Topify - Top List Prediction

Mehmet Ali Korkmaz [1] **Cem Doğan** [1] **Gamze Deniz** [1]

Abstract

Every year a Top List is published by Spotify. This Top List consists of the most popular 50-100 songs of that year. Being able to predict whether or not a song would be popular is challenging. Differently from previously used algorithms by others[4], we used Siamese Neural Networks and our dataset includes 5255 top songs and 20558 normal songs with 9 features. And we will be using SMOTE to data oversampling.

1. Introduction

People's one of the most common hobby is music. Deciding what people are looking for and liking in a song is one of the biggest concerns of music producers. Music producers have tried many solutions for this purpose [1]. The fact that this is a challenging problem got us interested so we are trying to find a new solution to this problem using Deep Learning. Focusing on features of popular songs could be a solution to this problem. And the features that we will use:

- **Acousticness:** The probability a recording was made by exclusively acoustic implies.
- **Tempo:** Tempo is a pace of a track which is indicated by beats per minute (BPM).
- **Valence:** Valence is a metric of positivity on a track. If valence value is high, recordings will sound more optimistic, whereas low-valence recordings sound more negative.
- **Liveness:** Liveness check if the track has audience noise and determines that whether the track recorded live or not.
- **Instrumentalness:** Criteria for evaluating whether the weight of the song depends on the lyrics or on the rhythm and flow. If it is close to the zero, it means that track is based upon lyrics like a rap song.
- **Loudness:** Loudness is track's total volume of a track and dB values are averaged throughout the whole track.
- **Danceability:** Dance ability of a track is for dancing based on several factors such as pace, rhythm, beat, and regularity. It indicates level of dance ability of track.
- **Energy:** Energy is a value indicating the level of activity of the song. Energetic tracks have features such as quickness, loudness, and boisterous vibe. We can think of metal music as energetic and classical music as not energetic.
- **Speechiness:** The participation of spoken words to a track. Speechiness is a value of a track that indicates the track is based on just speech-like or based on music.

Using these features we are comparing the songs up on the possibility of a popularity in a track. Among the methods we use to achieve our goal is the Synthetic Minority Over-sampling Technique, which we have tried to address imbalanced dataset problems in MLP models. We predicted that this would be inadequate and applied a different model and examined the impact of the Siamese Neural Network on imbalanced datasets.

2. Related Work

2.1. Song Popularity

Some people have thought about "What makes a song a hit?". One of them is Marty Dodson, a songwriter. He explains his thoughts on this subject[2]. And Collin Morris explains this problem with scientific viewpoint with including features of songs[3]. You can also find Ashrith Shetty's view of point on this subject[4].

2.2. ML Perspective

If you are wondering is there anyone who has researching this problem as a machine learning topic, yes, there is. Especially those who solve with the classical methods of Submission and Formatting Instructions for ICML 2021 machine learning can be found here[5], and here[6] also here[7].

2.3. Deep Learning Side

Also, there are more researchers about this subject who work on this more deeply (Deep Learning). Such as Abdul Meral, he built a neural network to find a solution to this problem[8]. And Stanford University students tried the similar thing with us[9]

2.4. Image Processing

It also crossed our minds to turn the songs into an image processing project by putting them into a visual diagram.However, we looked up a different way to find a solution to this issue. We were thinking of implementing Few-Shot Learning[10] which we believe in this method will be the solution. You can look into Harshall Lamba's research to learn what OneShot Learning is[11] and Toronto University's paper about Image Recognition with One-Shot Learning[12].

2.5. SMOTE

When working with unbalanced datasets, the difficulty is that most machine learning approaches will overlook the minority class, resulting in poor performance, despite the fact that performance on the minority class is often the most significant. Syntethic Minority Over-sampling Technique is a way to deal with this problem. You can find more on that with Notre Dame University publication[13].

2.6. Siamese Networks

A Siamese Network is based on two networks that receive different inputs but are linked at the top by an energy function. It can be used with image processing [14]. And it can be implemented with Keras [15].

3. The Approach

3.1. Dataset

The data we will use is taken from Spotify using spotipy module[16]. It consists of 5255 top songs and 20558 normal songs and 9 features such as acousticness, danceability, energy, instrumentalness, loudness, speechiness, audio valence, tempo, liveness. The dataset is visualized in Figure 1.

In our dataset we have 4:1 ratio between two classes, class distribution is not uniform Majority of set includes negative class and minority of the data set includes positive class therefore, we have an imbalanced dataset. We decided to turn our direction towards to Siamese Neural Network, and SMOTE, which is used in situations with imbalanced datasets. We've also turned our song comparison into a radar chart to make it easier to understand how to do this.

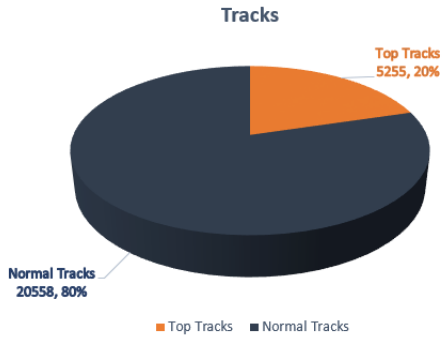

Figure 1. Percentage of Tracks

You can see in Figure 2 the song comparison.

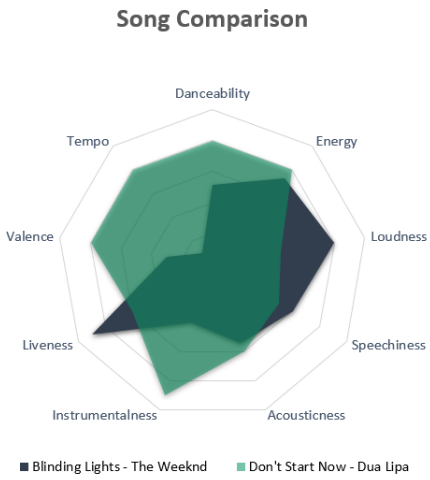

Figure 2. Comparing Top Tracks

Predicting whether a song would be in top list or not will require us to look at that song's features, then compare them with a top list song's features. When we compare these two songs' features we are looking for similarities between them. If there is enough similarities then we can say that the song would be in top list.

3.2. Multilayer Perceptron

3.2.1. Model 1: Sigmoid

First, we used a basic Multilayer Perceptron model. There are 6 layers in them, of which our input layer has 9 nodes, so we have reserved one node for each feature. In addition, we have 3 hidden layer and these consist of 128 nodes which uses ReLu activation function. And we have a Dropout layer that randomly sets input units to 0 with a frequency rate of 0.5 at each step during training time. Finally, our output layer contains 1 output node because our model is binary classification. In this model, we used **Sigmoid** function as

our activation function and **Binary Cross-Entropy** as loss function.

3.2.2. Model 2: Softmax

As with our first model, we started with a simple Multilayer Perceptron model. There are six layers in all, with nine nodes in our input layer, therefore one node has been set aside for each feature. We also have three hidden layers, each of which has 128 nodes and utilizes the ReLu activation function. We also have a Dropout layer that, at each step during training, sets input units to 0 at random with a frequency rate of 0.5. Finally, our output layer contains 2 output node. In this model, we used **Softmax** function as our activation function and **Sparse Categorical Cross-Entropy** as loss function.

3.3. Synthetic Minority Oversampling Technique

We know that our dataset is imbalanced. When working with imbalanced datasets, the problem is that most machine learning algorithms ignore the minority class, resulting in poor performance, despite the fact that the minority class is typically the most important.One method for addressing this issue is to oversample the instances in the minority class. Before developing a model, this may be performed by simply reproducing minority class examples in the training dataset. **Synthetic Minority Oversampling Technique** is a method of synthesizing new examples. After we used this technique we acquired new data similar to our base data. You can see the new dataset in Figure 3.

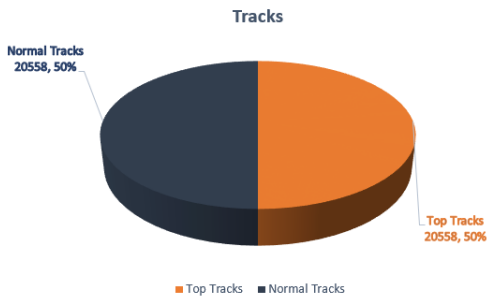

Figure 3. Oversampled Dataset

3.4. Siamese Neural Network

We took the Multilayer Perceptron model that we used previously and integrated it into the **Siamese Neural Network**. Siamese Neural Network takes (9,1) vectors as inputs then creates pairs from data which has the same labels, then we pass these pairs through the Multilayer Perceptron models we previously took, which shares weights, and calculated the **Euclidian Distance** between their outputs. If the distance is above 0.5, we accept them as similar and if it is below it, we accept them different.

4. Experimental Results

Dataset Description: The dataset contains 5255 top songs and 20558 normal songs with initially 16 features. First, we shuffle the data. Then, since there are features that we won't be considering while training our model such as Artist name, Song name, Key, Mode, Duration(ms), Track URI and Year we drop those 7 features. Our experiments were completed using the same test data. Since there is a known approach to solve this problem in Deep Learning using Tensorflow Keras Sequential Neural Network Model, firstly we experimented on this model. We also made several experiments to view SMOTE's effects. Finally, we tried Siamese Neural Network using Keras in our experiments.

4.1. Model 1: Sigmoid

4.1.1. without SMOTE

The imbalanced data caused our model to predict every sample as normal track and we got these results:

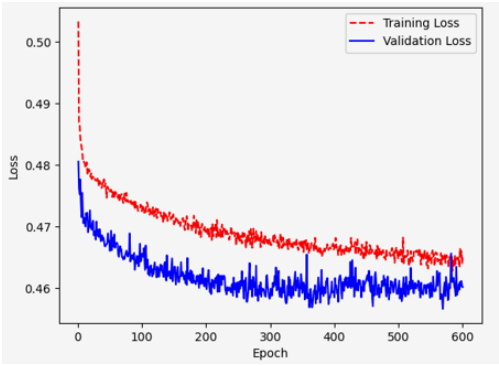

Figure 4. Loss-Epoch Graph

- Accuracy : 50%
- F1-Score : 0.08%

4.1.2. with SMOTE

After oversampling and balancing our data we got the Loss-Epoch graph Figure 5.

- Accuracy : 60.73%
- F1-Score : 56.65%

Comparing to before, accuracy and F1 score became higher.

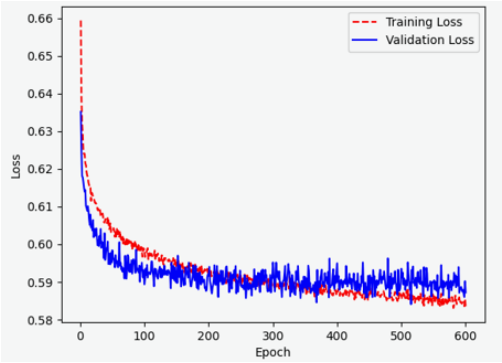

Figure 5. Loss-Epoch Graph

4.2. Model 2: Softmax

4.2.1. WITHOUT SMOTE

When we used Softmax instead of Sigmoid our Loss-Epoch graph becomes:

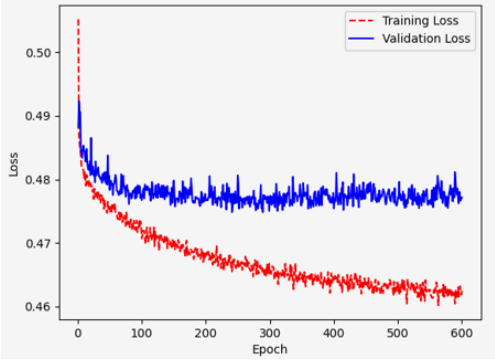

Figure 6. Loss-Epoch Graph

On Figure 6, we can see that our model is overfitting. And the results:

- Accuracy : 50%
- F1-Score : 0.03%

Our accuracy hasn't changed but our F1 score has changed.

4.2.2. WITH SMOTE

Using Softmax on balanced data, 55% of the songs got labeled as Top Tracks. Our Loss-Epoch graph is visualized in Figure 7.

With the results we got:

- Accuracy : 60.42%
- F1-Score : 66.35%

We can see that our accuracy and F1 score became higher.

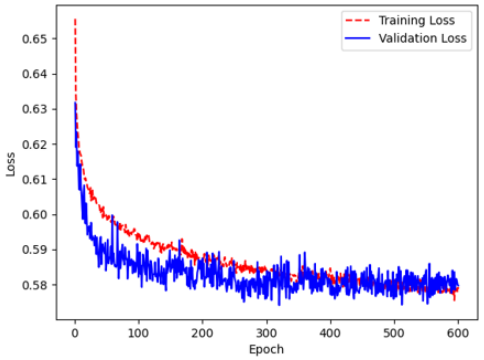

Figure 7. Loss-Epoch Graph

4.3. Final Model: Siamese Neural Network

4.3.1. WITHOUT SMOTE

As a result of our research, we have learned that the Siamese Neural Network can be effective in imbalanced datasets. So we initially tested our model in data without SMOTE. The content of the MLP model in the Siamese neural network is as follows;

- Input Layer with 9 nodes.
- Three Hidden Layers with 128 nodes which use ReLu function.
- Dropout Layer with 0.5 frequency rate.
- Epoch: 200
- Batch size is 64
- Loss Function: Binary Crossentropy

After we train our MLP models with these parameters, we measure the Euclidean distance between the outputs. And we classify it according to the similarity. And the test results of different hyperparameters will be compared to this graph;

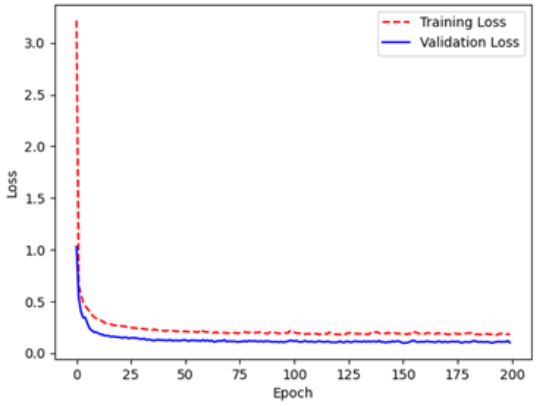

Figure 8. Loss-Epoch Graph

As you can see on Figure 8, we got shapely results. Our training loss and validation loss close to each other and decreasing constantly. When we tried to predict the outcome of our test data with this trained model we got these results:

- Accuracy : 95.71%
- F1-Score : 95.41%

We wanted to try different hyperparameters with this model. We changed batch size to 256 and got this graph;

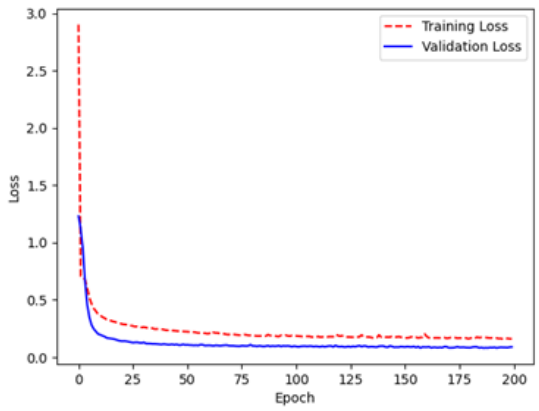

Figure 9. Loss-Epoch Graph

If we look at the Figure 9, we can deduce that batch size did not affect much. But we got slightly better accuracy. And we should note that test data is not changed in any prediction. Results are;

- Accuracy : 96.14%
- F1-Score : 95.94%

We changed epoch to 600 and got this graph;

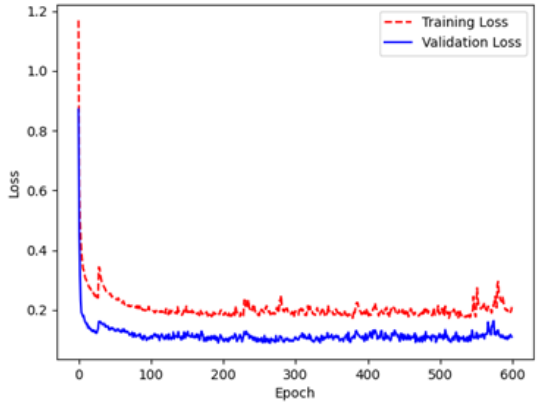

Figure 10. Loss-Epoch Graph

At the Figure 10, we saw the graph becomes slightly noisy but accuracy did not change. And also this model is slightly underfitting respect to the Figure 8. You can see that accuracy and f-1 score does not differ;

- Accuracy : 95.31%
- F1-Score : 94.86%

We wanted the see if increasing complexity of our model will have affect in the good direction. And we changed hidden layers' node size to 256 and got this graph;

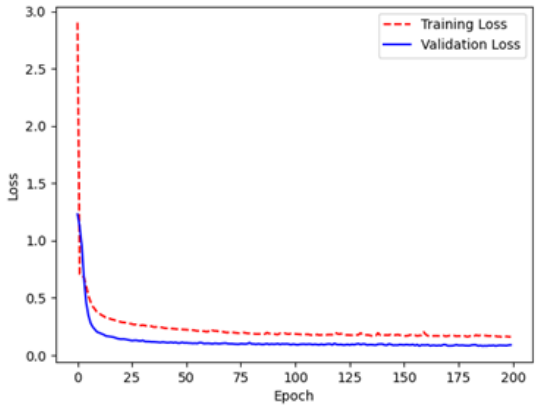

Figure 11. Loss-Epoch Graph

Making our model more complex, causes convergence between validation and training losses requires more epoch. But, accuracy and F1 scores are very close between Figure 11 and Figure 8. You can see the results;

- Accuracy : 95.22%
- F1-Score : 94.85%

Let's look at the different complex changer: Hidden Layer, we turn our layer size (3) to 5 to see if it will make difference.

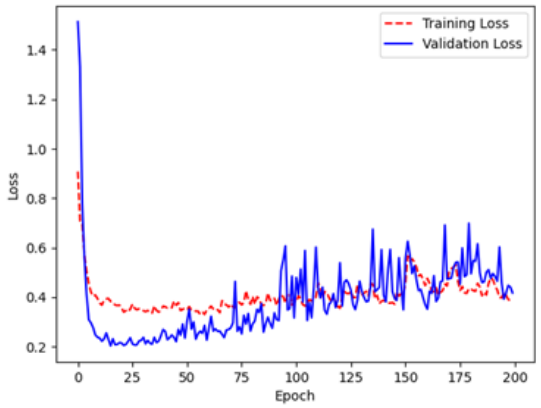

Figure 12. Loss-Epoch Graph

We found out complexity of our model became too high and it caused our charts to become very noisy. Also, it increased our validation and training losses instead of decreasing them. Plus, our accuracy and F1 score experienced a negative change. Prediction of our model started to get out of stability as you can see on the results;

- Accuracy : 84.64%
- F1-Score : 81.16%

4.3.2. WITH SMOTE

Using the same hyperparameters as Figure 8, we tried SMOTE on our Siamese model this time. And it resulted as this;

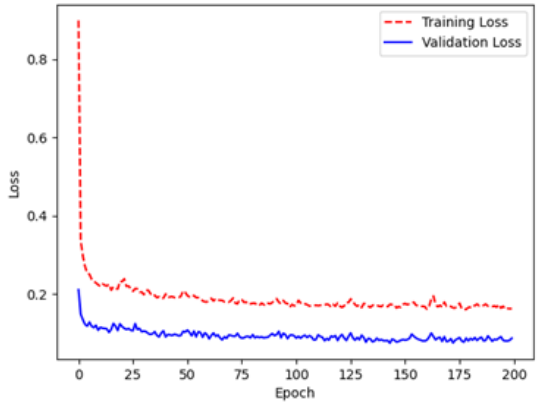

Figure 13. Loss-Epoch Graph

As a result, our loss values decreased more than usual. However, SMOTE does not work with our Siamese model efficiently. And it caused underfitting with our model. At the end, we got;

- Accuracy : 93.02%
- F1-Score : 92.29%

5. Conclusion

We wanted to develop a machine to predict if a song will be a Top Hit or not. For this purpose, we first tried an MLP model. Since our data was unbalanced, our model was overfitting, and our accuracy was low. We used SMOTE to overcome this problem, but we were not satisfied with 60% accuracy. Therefore, we tried Siamese Networks, which is commonly used for imbalanced datasets. We got our highest accuracy and F1 score using Siamese networks with batch size 256. We also used SMOTE with Siamese Networks. As a result, our model was underfitting, and also the accuracy decreased. It became apparent that our Siamese model should not be used with SMOTE. Using our Siamese model will surely be enough to tell apart a hit song from the other.

6. References

1. Marty Dodson 2020, *What REALLY Makes A Song Hit*
2. *What Makes A Song A Good Song? Music To Your Home.* June 2, 2021,
3. Collin Morris 2018, *What Makes A Hit*
4. Ashrith 2018, *What Makes A Song Likeable*
5. *Spotify Popularity Prediction-ML Practice* 2019
6. George McIncitre 2017, *A Machine Learning Deep Dive into My Spotify Data*
7. Minna Reiman, Philippa Örnell 2018, *Predicting Hit Songs with Machine Learning*
8. *Spotify - Deep Learning in 10 Steps* 2019
9. Stanford University, Pham, J. P., Kyauk, E. K., & Park, E. P. (2015). *Predicting Song Popularity.*
10. İzgi Arda Özsubaşı 2020, *Few-Shot Learning (FSL): What it is and its Applications*
11. Harshall Lamba 2019, *One Shot Learning with Siamese Networks using Keras*
12. Gregory Kzoch, Richard Zemel, Ruslan Salakhutdinov 2016, *Siamese Neural Networks for One-shot Image Recognition*

13. Brownlee, J. (2021, March 16). *SMOTE for Imbalanced Classification with Python. Machine Learning Mastery.*

14. Singh, P. (2019, August 31). *Siamese Network Keras for Image and Text similarity.*

15. *Keras implements Siamese Network - Programmer Sought.* June 2, 2021.

16. *Spotipy*