   (a) *Split Data:* Divide the dataset into training and testing sets to evaluate model performance effectively.
   (b) *Feature Scaling:* Scale all features to maintain uniform scales, preventing dominance of certain features during training.
4. **Exploratory Data Analysis (EDA) :**
   (a) *Histograms and Count Plots:* Visualize the distribution of song features through histograms and count plots to understand their characteristics.
   (b) *Correlation Matrix:* Generate a correlation matrix to identify relationships between different features and assess their impact on song likability.
5. **K Means Clustering :** I will group songs based on features and identify which cluster contains the highest number of liked songs.
6. **Logistic Regression Implementation :** I will create a class for Logistic Regression. In that, I will define the sigmoid function, fit function and predict funtion.
7. **Analysis of Logistic Regression Results:** I will check how well the model is performing by calculating its confusion matrix and defining a function to calculate the accuracy.

3 Data Preprocessing

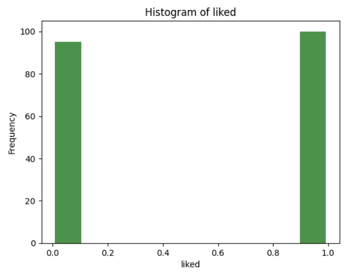
I checked for NaN values and counted non-null values in my dataset and found out that it does not contain such values. I also generated a descriptive statistics for each feature to understand the data better

4 Exploratory Data Analysis

4.1 Histograms

I created histograms for each dataset feature to visually highlight distribution patterns. This helps identify important trends for further modeling and analysis.

Histograms for liked feature is given below -

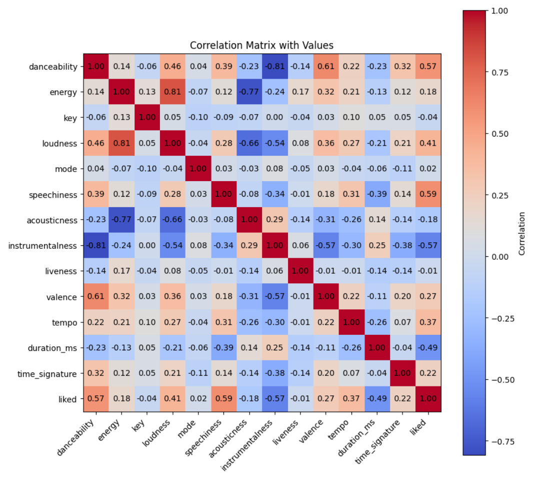


4.2 Like Vs. Disike Count

We can see the dataset contains **100** liked songs and **95** not liked songs.

4.3 Co-relation Matrix

I created a heatmap showing correlation coefficients between features. Coefficients range from -1 to 1, with 1 indicating a perfect positive correlation, -1 a perfect negative correlation, and values near zero suggesting weak or no linear relationship.

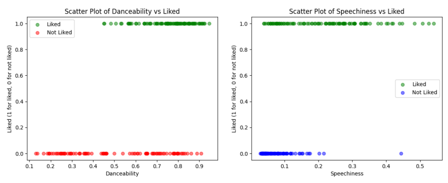


After analyzing the correlation between all features and the target column 'Liked,' we can observe that the user exhibits a preference for songs with:

1. High scores in danceability **(0.57)**, loudness**(0.41)**, and speechiness**(0,59)**
2. Low scores in instrumentalness **(-0.57)** and duration **(-0.49)**

4.3.1 Relationship of liked songs with danceability and speechiness

I created two scatter plots, one comparing liked songs with danceability and the other with speechiness, aiming to analyze and understand the patterns I created two scatter plots for daceability vs. liked on the same axes: The first scatter plot represents liked songs, using green markers . The second scatter plot represents not liked songs, using red markers with an alpha of 0.5.



Observations :

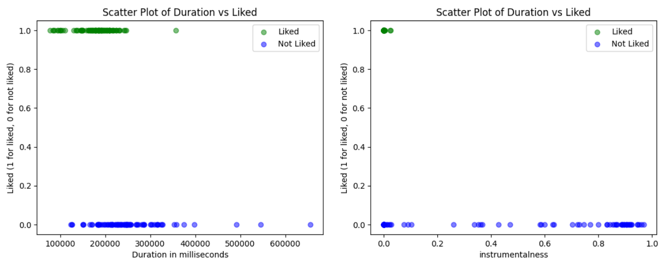
1. Speechiness shows a positive relation with liked songs, but even songs with low speechiness are liked. Low speechiness doesn't guarantee user dislike. To get a clearer view, I will create a countplot combining both features is recommended.
2. I counted number of liked songs with speechiness > 0.3 and danceability > 0.6. There are a total of 22 liked songs exhibiting high danceability and high speechiness, with only one exception that the user did not like. It appears that songs with high scores in both speechiness and danceability are highly probable to be favored by the user.

4.3.2 Relationship of liked songs with instrumentalness and duration

In order to find the relationship between liked songs with instrumentalness and duration, I calculated the average duration of liked songs and songs that are not liked by the user.

The average duration of liked songs is 2.9832048333333336 minutes The average duration of songs not liked by the user is 4.160616315789474 minutes

Liked songs average below 3 minutes, disliked songs exceed 4 minutes. Higher instrumentalness correlates with longer durations. The user prefers shorter, less instrumental songs, as seen in scatter plots



5 KMeans Clustering

I aim to group similar songs in the dataset into clusters and visualize the count of songs within each cluster. Subsequently, I plan to assign names to these clusters based on the average danceability, speechiness, duration and energy values specific to each cluster.

I defined a class called KMeans using NumPy and Pandas.

5.1 Standardizing data

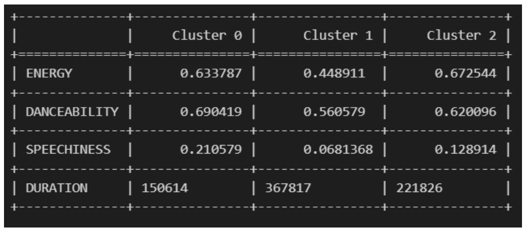
I converted the dataframe to a numpy array, then changed data entries to float for accurate calculations. Now, I'm standardizing the data by subtracting row means and dividing by row standard deviations, ensuring consistent scales for features

Algorithm 1 K-Means Clustering Algorithm

```
**Input:** Data $X$, Number of clusters $K$, Maximum iterations $n\_iters$, Convergence threshold $\epsilon$
**Output:** Cluster labels
Initialize centroids randomly: $C \leftarrow$ random $K$ samples from $X$
Initialize clusters: $clusters\_list \leftarrow$ [[] for _ in range($K$)]
Initialize iteration counter: $iter \leftarrow 0$
**while** $iter < n\_iters$ **do**
    **for** each sample $x$ in $X$ **do**
        Calculate distances to centroids: $distances \leftarrow$ [euclidean_distance($x, c$) for $c$ in $C$]
        Find the index of the closest centroid: $closest\_index \leftarrow \arg\min(distances)$
        Assign $x$ to the closest cluster: $clusters\_list[closest\_index]$.append($x$)
    **end for**
    Save old centroids: $C_{\text{old}} \leftarrow C$
    **for** each cluster $c$ in $clusters\_list$ **do**
        Update centroids: $C[c] \leftarrow$ mean($c$)
    **end for**
    Calculate distance between old and new centroid: $distance \leftarrow \sum_{i=1}^{K}$ euclidean_distance($C_{\text{old}}[i], C[i]$)
    **if** $distance < \epsilon$ **then**
        **Break**                                        ▷ Convergence criterion met
    **end if**
    Increment iteration counter: $iter \leftarrow iter + 1$
**end while**
Assign labels based on cluster memberships
**Return** Cluster labels
```

5.2 Assigning names to clusters

I created a class instance (k1) with three clusters and stored the labels in "ypred1". Using key features (danceability, speechiness, energy, and duration), I computed mean values for each cluster, assigning names based on the feature patterns. Here are the mean values for each cluster:



```
+-------------+--------------+---------------+--------------+
|             |    Cluster 0 |     Cluster 1 |    Cluster 2 |
+=============+==============+===============+==============+
| ENERGY      |     0.633787 |     0.448911  |     0.672544 |
+-------------+--------------+---------------+--------------+
| DANCEABILITY |    0.690419 |     0.560579  |     0.620096 |
+-------------+--------------+---------------+--------------+
| SPEECHINESS |     0.210579 |     0.0681368 |     0.128914 |
+-------------+--------------+---------------+--------------+
| DURATION    | 150614       | 367817        | 221826       |
+-------------+--------------+---------------+--------------+
```

Observations :

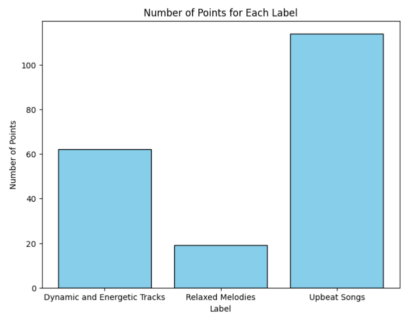
1. We observe that cluster 1 exhibits the lowest danceability, energy, and speechiness among the three clusters. Additionally, it boasts the highest mean duration, typically associated with relaxed songs. Therefore, we can label cluster 1 as ***"Relaxed Melodies."***

2. Cluster 0 stands out with the highest energy, speechiness, and danceability, accompanied by the lowest duration. These characteristics are often found in dance songs. Accordingly, we can name cluster 0 as ***"Dynamic and Energetic Tracks."***

3. Cluster 2 displays moderate values in energy (close to cluster 1), danceability, speechiness, and duration. As a result, we can assign the name ***"Upbeat Songs"*** to this cluster.

5.3 Countplot for the Clusters

We can count the number of songs in each cluster.



We can see that dataset has -

- Relaxed Melodies - 19
- Dynamic and Energetic Tracks - 62
- Upbeat Songs - 114

I calculated the percentage of liked songs in each cluster.

- Percentage of Liked Songs in Relaxed Melodies : 5.263157894736842
- Percentage of Liked Songs in Dynamic and Energetic Tracks : 79.03225806451613
- Percentage of Liked Songs in Upbeat Songs : 43.859649122807014

We can see that user the user prefers the songs with higher energy and dynamism. Now, I wanted to find based on given song features, whether the user will express liking toward the song.For that, I used Logistic Regression.

6 Logistic Regression

Sigmoid Function:

The core of logistic regression is the sigmoid function, $S(z) = \frac{1}{1+e^{-z}}$. It maps any real-valued number to the range $[0, 1]$, providing a smooth transition between the two classes.

Log Loss (Cost Function):

Log loss measures the difference between predicted probabilities and true labels. For a single training example,

$$J(\theta) = -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x)).$$

The goal is to minimize this cost function during training.

Gradient Descent:

Parameters (θ) are updated using gradient descent to minimize the log loss. The update rule:

$$\theta_i = \theta_i - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_i^{(i)}.$$

Convergence Criterion:

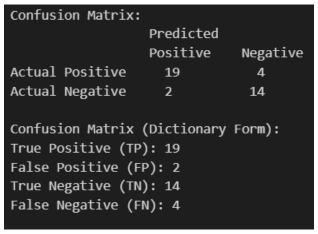
A convergence criterion is implemented to stop training when the change in cost falls below a threshold. This prevents unnecessary iterations when the model has converged.

6.1 Confusion Matrix and Accuracy

I splitted my data into X_train and X_test. I trained my model with X_train and tested it on X_test. I made a confusion matrix that summarizes the performance of a machine learning model on a set of test data. Here is the confusion matrix that shows how well the model has performed -

Algorithm 2 Logistic Regression Algorithm

```
1: Input: Data matrix $X$, Target vector $y$, Learning rate $\alpha$, Maximum iterations $n\_iters$, Convergence threshold $\epsilon$
2: Output: Trained weights $W$ and bias $b$
3: Initialize weights: $W \leftarrow \mathbf{0}$
4: Initialize bias: $b \leftarrow 0$
5: Initialize previous cost: $previous\_cost \leftarrow \infty$
6: for $iteration \leftarrow 1$ to $n\_iters$ do
7:     Compute linear predictions: $linear\_pred \leftarrow X \cdot W + b$
8:     Compute sigmoid activations: $predictions \leftarrow \text{sigmoid}(linear\_pred)$
9:     Compute log loss: $loss \leftarrow -y \cdot \log(predictions) - (1 - y) \cdot \log(1 - predictions)$
10:    Compute mean loss: $current\_cost \leftarrow (loss)$
11:    Compute gradients:
12:        $dw \leftarrow \frac{1}{n\_rows} \cdot X^T \cdot (predictions - y)$
13:        $db \leftarrow \frac{1}{n\_rows} \cdot \sum_{i=1}^{n\_rows}(predictions_i - y_i)$
14:    Update weights and bias:
15:        $W \leftarrow W - \alpha \cdot dw$
16:        $b \leftarrow b - \alpha \cdot db$
17:    if $|previous\_cost - current\_cost| < \epsilon$ then
18:        Break                                    ▷ Convergence criterion met
19:    end if
20:    $previous\_cost \leftarrow current\_cost$
21: end for
22: Return Trained weights $W$ and bias $b$
```



```
Confusion Matrix:
                 Predicted
                 Positive    Negative
Actual Positive     19          4
Actual Negative     2           14

Confusion Matrix (Dictionary Form):
True Positive (TP): 19
False Positive (FP): 2
True Negative (TN): 14
False Negative (FN): 4
```

Accuracy of the model will be given by -

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Number of Samples}}$$

After evaluating the logistic regression model, the calculated accuracy is 0.84. This implies that the model correctly predicted the outcome for approximately 84% of the instances in the test set.

7 Conclusion

In summary, this project helped in observing notable patterns in their preferences of a Spotify user . The correlation analysis highlighted the user's inclination towards songs with elevated danceability, loudness, and speechiness, while showing a tendency to avoid lengthy compositions and those with substantial instrumental elements. Through KMeans clustering, we identified three distinct clusters, each representing a unique music style. Further, employing logistic regression allowed us to build a predictive model, achieving an accuracy of 84.6% in classifying liked songs. These findings not only help us in understanding the user's musical tastes but also hold implications for refining recommendation systems and personalizing the user experience on the platform.

8 References

- https://www.kaggle.com/code/wjkwok/spotify-song-features-analysis
- **Data :** https://www.kaggle.com/datasets/bricevergnou/spotify-recommendation
- https://www.kaggle.com/code/chaitalijawale/spotify-eda-ml-model-deepl
- https://heena-sharma.medium.com/logistic-regression-7773c76522d6
- https://www.geeksforgeeks.org/confusion-matrix-machine-learning/
- https://github.com/AssemblyAI-Examples/Machine-Learning-From-Scratch/blob/main/03%20Logistic%20Regression/LogisticRegression.py
- https://github.com/AssemblyAI-Examples/Machine-Learning-From-Scratch/tree/main/01%20KNN