FOR-TUNE TELLER

Bakhtiyar Syed
Aakash KT
Saurav Malani
Adarsh Pal Singh

MOTIVATION

1. The impact and value that the ability of predicting a song's popularity even before its release can help artists fine-tune their composition.

2. Offers the music advertisment companies assistance in investing in a particular artist/song that can generate for them the most success.

GOAL

→Development of a learning algorithm that classifies the song into 3 categories- *Highly Popular*, *Popular* and *Unpopular* based on the musical features and other song metrics.

→To mitigate the effects of unwanted factors as much as possible by carefully choosing the song features and comparing different learning algorithms for evaluation of their performance.

TIMELINE OF THE PROJECT

ESTIMATED TIMELINE

1 - 8 Nov'

DATA COLLECTION
AND
FEATURE EXTRACTION

9 - 14 Nov'

IMPLEMENT BASELINE
CLASSIFIERS

14 - 26 Nov'

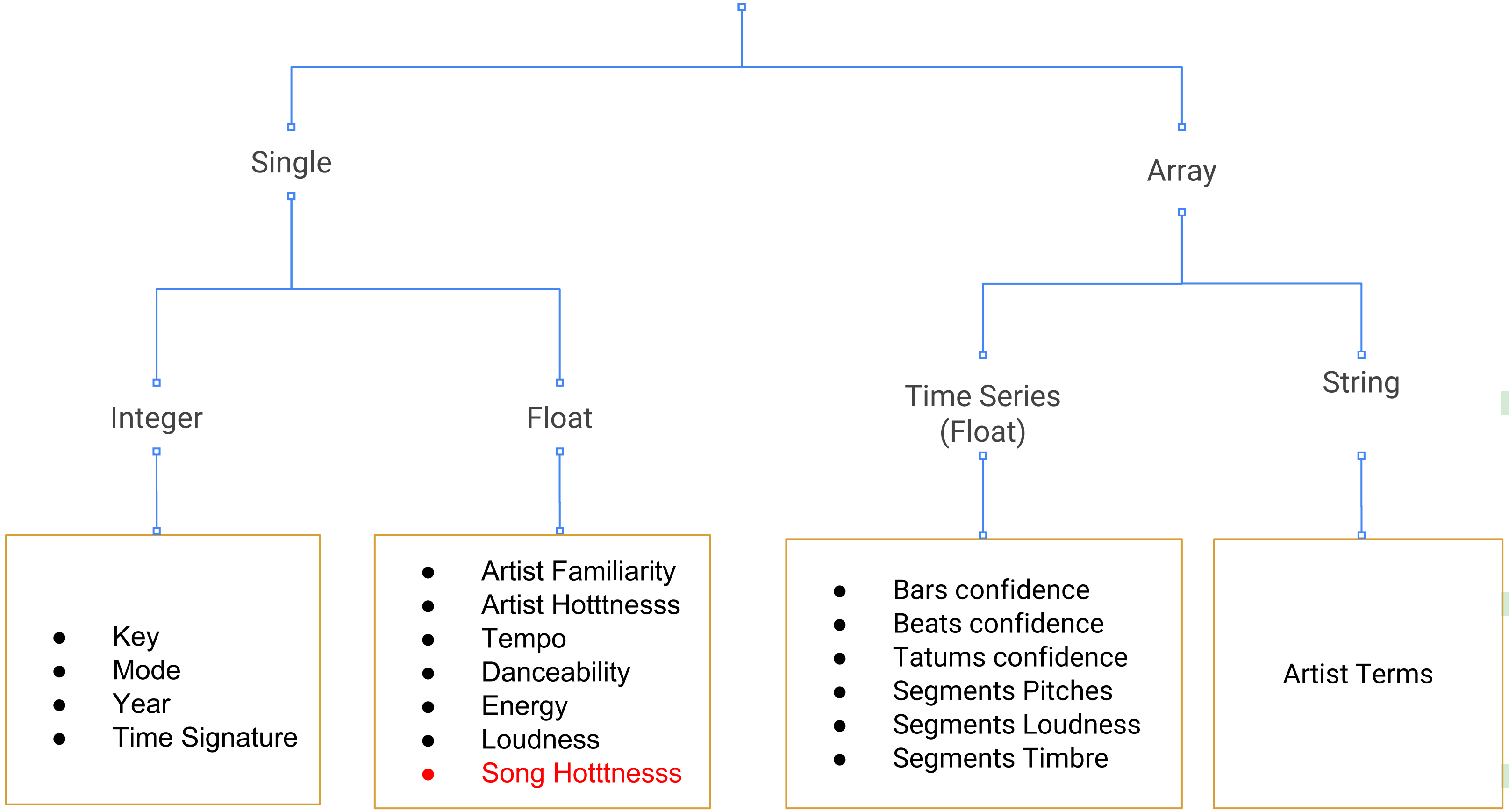
FUTURE
ENHANCEMENTS

Dataset Formation

Million Song Dataset (MSD)

- Collection of audio features and metadata for ~ 1,000,000 songs.
- 24 musical features.
- Full dataset ~280 GB.
- MSD Subset available with 10k songs at ~1.8GB.

MSD Features



- Single
  - Integer
    - Key
    - Mode
    - Year
    - Time Signature
  - Float
    - Artist Familiarity
    - Artist Hotttnesss
    - Tempo
    - Danceability
    - Energy
    - Loudness
    - Song Hotttnesss
- Array
  - Time Series (Float)
    - Bars confidence
    - Beats confidence
    - Tatums confidence
    - Segments Pitches
    - Segments Loudness
    - Segments Timbre
  - String
    - Artist Terms

Problems with MSD

- Features have multiple data types- Int, Float, String Array, Float Array (Time Series).
- Missing Values.
- Data of each song stored in a separate HDF file.
- HDF files stored in recursive folders.
- Outdated dataset.

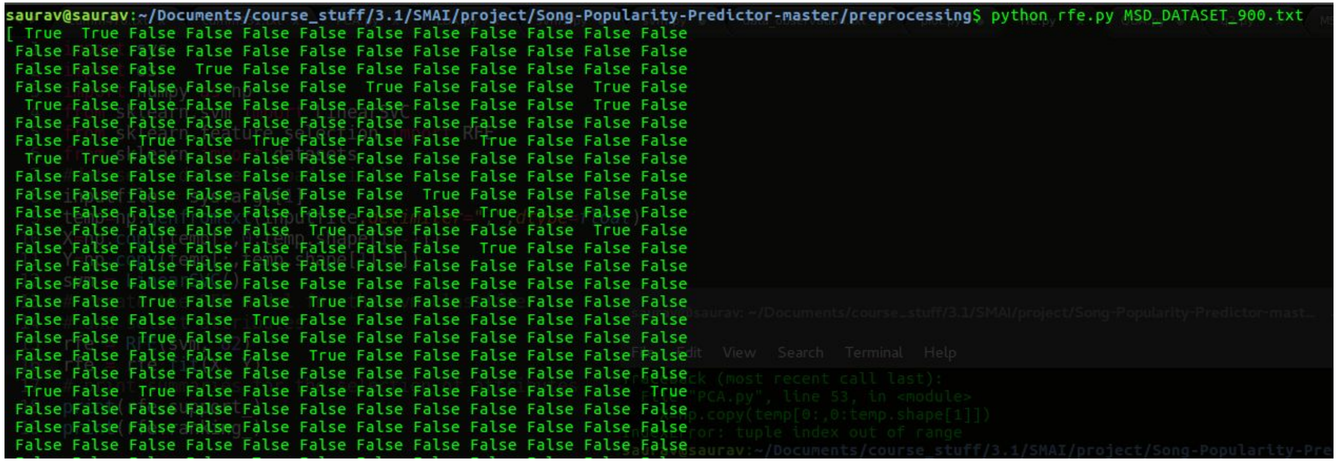
Our Approach

- Int/Float Features taken as is.
- BoW formed with String Arrays and appended as Features.
- Time Series data: Mean and Variance taken as features.
- Songs with missing values ignored (10k reduced to ~3k).
- Our Python wrapper visits each folder recursively to extract data from HDF files and outputs a single CSV file.

Playing with Dimensions!

RFE

Starts by evaluating the use of all features and incrementally removes features until the model is optimized.

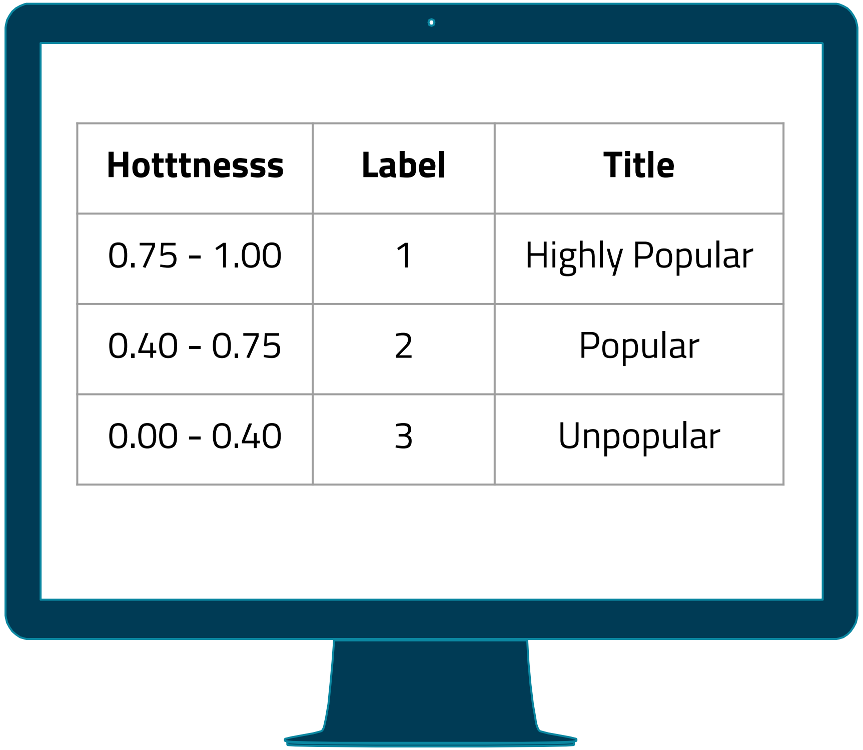


```
saurav@saurav:~/Documents/course_stuff/3.1/SMAI/project/Song-Popularity-Predictor-master/preprocessing$ python rfe.py MSD_DATASET_900.txt
[ True  True False False False False False False False False False False
 False False False False False False False False False False False False
 False False False  True False False False False False False False False
 False False False False False False  True False False False  True False
  True False False False False False False False False False  True False
 False False False False False False False False False False False False
 False False  True False  True False False False  True False False False
  True  True False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False  True False False False False
 False False False False False False False False  True False False False
 False False False False False  True False False False False  True False
 False False False False False False False False  True False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False  True False False  True False False False False False False
 False False False False  True False False False False False False False
 False False  True False False False False False False False False False
 False False False False False  True False False False False False False
 False False False False False False False False False False False False
  True False  True False False False False False False False False  True
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
```

Label Description

DATA LABELS

Song Hotttnesss measure used to determine Song Popularity



Hotttnesss	Label	Title
0.75 - 1.00	1	Highly Popular
0.40 - 0.75	2	Popular
0.00 - 0.40	3	Unpopular

Methods

Linear Discriminant Analysis

- Naive Bayes Generative learning algorithm

$P(y|x) = P(x|y)\, P_{prior}(y) \sum_{g \in Y} P(x|g)\, P_{prior}(g)$

- Scale very easily to massive data sets.
- As most of our features are real valued an extracted from the audio signal of the song, we made a fairly reasonable assumption that the feature vector has a multivariate normal distribution. GDA is known to be asymptotically efficient if the feature vector is Gaussian.

Quadratic Discriminant Analysis

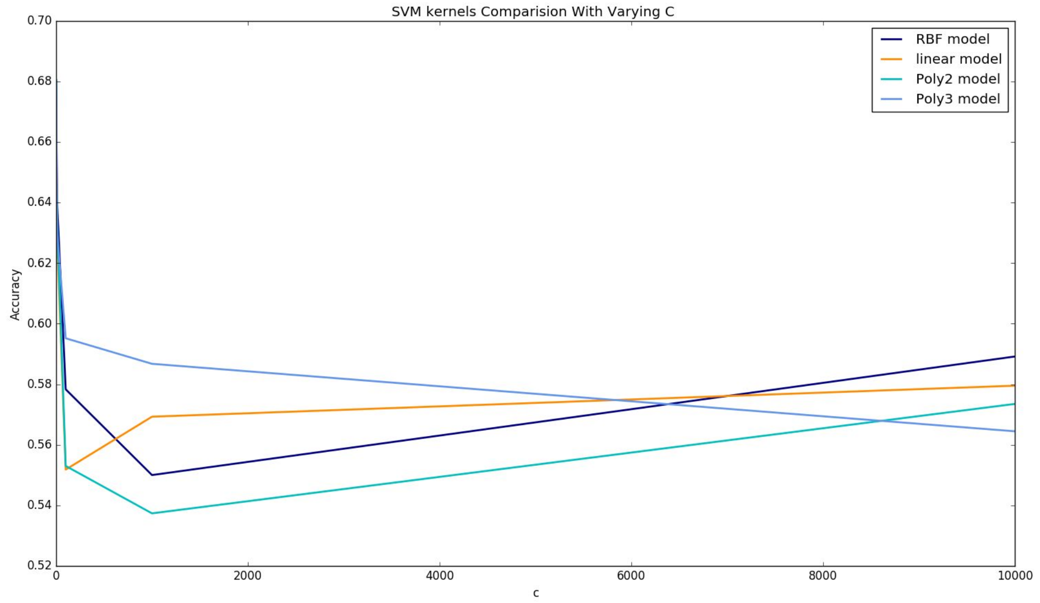
In our second variation of GDA, we will use different covariance matrices for each class. Therefore, all parameters would be updated in the same way as before except Σ, which would now be updated separately for each class.

Multi class classification problem (SVM)

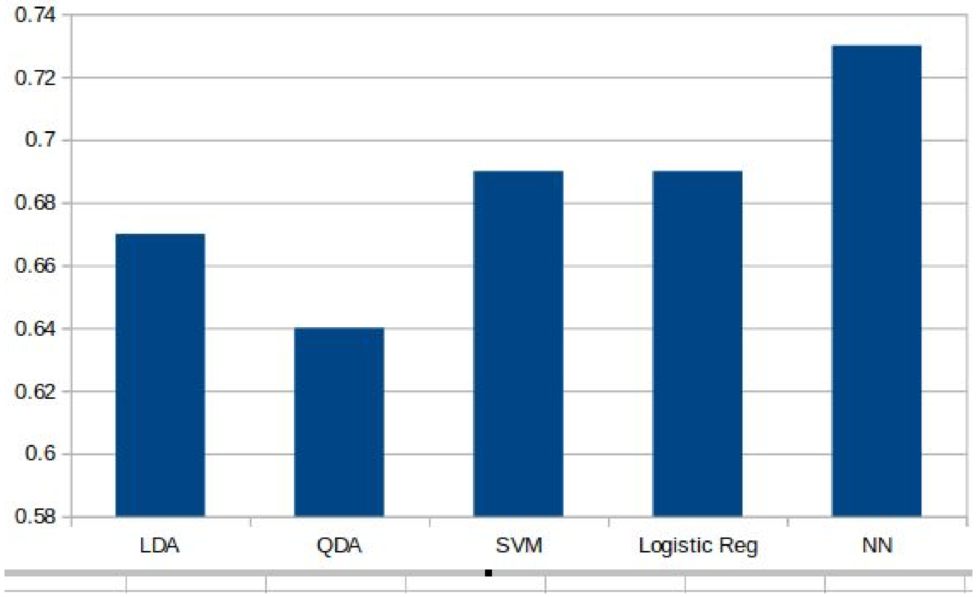
- Essentially, minimise the following to get separating hyperplane.
- One v/s all classifier

$$
\begin{aligned}
\min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\
\text{subject to} \quad & y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \\
& \xi_i \geq 0.
\end{aligned}
$$

SVM Logistics



Test Accuracy



ENHANCEMENTS

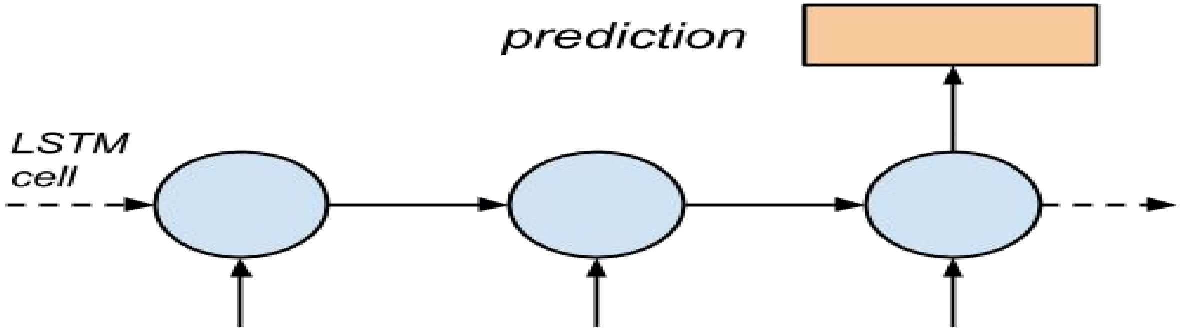
Time-Series data

- Since musical compositions can be arranged sequentially, we considered the possibility of using neural networks which can use sequential data for classification.

- **Inspiration from Language Models:** Since RNN uses sequential information to capture valuable information for natural language, we planned to use sequential information from the music data and using sequential NNs to solve our problem.

Time-Series data

LSTM: (Long short-term Memory) Network:



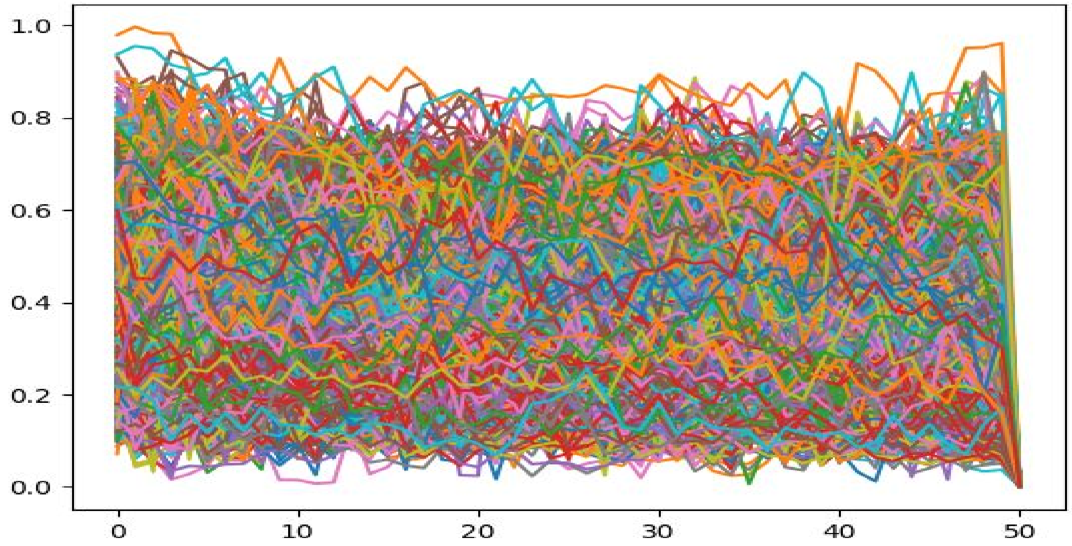
Musical Features

Sequential features that were helpful to us:

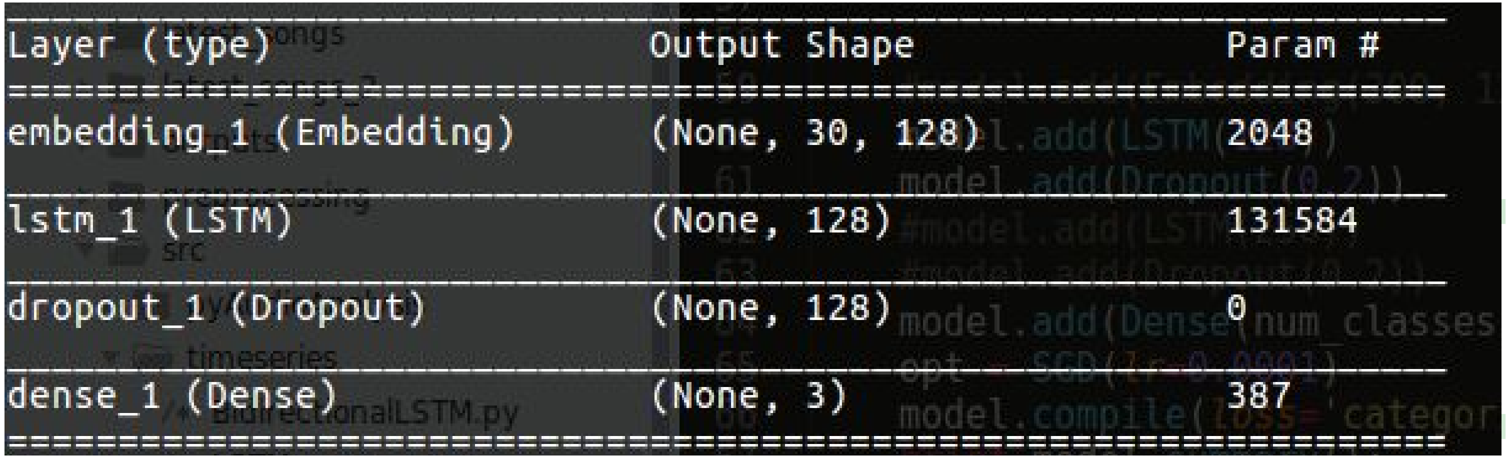
1. Bars confidence
2. Beats confidence
3. Segment confidence
4. Tatums confidence

We created a sequential array of 50 units from the above data to feed into the LSTM

Window feature distribution



Summary of the Model



```
________________________________________________________________________
Layer (type)                 Output Shape              Param #
========================================================================
embedding_1 (Embedding)      (None, 30, 128)           2048
________________________________________________________________________
lstm_1 (LSTM)                (None, 128)               131584
________________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
________________________________________________________________________
dense_1 (Dense)              (None, 3)                 387
========================================================================
```

Results

We split the data into 60-20-20 (Train, Validation, Test)

Validation Accuracy: 62 %

Test Accuracy: 51.38 % (on unseen data)

SOME GARAGE WORK

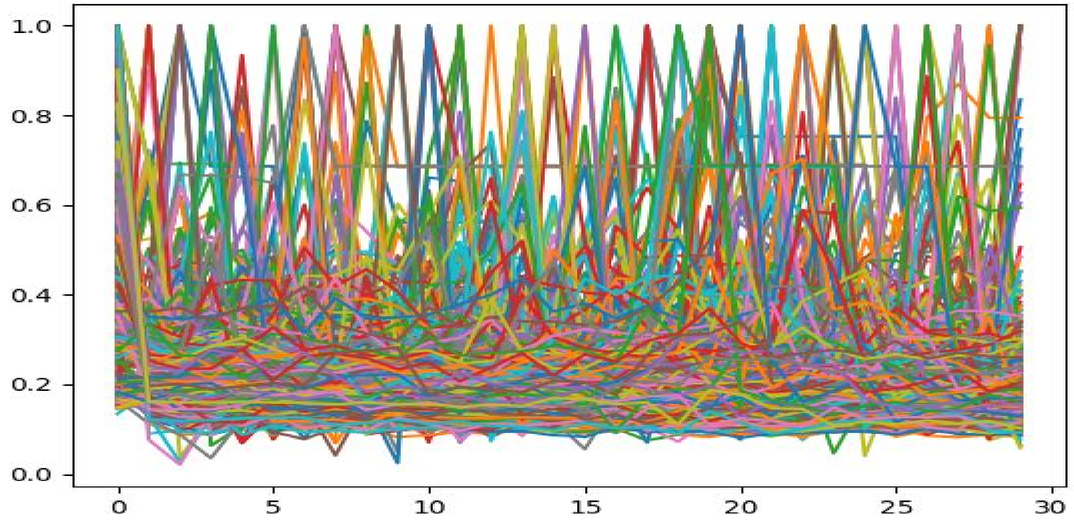
Motivation

The power of classifying sequential data gave us a huge boost to explore a few new things that we found interesting enough to give a shot at.

Since the MSD dataset constrained us, we decided to use the latest song set from Youtube and to capture its compositional features to try and predict whether a song is popular or not.

Our metric here was the play count (no. of times the song was played).

Window Feature distribution



Experiments

We used PyAudioAnalysis to capture the compositionality features like zero crossing rate, chroma vector , etc. (features at https://github.com/tyiannak/pyAudioAnalysis/wiki/3.-Feature-Extraction)

Our dataset formation was again using the sequential data and using our garage-defined A-Feature, which used averaging over all the formed features to form our time-series data.

Experiments

We extracted 889 songs and applied A-feature to get the time-series data.

Again, splitting our data into 90-10 & obtained the following results:

Validation Acc: 41.57%

On entirely new data downloaded later

Test Acc: 35.42 %

THANKS!

Any questions?