Spotify Chart Trend Analysis

1. Introduction

The digital music industry has been evolved over the period and there is a lot of technology progression that has changed the audience' way of listening to music and listen over trends. Nowadays we are able to search for music of our choice, look for a particular collection and even get recommendations for playlists.

Considering this, there is a vast amount of research work is going on, but the research regarding predicting top songs and detecting its characteristics is still underrated. The top songs are connecting to the majority of the audience based on their particular set of features that makes them stand out from the rest of the songs. We can use these particular set of features, process them with machine learning and find the pattern associated which can be used to the prediction of the song to be hit or not.

Spotify has truly separated themselves through leveraging data for personalizing the experience of listening to and discovering music. From its machine-learning and data sifting technology, Spotify analyzes your listening habits and builds out customized recommendations, such as playlists and music suggestions based on the genres and artists you're listening to regularly.

This is most exemplified, however, through the immensely popular Discover Weekly playlist. This playlist is refreshed every Monday with 30 new songs you've most likely never heard of but will probably love because they are chosen based on your most recent listening habits. For example, if October by PRXZM, a song with less than 1,000 views on YouTube, appears on your Discover Weekly playlist, Spotify's technology has deemed this song a worthy candidate for your potential discovery because you were frequently listening to mellow electronic music.

Recent studies in the field of machine learning have tried to work on this problem whether machine learning should be used to predict the hit songs or not. The main purpose of our project is to explore more about this problem and come with algorithms which satisfies the problem statement.

We are working on Spotify (an audio streaming platform) songs list of 2017 and predicting which songs will be on tomorrow's top charts and popularity of the songs.

1.1 Purpose and Goal

To start with, we have taken the dataset from Kaggle and analyzed different music tracks. Each song is having a different set of features. Different machine learning algorithms are then used and applied on to dataset and results have been compared.

Thus the main goal of this project is to check whether machine learning algorithms are used to predict the top songs and with what accuracy it can be done.

2. Background

2.1 Machine learning algorithms

In this work we are focusing on supervised learning, since we are working with a labelled dataset. The task is to construct a model to predict the top list songs that would appear in the charts. For this reason, we picked common machine learning techniques and ARIMA model to predict the same.

2.1.1 Logistic Regression

Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables

2.1.2 Random Forest

Random forests or random decision forests are learning methods for classification, regression and other tasks, that operate a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

2.1.3 Decision Tree Classifier

Decision Tree Classifier is a simple and widely used classification technique. It applies a straightforward idea to solve the classification problem. Decision Tree Classifier poses a series of carefully crafted questions about the attributes of the test record. Each time it receives an answer, a follow-up question is asked until a conclusion about the class label of the record is reached. The decision tree classifiers organized a series of test questions and conditions in a tree structure

2.1.4 K-Nearest Neighbours

K-Nearest Neighbours classification implements learning based on the K nearest neighbours of each query point. This method is a type of non-generalizing learning since it only stores instances of the training data, it does not create an internal model for generalizing the data. The classification is evaluated from a majority vote of the nearest neighbours of each query point, and each point is assigned to the class which is the most common among its neighbours. The choice of the value of k is dependent on the dataset: if k is set to a low value, the point is assigned to a class with only a few neighbours, and if k is a high value, then the effect of noise is suppressed and the classification is of a more general form

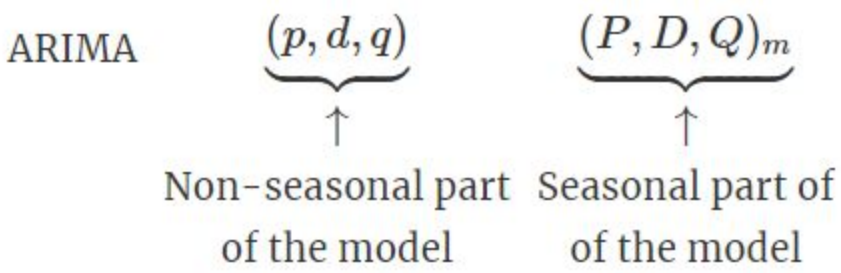
2.1.5 Arima Model

Introduction of ARIMA

ARIMA models provide another approach to time series forecasting. Exponential smoothing and ARIMA models are the two most widely used approaches to time series forecasting, and provide complementary approaches to the problem. While exponential smoothing models are based on a description of the trend and seasonality in the data, ARIMA models aim to describe the autocorrelations in the data.

SEASONAL ARIMA MODEL

A seasonal ARIMA model is formed by including additional seasonal terms in the ARIMA models we have seen so far : It is written as follows:



where m= number of observations per year. We use uppercase notation for the seasonal parts of the model, and lowercase notation for the non-seasonal parts of the model.

ACF/PACF(Auto Correlation / Partial Auto Correlation)

The seasonal part of an AR or MA model will be seen in the seasonal lags of the PACF and ACF. For example, an ARIMA(0,0,0)(0,0,1)

model will show:

- a spike at lag 12 in the ACF but no other significant spikes;
- exponential decay in the seasonal lags of the PACF (i.e., at lags 12, 24, 36, …).

Similarly, an ARIMA(0,0,0)(1,0,0)

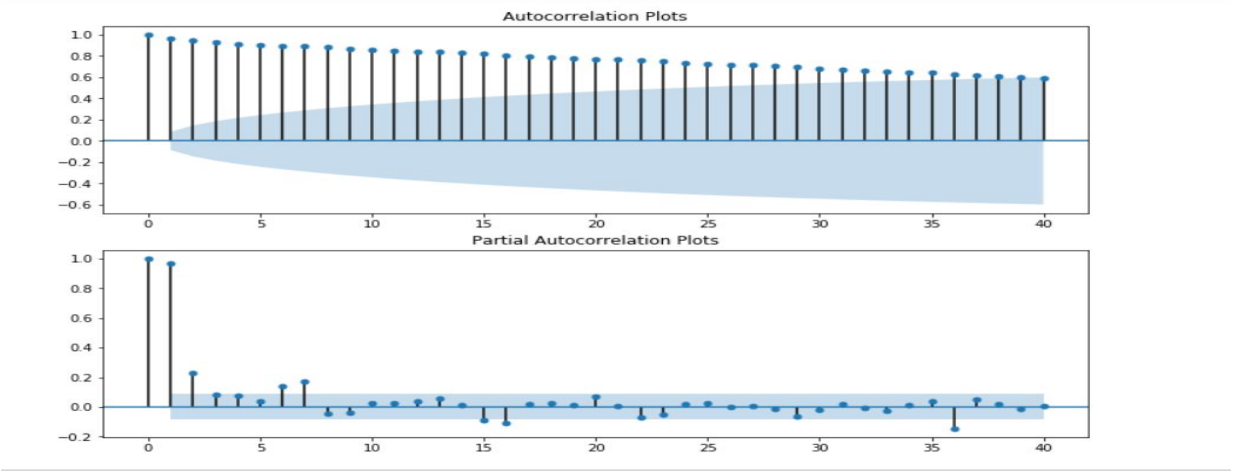
model will show:

- exponential decay in the seasonal lags of the ACF;
- a single significant spike at lag 12 in the PACF.

In considering the appropriate seasonal orders for a seasonal ARIMA model, restrict attention to the seasonal lags.

The modelling procedure is almost the same as for non-seasonal data, except that we need to select seasonal AR and MA terms as well as the non-seasonal components of the model.

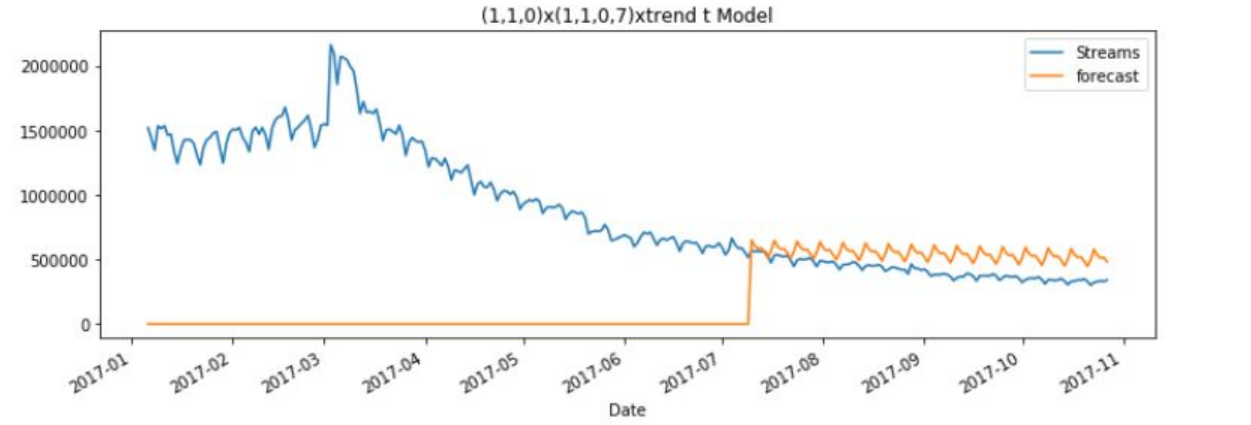
```
f= plt.figure(figsize = (12,8))
ax1= f.add_subplot(211)
f = plot_acf(closer, lags=40, ax=ax1, alpha = 0.1)
ax1.set_title('Autocorrelation Plots')
ax2 = f.add_subplot(212)
f = plot_pacf(closer, lags=40, ax=ax2, alpha = 0.1)
ax2.set_title('Partial Autocorrelation Plots')
plt.show()
```



A seasonal ARIMA model uses differencing at a lag equal to the number of seasons (s) to remove additive seasonal effects. As with lag 1 differencing to remove a trend, the lag s differencing introduces a moving average term. The seasonal ARIMA model includes autoregressive and moving average terms at lag s.

Seasonal ARIMA models can potentially have a large number of parameters and combinations of terms. Therefore, it is appropriate to try out a wide range of models when fitting to data and choose a best fitting model using an appropriate criterion

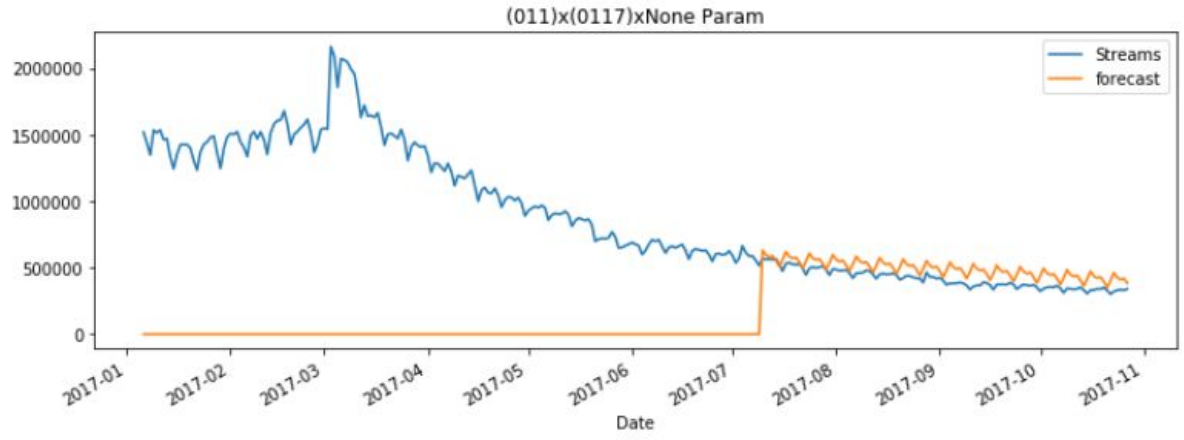
```
X_train, X_test,row_info = get_X(shape_of_you)
#Current Modl
mdl1 = SARIMAX(X_train, trend = 'n', order=(1,1,0), seasonal_order=(1,1,0,7))
mdl_shape(mdl1,row_info)
```



The implementation is called SARIMAX instead of SARIMA because the "X" addition to the method name means that the implementation also supports exogenous variables.

Exogenous variables are optional can be specified via the "exog" argument.

```
X_train, X_test, row_info = get_X(shape_of_you)
#Current Modl
mdl2 = SARIMAX(X_train, trend = 'n', order=(0,1,1), seasonal_order=(0,1,1,7))
mdl_shape(mdl2, row_info,name = '(011)x(0117)xNone Param')
```



The trend and seasonal hyperparameters are specified as 3 and 4 element tuples respectively to the "*order*" and "*seasonal_order*" arguments.These elements must be specified.

3. Method

The machine learning workflow is separated into 4 parts: extract data, preprocess data, model building, and prediction

3.1 Extract data/ Specifications

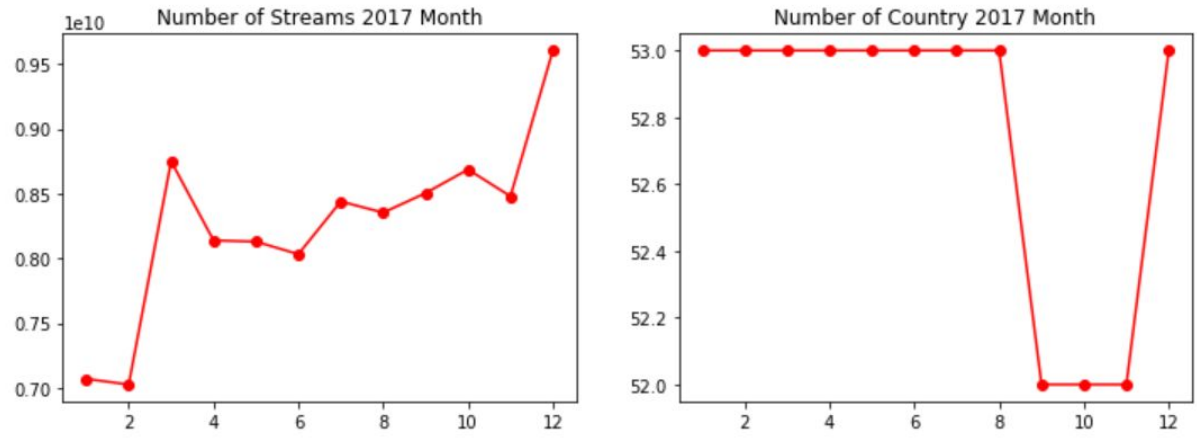
The datasets we used in this research contains the day by day positioning of the 200 most listened tunes in 53 nations from 2017 by Spotify clients. It contains in excess of 2 million lines, which includes 6629 artists, 18598 songs for an all-out check of one hundred five billion streams tally. The information ranges from 1st January 2017 to 31st December 2017 and will stay up with the latest on following adaptations. It has been gathered from Spotify's provincial graph information.

Other dataset includes the names, artists, Spotify URIs and audio features of tracks from the Top Tracks of 2017 playlist. It covers attributes such as danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo, duration, time_signature.

3.2 Preprocess data

The preprocessing part of the workflow is cleaning the data, we removed null values. To avoid this issue, we decided not to use complete datasets, but instead use 2 different datasets for features. We had to make sure, we are not including any duplicate data that could affect the model.

We plotted number of streams and country to the months of 2017, to check the distribution of users.



These two graphs unveil the number of users increased! Spotify expand their area to music industry over and over.

The dataset is split up in two sets: a training set and a test set. The training set is the part of the dataset, consisting of 80% of the data. It is used to train the model so it can make predictions on the remaining 20% of the test data.

3.3 Model building

We used 6 different machine learning algorithms and ARIMA to get the optimal model. Initially we started with ARIMA, to predict the time series forecasting. The algorithms we used are Decision Trees, Random Forest, Logistic Regression, K-Nearest neighbors classifier, Linear Support Vector Classification and XGBoost.

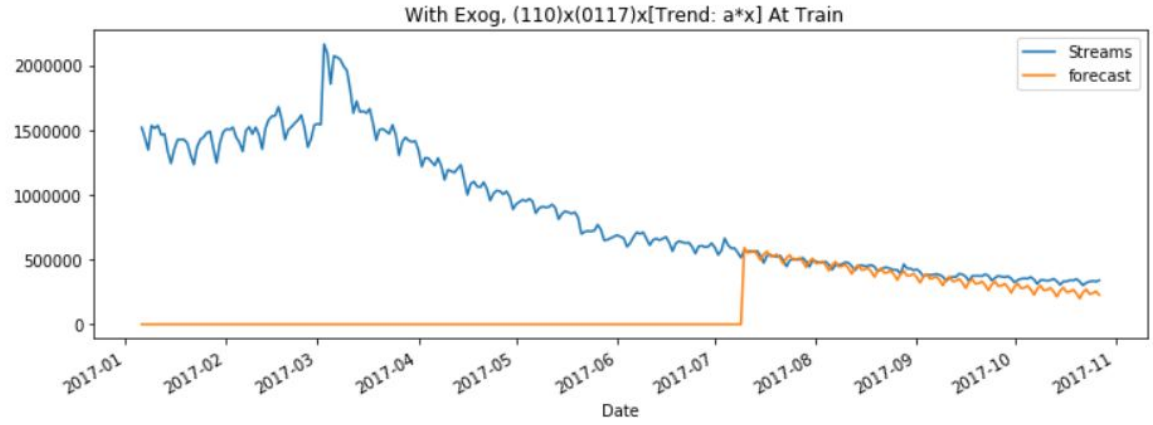
The models were implemented to check the popularity of the songs based on the features, with top 25% of songs being popular and bottom 75% as not popular.

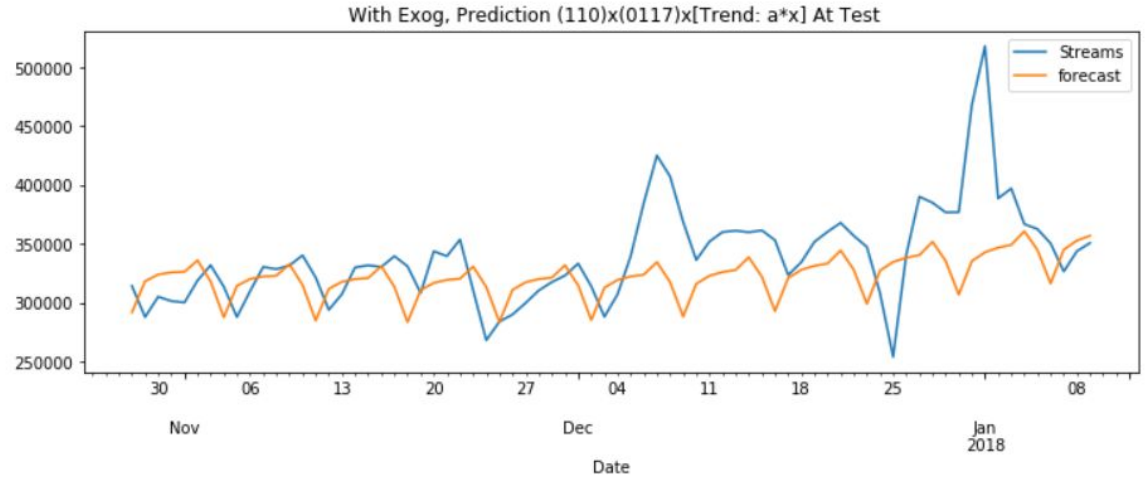
ARIMA model was implemented to predict the streams of the songs, i.e the most trending songs in 2017.

To build the model, we utilized the trained dataset explained previously. In light of training data, the calculation made a grouping rule. This rule, makes the model ready to order new models and to predict whether or not a song is popular.

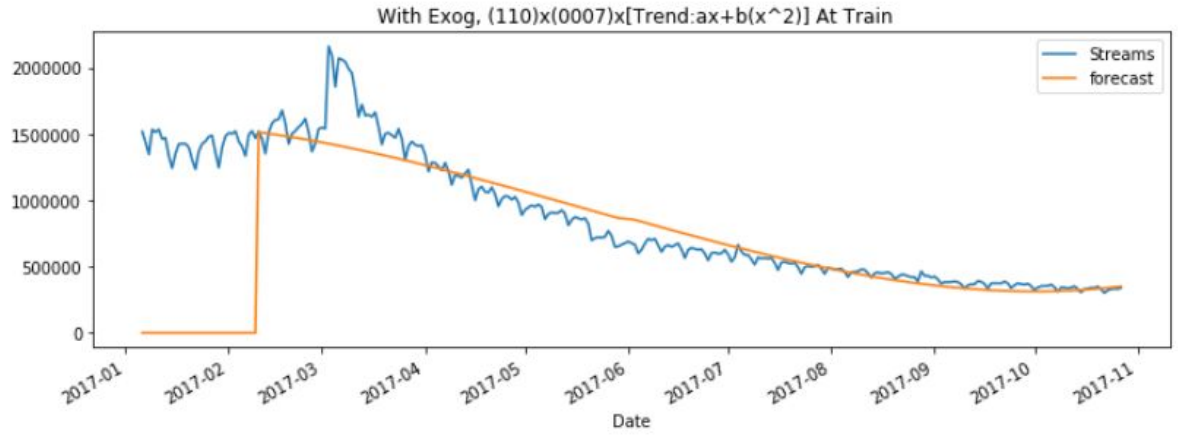
3.4 Prediction

To validate the models we tried changing the parameters in the ARIMA model to reflect the trend. The first experiment showed a bad model where we parameters as order(1,1,0) and seasonal(1,1,0,7). Further, tweaking the parameters we made few changes: order(1,1,0) and seasonal(0,1,1,7). This improved the model pretty decent.
To improvise the model we tested Exogenous SARIMAX with x variables. Here, we first tried tweaking the parameters again using order(1,1,0) and seasonal(0,1,1,7) we got





The attached x variable helped a lot in improving the model. Additionally, we tried to make changes in the parameter to test if we could reach to the optimal model, we used order(0,1,1) and seasonal(0,0,0,7). This gave the below result:



The exogenous SARIMAX proved to be a good model. In the previous prediction the power peak up, ARIMA completely ignored the peak up the all equations.

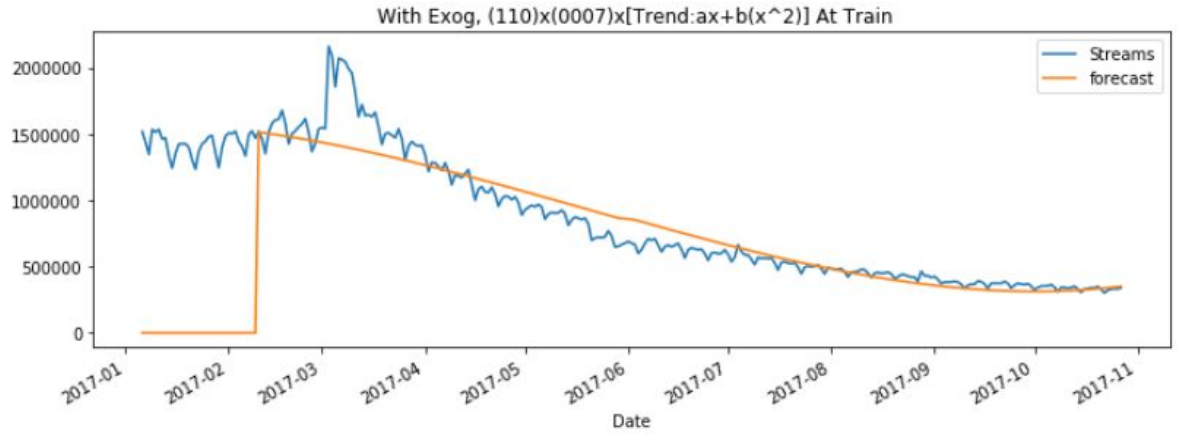
We implemented the 5 different models using the SpotifyFeatures dataset and tried to predict the popularity of the song based on features such as key, danceability, and acousticness. Year, artist, era, and genre were excluded from the dataset.
We compared the models and achieved a maximum accuracy of 92% with Random Forest.

4. Results

A total of 7 different models were built on our dataset using the classification techniques. The accuracy of the models was as follows:

	Model	Accuracy
1	RandomForestClassifier	0.921054
3	DecisionTreeClassifier	0.873610
2	KNeighborsClassifier	0.776475
0	LogisticRegression	0.749795
4	LinearSVC	0.695000

Using a dataset of Spotify Tracks, we were able to predict popularity using audio-based metrics such as key, mode, and danceability without external metrics such as artist name, genre, and release date. The Random Forest Classifier was the best performing algorithm with 92.0% accuracy. The Decision Tree Classifier was the second best performing algorithm with 87.5% accuracy. Linear Support Vector Classification performed worst with least accuracy rate of 69.5%.



5. Future Scope and Conclusion

5.1 Future Scope

We plan to implement LSTM and to check if it gives better results than the algorithms we used while carrying out this project. LSTM model can also be improvised to get better results like modifying the batch size,window size, hidden layer, clip margin and epochs.Also, LSTM can

further iterate through every window in batch and for every point in window we can feed LSTM to get the next output. End result will be considered as the prediction.

5.2 Conclusion

Using a dataset of Spotify Tracks, we were able to predict popularity using audio-based metrics such as key, mode, and danceability without external metrics such as artist name, genre, and release date. The Random Forest Classifier was the best performing algorithm with 92.0% accuracy. The Decision Tree Classifier was the second best performing algorithm with 87.5% accuracy. Linear Support Vector Classification performed worst with least accuracy rate of 69.5%

6. References

1. List of Countries https://gist.github.com/frankkienl/a594807bf0dcd23fdb1b
2. Positive Word List: http://positivewordsresearch.com/list-positive-emotion-words/
3. ARIMA Model Python : https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/
4. ARIMA Model Parameter Fitting: https://people.duke.edu/~rnau/411arim.htm
5. ARIMA Seasonal Parameter: http://www.seanabu.com/2016/03/22/time-series-seasonal-ARIMA-model-in-python/
6. https://www.kaggle.com/kerneler/starter-ultimate-spotify-tracks-db-1d4998a2-b
7. https://www.kaggle.com/huanntran100/spotify-song-popularity-prediction
8. https://www.kaggle.com/caicell/spotify-chart-trend-seasonal-arima#1.-Basic-Auditing
9. https://www.kaggle.com/zaheenhamidani/ultimate-spotify-tracks-db/downloads/SpotifyFeatures.csv/notebook
10. H. He and E. A Garcia. “Learning from imbalanced data. Knowledge and Data Engineering”, IEEE Transactions on, 21(9):1263–1284, 2009.

Group Members:
Pornima Bansode - NUID 001837517,
Shreya Jain - NUID 001832012,
Harshal Girme - NUID 001278531,
Ojas Phansekar - NUID 001826636,
Dharit Shah - NUID 001278838