Spotify Song Hit Predictor

CP468 - Artificial Intelligence

SUBMITTED BY
Stefan Bukarica - 190563930
Duc Minh Nguyen - 203009140

Wilfrid Laurier University

Under the Instruction of:

Dr. Sukhjit Singh Sehra

Abstract

In this paper we discuss and explore how to predict whether a song will be a hit or not. A hit is defined as a song trending on the Billboard Hot 100 Chart. The Billboard Hot 100 is a record chart of the most popular songs of any genre within the United States. The popularity is defined by a few metrics such as radio play, use in media, and internet streaming service popularity (number of plays). For the purpose of the project the data that was gathered was through the Spotify API, which displays an adjusted popularity of the songs based only on the Spotify streaming service. However, due to the popularity of streaming services, it is noted that Spotify's popularity is almost identical to the popularity represented on Billboard. In the project, we first import the data and perform some preprocessing to make sure our dataset is cleaned and ready for training. Then, we explore the various attributes of the dataset, and determine their influence in whether a song is a hit or not. We then train a few machine learning models via the sk-learn package in Python, to determine which models perform best. Finally, we take a look at the higher performing models and attempt to predict whether random and manually generated songs can be a hit. The importance of the music industry cannot be understated, especially with the boom in music streaming services over the past 10 years, it is important for artists and labels to see what trends can influence creating a hit song. This could be useful for label executives to enhance their monetary gain with well performing songs.

Acknowledgement

We would like to acknowledge the producer of the dataset on Kaggle, Farooq Ansari

Furthermore, the acknowledgements Ansari has provided passed down;

"spotipy": Python module for Spotify's API (https://pypi.org/project/spotipy/)

"billboard": Python module for Billboard's API (https://pypi.org/project/billboard.py/)

Spotify, the company itself. For keeping a database of such in-depth details of every track in their library. And for exposing their API for the world to use.

List of Figures:

List of Tables:

List of Abbreviations:

Abbreviation	**Full Form**
SVM	Support Vector Machine
ML	Machine Learning

Table of Contents

Chapter 1: Introduction

Songs have become essential to many people's lives. This project used the existing data set to train and test and give the prediction of the next popular song. Which song could become a smash hit in the future based on the dataset. The goal of this project is to use a machine learning model and predict the popularity/ hit chances of a song. To get an understanding of the dataset, below are the attributes of this dataset:

Attribute	Description
track	The Name of the track.
artist	The Name of the Artist.
uri	The resource identifier for the track.
danceability	Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.
energy	Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy.
key	The estimated overall key of the track. Integers map to pitches using standard Pitch Class notation. E.g. 0 = C, 1 = C?/D?, 2 = D, and so on. If no key was detected, the value is -1.
loudness	The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typically range between -60 and 0 db.
mode	Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.

speechiness	Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.
acousticness	A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
instrumentalness	Predicts whether a track contains no vocals. "Ooh" and "aah" sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly "vocal". The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.
liveliness	Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live.
valence	A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).
tempo	The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.
duration_ms	The duration of the track in milliseconds.
time_signature	An estimated overall time signature of a track. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure).
chorus_hit	This is the author's best estimate of when the chorus would start for the track. It's the timestamp of the start of the third section of the track. This feature was extracted from the data received by the API call for Audio Analysis of that particular track.
sections	The number of sections the particular track has. This feature was extracted from the data received by the API call for Audio Analysis of that particular track.

target	The target variable for the track. It can be either '0' or '1'. '1' implies that this song has featured in the weekly list (Issued by Billboards) of Hot-100 tracks in that decade at least once and is therefore a 'hit'. '0' Implies that the track is a 'flop'. The author's condition of a track being 'flop' is as follows: - The track must not appear in the 'hit' list of that decade. - The track's artist must not appear in the 'hit' list of that decade. - The track must belong to a genre that could be considered non-mainstream and / or avant-garde. - The track's genre must not have a song in the 'hit' list. - The track must have 'US' as one of its markets.
Decade	The decade that the song was made in, values include: 1960, 1970, 1980, 1990, 2000, 2010.

Table 1. List and Description of Attributes

These attributes will be looked at in the next chapter as we explore their presence in the dataset. Furthermore, the datasets provided are multiple csv files, each with the corresponding songs organized by decade 1960-2010. This will help us identify trends through the years, as shown later in this paper. Our goal is to be able to identify what attributes contribute to making a song a hit. Then we will train a few machine learning models to identify which ones perform best. Finally, we will test the prediction of the model corresponding to the most influential attributes making up a “hit” song. This problem is a classification problem since the variable target is what we are trying to predict and it has a binary value of 1 or 0.

Chapter 2: Exploratory Data Analysis

Dataset:

For this project we have combined all 6 of the datasets of decades, into one dataframe. We then add a column named “decade” to bind each song to the correct decade it was released in. We then shuffled this complete data frame before preprocessing to ensure it is in a random order. For the preprocessing, we remove the categorical attributes which are not necessary for our exploration and machine learning training. These attributes are ‘track’, ‘artist’, and ‘uri’. These only label the individual songs, each of them acting as an ID, they do not tell us anything about the quality or popularity of a given song. After splitting the data set into a 80/20 train-test split, we made sure to normalize/standardize the values using sk-learn’s StandardScaler. This produces a mean of 0 and variance of 1 across all attributes. We do this in order to remove bias. This is because variables that are measured at different scales do not contribute equally to fitting a model and can end up creating a bias.

Attribute Analysis:

We began by plotting the ranges of hit vs flops for the following attributes: tempo, danceability, duration, loudness, speechiness, valence, energy, key, and acousticness. Below are these 9 graphs shown, with blue being the hits, and orange being the flops. This will help us determine the ranges we should test for when we want to predict whether a song is a hit or not.

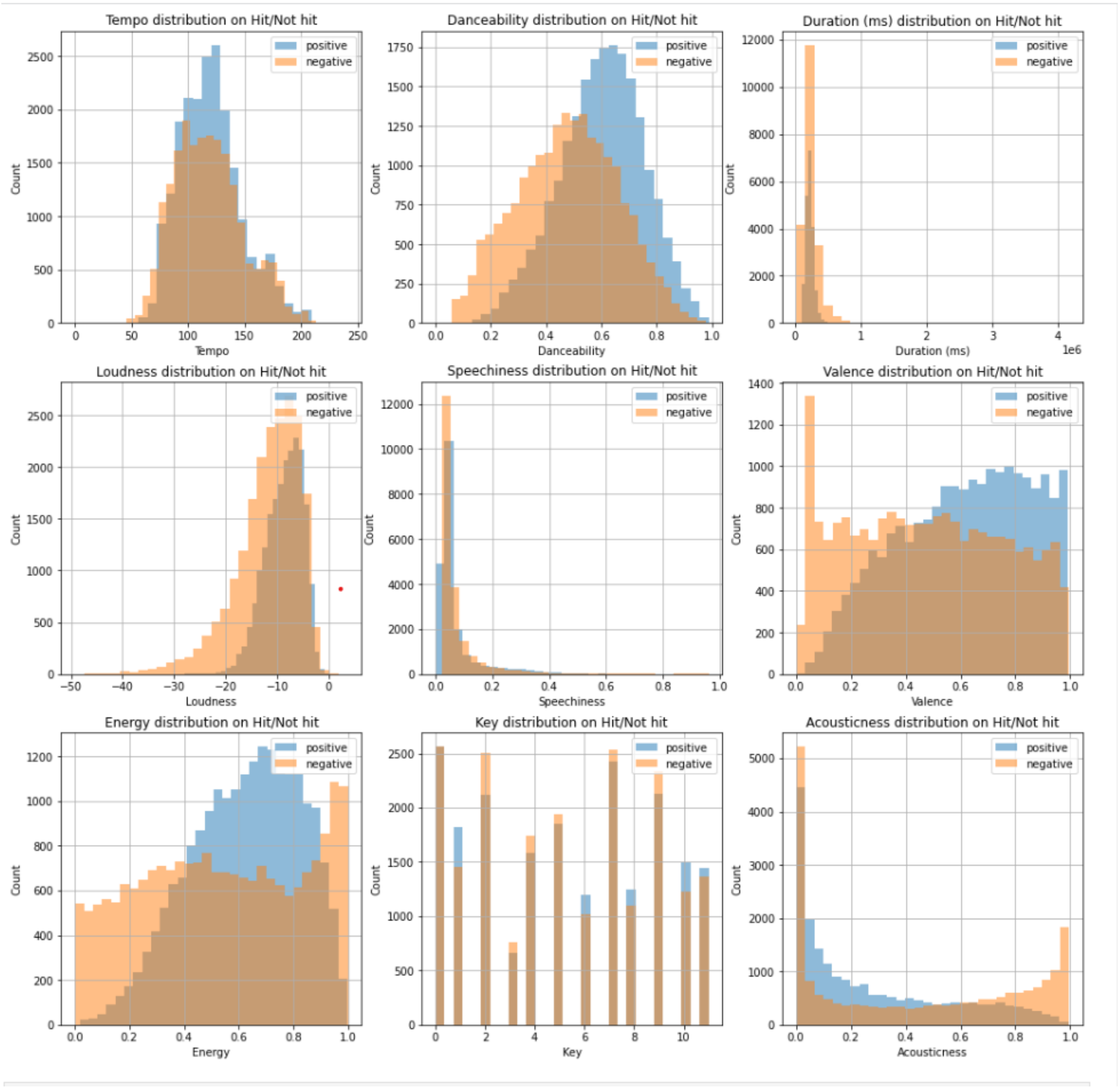


Fig 1. Graphs of Attributes grouped by Hit and Flop

Below this is a table of the indicated hit ranges for each of the above attributes of the dataset. It is important to note that values outside of the estimated hit range are considered flops for when we start to make predictions. The following table identifies what ranges of attributes will likely produce a hit song:

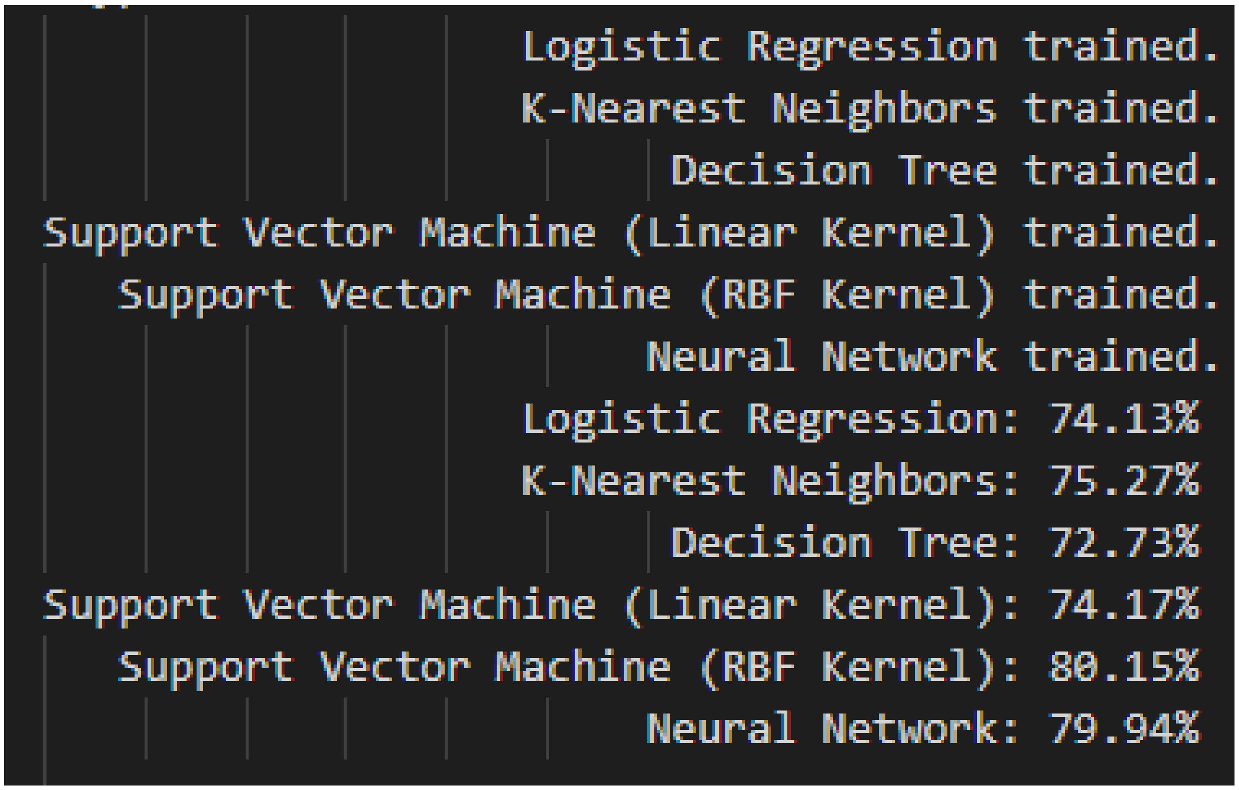
Attribute	“Hit” Range
Tempo	80–160
Danceability	0.5-1.0
Loudness	-20 - 0
Speechiness	0.0-0.2
Valence	0.5-1.0
Energy	0.4-0.9
Key	2,4,8,10
Acousticness	0.0-0.6

Table 2. “Hit” Ranges of Attributes

Note that these are the attributes for which we noticed an apparent difference in hits vs flops, with these being a clearly identified hit range based on the graphs.

Chapter 3: Model Selection

Now that our data has been preprocessed and split into training and test sets, it's time to start training different models. We start by training a few popular machine learning models; Logistic Regression, K-Nearest Neighbors, Decision Tree, SVM (Linear Kernel), SVM (RBF Kernel), and finally a Neural Network. Each of these models was imported from sk-learn. Once each of the models is trained we calculate the accuracy of each using the '.score' method from sk-learn. The results are as follow:



```
                Logistic Regression trained.
                K-Nearest Neighbors trained.
                      Decision Tree trained.
Support Vector Machine (Linear Kernel) trained.
   Support Vector Machine (RBF Kernel) trained.
                     Neural Network trained.
                Logistic Regression: 74.13%
                K-Nearest Neighbors: 75.27%
                      Decision Tree: 72.73%
Support Vector Machine (Linear Kernel): 74.17%
   Support Vector Machine (RBF Kernel): 80.15%
                     Neural Network: 79.94%
```

Fig 2. Machine Learning Models

Now that we can see the accuracy we will continue with the SVM RBF Kernel algorithm and a Neural Network algorithm, since these two models produced the highest accuracy.

Chapter 4: Prediction

Now that we have our models, we will create two sets of predictions y1_pred, and y2_pred, each corresponding to the SVM and Neural Network model, respectively. We then create a confusion matrix to display the contrast of actual vs predicted results. As we can see in the top left and bottom right squares display which predictions are the same as in the test set. The figure below is a representation of our SVM model predictions.

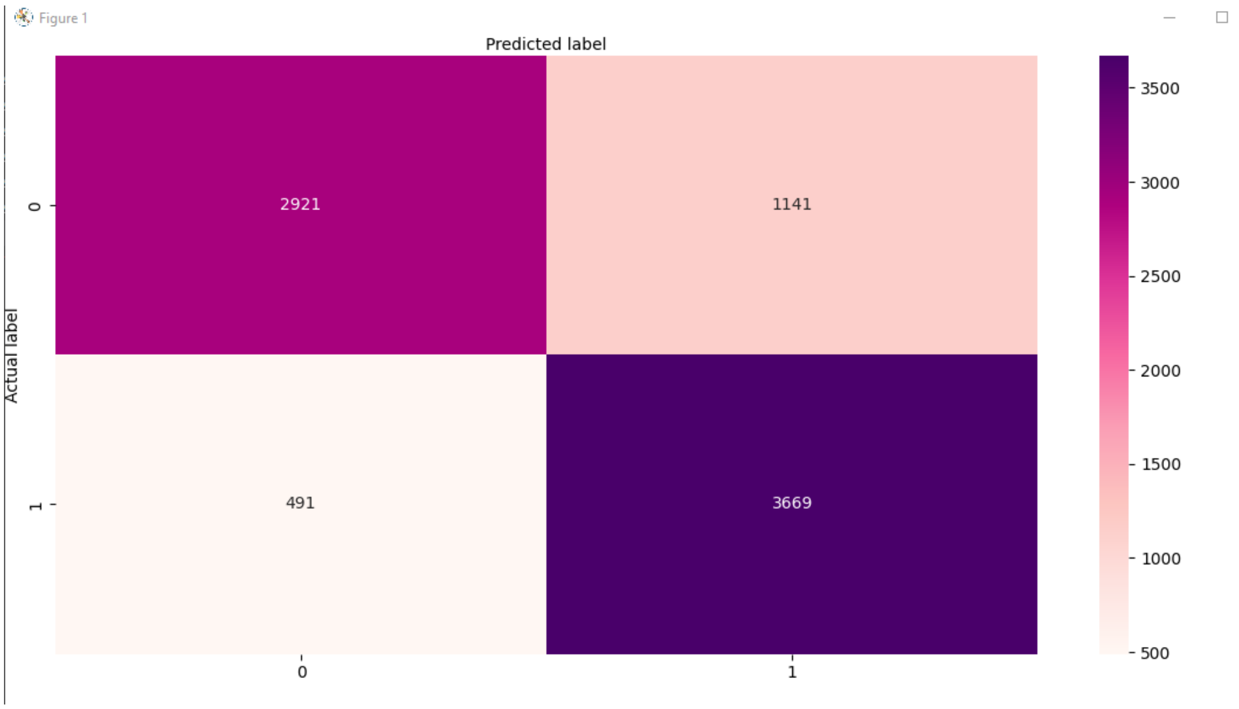


Fig 3. SVM Predictions in Contrast with Test Set.

The next diagram is the same concept of a confusion matrix however, it is with the Neural Network model, as we can see it performs relatively well with its predictions matching the test set.

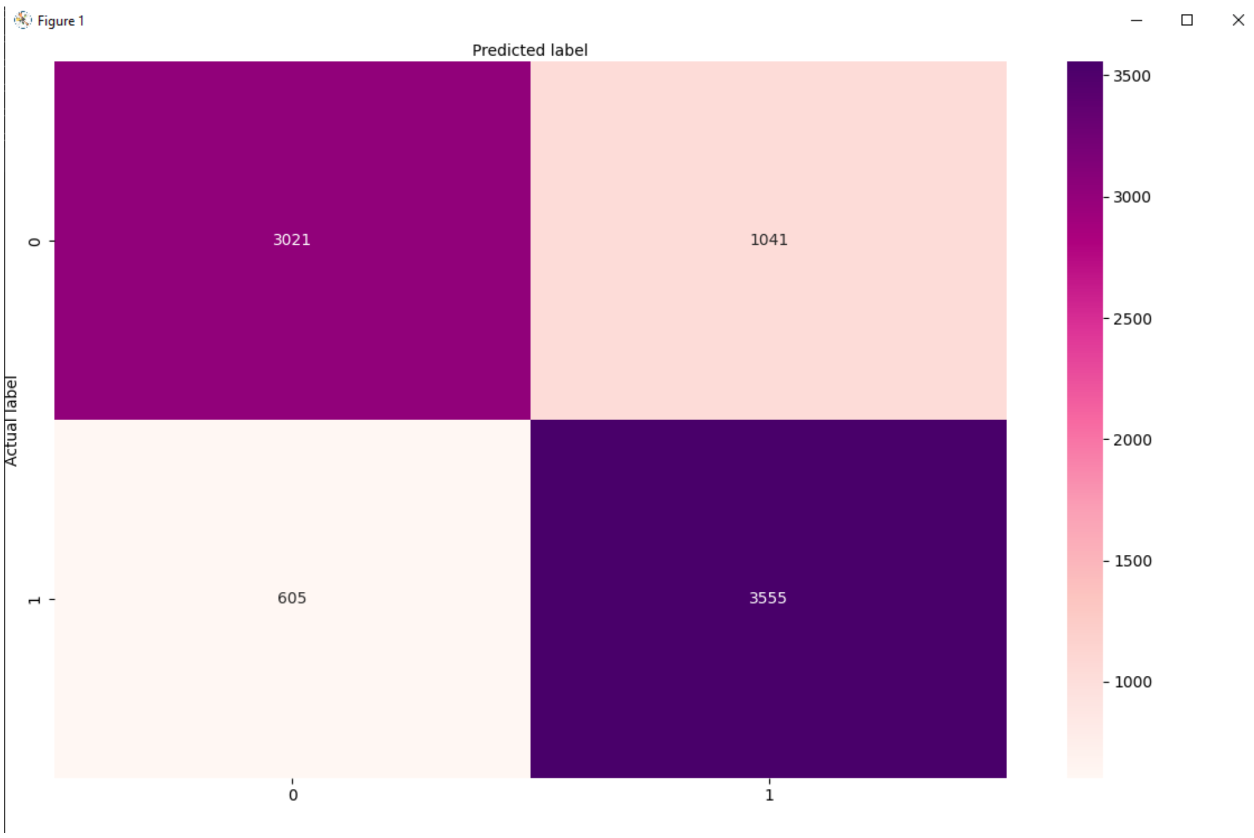
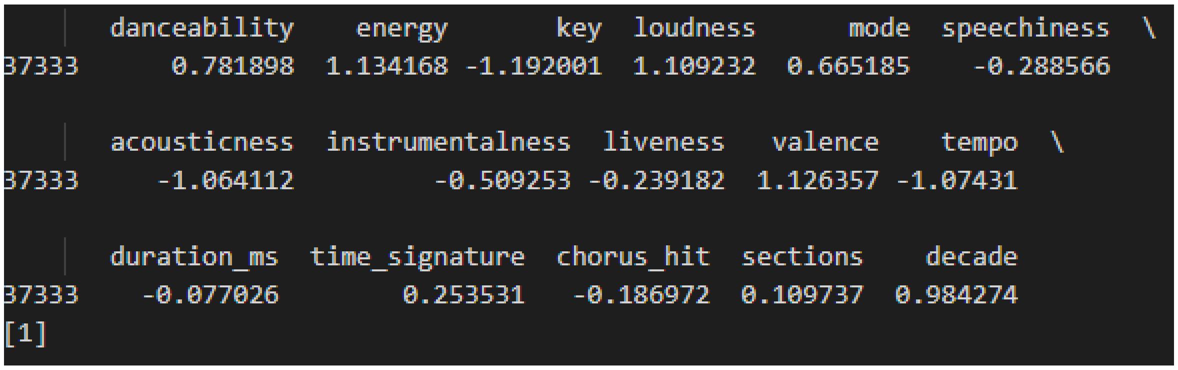


Fig 4. Neural Network Predictions in Contrast with Test Set.

After this, we created multiple random samples to see if the hit songs follow the trends we analyzed for hit songs earlier on.



```
      danceability    energy       key  loudness      mode  speechiness  \
37333     0.781898  1.134168 -1.192001  1.109232  0.665185    -0.288566

      acousticness  instrumentalness  liveness   valence    tempo  \
37333    -1.064112         -0.509253 -0.239182  1.126357 -1.07431

      duration_ms  time_signature  chorus_hit  sections    decade
37333   -0.077026        0.253531   -0.186972  0.109737  0.984274
[1]
```

Fig. 5 Prediction of a random song in Dataset

In this case, the values of this random song are standardized, analyzing this we can see that it has a low duration, high danceability, high energy, and high loudness. More importantly we can see that this is one of the more recent songs released since decade is close to 1. This makes sense as many of the hits in today's music have similar traits as the ones listed above. As we can see, the model predicted 1 indicating it was a hit as assumed. We repeat these investigations with the Neural Network Model in our code to confirm our assumptions.

Chapter 5: Conclusion

Through our analysis we have managed to identify some obvious trends of hit songs since 1960. We then cleaned our dataset and ensured it was ready to be used to train and test our ML models. Finally, we evaluate the performance of multiple models in order to be able to select two of them to run predictions on. We used a SVM with an RBF Kernel, and a Neural Network model, respectively to conduct our predictions. Both of them had a similar effectiveness and were able to predict roughly 80% of songs as a hit or flop correctly.

Chapter 6: Future Scope

For the future, a better dataset would be ideal. The first thing to collect would be more records of songs, as there are many more songs available in the Spotify API. An increase in records would likely increase the accuracy of the ML models since the training dataset would be much larger. Furthermore, we could change the ideas of what constitutes a "hit" or not. With changing times in the music industry, social media and streaming services have taken over traditional indicators of popularity such as radio play. For example, a song that is trending on TikTok is incredibly popular considering the amount of people that would hear it, however that doesn't always translate to streaming and radio play metrics. Therefore, a new model of popularity could be incredibly important in producing an actual indicator of current popularity. Furthermore, we see spikes in popularity of older songs on streaming services after the song trends on social media apps like Instagram and TikTok. With this new, corrected, popularity attribute it could redefine what is truly popular and what is not, which would be incredible to explore.

Bibliography

- Spotify dataset from Kaggle