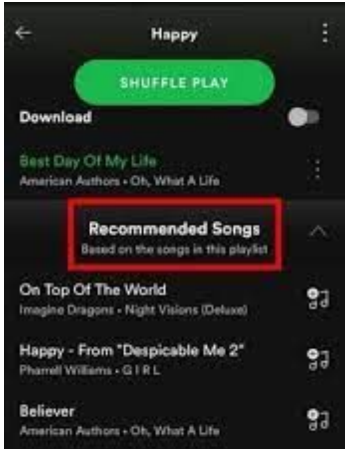
IT350 - Data Analytics

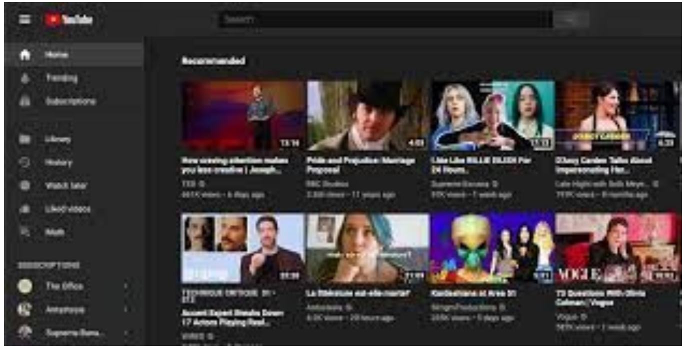
Automatic Playlist Recommendation System

Saumya Karri	181ME273
Anshuman Sinha	181EE209
Sathvika B. Mahesh	181CV141

Introduction

- A playlist is simply a sequence of tracks intended to be listened to together
- Automatic music playlist continuation is a form of the more general task of sequential recommendation
- Danceability, Energy, Loudness, Instrumentalness, Liveliness, Tempo
- Broader usage in music softwares like Spotify and in any data-streaming platforms like YouTube.





Methodology

Data Extraction

- One slice of dataset chosen randomly - playlist 20000 to 20999
- Attributes like album name, danceability, energy, acousticness etc.
- Creating a dataframe which gives trackwise features

Data cleaning

- Dropped non-quant and null values
- Normalized using z-score normalization
- Found correlation between features using heatmap
- Dropped heavily correlated attributes

Data modelling

- Used 6 classification models on the data
- Predicted the ratings of each song
- Calculated accuracy scores for each model

Implementation

Classification models

1. Linear regression
2. Logistic regression
3. Random forest regression
4. Random forest classification
5. Gradient Boosting regression
6. K nearest neighbour classification

Linear regression is a supervised Machine Learning approach for predicting continuous variables.
Logistic regression is a supervised Machine Learning algorithm that helps in binary classification.
Random Forest regression/classification is an ensemble approach that uses several decision trees and a technique called Bootstrap and Aggregation, also known as bagging, to solve both regression and classification problems. Classification transfers the input data item to a set of discrete labels, regression converts the input data item into continuous real values.
Gradient Boosting builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions.
K-Neighbors classification is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions).

Snapshots

The 'number of tracks' characteristic of each playlist

```
# Summary Statistics for 'num_tracks'
print("Summary Statistics for number of tracks\n")
# Using in-built describe method of pandas that calculate a summary of given statistics
describe = df['num_tracks'].describe()
print(str(describe)+'\n')

# Using functions 'var' and 'ptp' to calculate variance and range that the 'decsribe' function did not give us
print('Variance\t' + str(df['num_tracks'].var()))
print('Range\t\t' + str(max(df['num_tracks']) - min(df['num_tracks'])))
```

[28] ✓ 0.9s

```
Summary Statistics for number of tracks

count    1000.000000
mean       68.101000
std        53.532612
min         5.000000
25%        26.000000
50%        52.000000
75%        95.000000
max       246.000000
Name: num_tracks, dtype: float64

Variance        2865.740539539543
Range           241
```

The 'duration in ms' characteristic of each playlist

```
# Summary Statistics for 'duration'
print("Summary Statistics for duration\n")
# Using in-built describe method of pandas that calculate a summary of given statistics
describe = df['duration_ms'].describe()
print(str(describe)+'\n')

# Using functions 'var' and 'ptp' to calculate variance and range that the 'decsribe' function did not give us
print('Variance\t' + str(df['duration_ms'].var()))
print('Range\t\t' + str(max(df['duration_ms']) - min(df['duration_ms'])))
```

[30] ✓ 0.5s

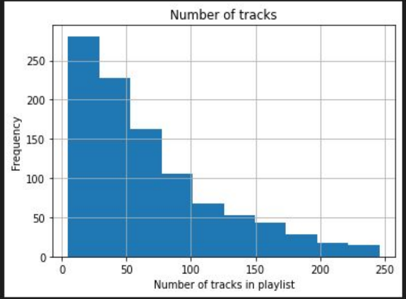
```
Summary Statistics for duration

count    1.000000e+03
mean     1.604941e+07
std      1.280911e+07
min      1.044377e+06
25%      6.118938e+06
50%      1.228655e+07
75%      2.234913e+07
max      8.309966e+07
Name: duration_ms, dtype: float64

ance          164073286096730.56
e             82055288
```

```
NTracksHist = df['num_tracks'].hist()
xlabel = NTracksHist.set_xlabel("Number of tracks in playlist")
ylabel = NTracksHist.set_ylabel("Frequency")
title = NTracksHist.set_title("Number of tracks")
```

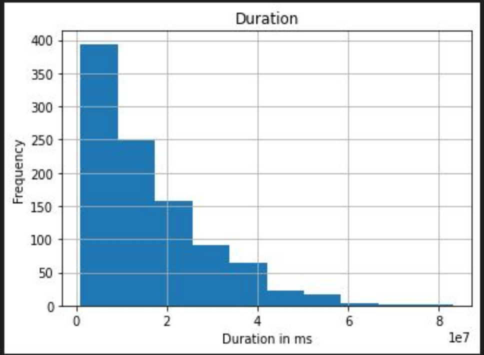
[29] ✓ 1.3s



The 'duration in ms' characteristic of each playlist

```
Duration = df['duration_ms'].hist()
xlabel = Duration.set_xlabel("Duration in ms")
ylabel = Duration.set_ylabel("Frequency")
title = Duration.set_title("Duration")
```

[31] ✓ 0.5s

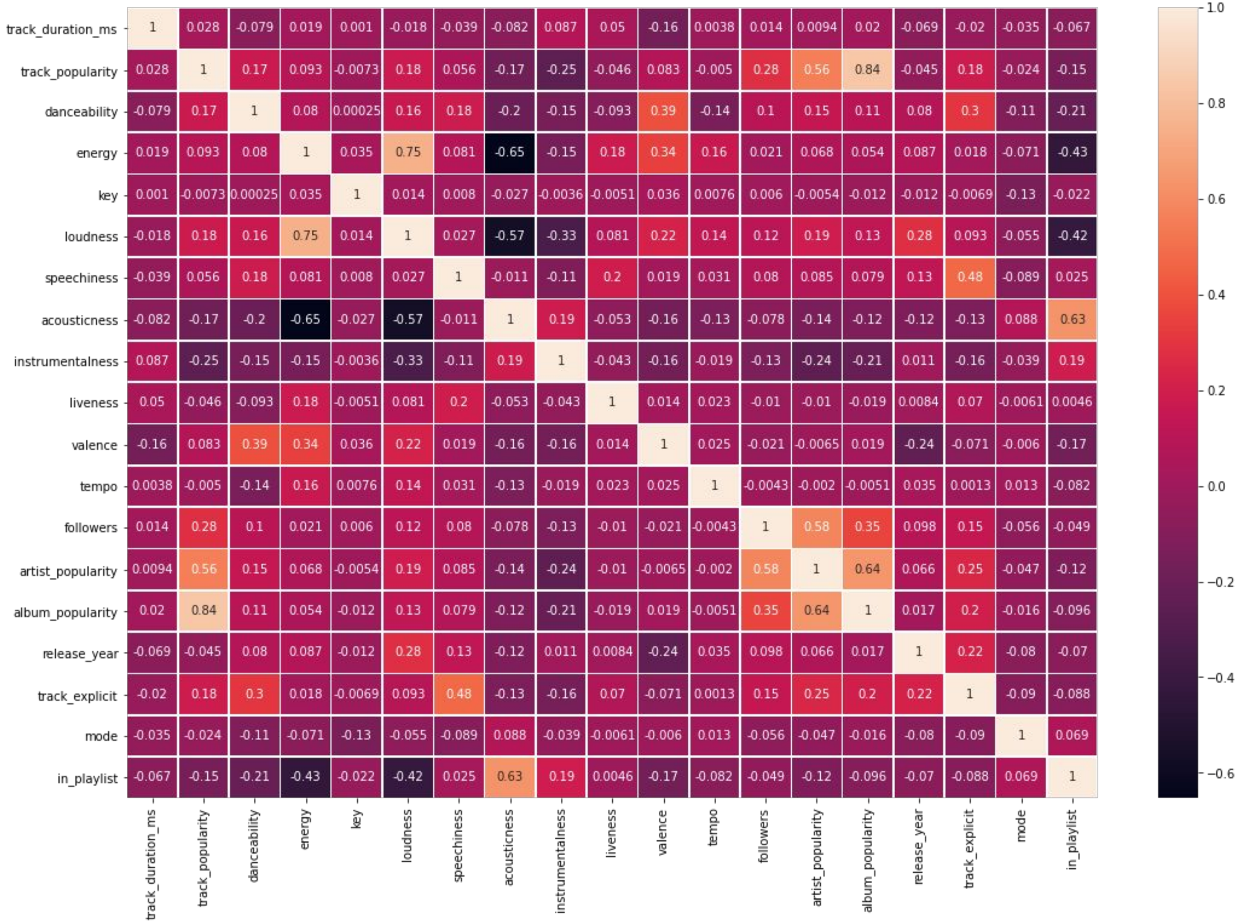


Snapshots

Track Feature

Correlation Heat Map

This graph shows the correlation among the 16 features per song of the chosen set of 1000 playlists.



Results & Analysis

Classification method	Accuracy scores	Confusion matrix
Logistic Regression	0.95589	[[5749 120] [169 373]]
Random Forest Classification	0.99988	[[5869 0] [8 534]]
Gradient Boosting Regressor	0.973254	[[5869 0] [8 534]]
K-Neighbor Classification	0.96064	[[5723 146] [154 388]]

10 fold cross-validation with CV score **0.95492**

Literature Survey

- *"Current Challenges and Visions in Music Recommender Systems Research" M.* Schedl, H. Zamani, C.-W. Chen, Y. Deldjoo, M. Elahi.
    - Paper link: https://arxiv.org/pdf/1710.03208.pdf
    - Gives a broad definition of the problem and provides many new scopes and potential innovations for further research in Music Recommender Systems.
- *"An Analysis of Approaches Taken in the ACM RecSys Challenge 2018 for Automatic Music Playlist Continuation"* H. Zamani, M. Schedl, P. Lamere, C.-W. Chen.
    - **Base Paper** link: https://arxiv.org/pdf/1810.01520.pdf
    - Lays out a complete summary of important metrics and approaches to solve the problem of Automatic Music Playlist Continuation.

Thank you