T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

SONG POPULARITY PREDICTION

ABDURRAHMAN BULUT

SUPERVISOR
DR. GÖKHAN KAYA

GEBZE
2022

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

SONG POPULARITY PREDICTION

ABDURRAHMAN BULUT

SUPERVISOR
DR. GÖKHAN KAYA

2022
GEBZE

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 15/01/2023 by the following jury.

JURY

Member
(Supervisor) : Dr. Gökhan Kaya

Member : Dr. Yakup Genç

ABSTRACT

In this research, I propose a machine learning-based approach for predicting music popularity on streaming platforms using data from Spotify. The approach utilizes various classification and regression methods such as decision tree, logistic regression, perceptron, decision tree, an ensemble method, and a neural network model using Keras. I collected a large dataset of nearly 586,672 songs with 15 features from the Kaggle website, which is prepared by utilizing Spotify API. The experimental results show that the proposed approach achieved high accuracy in predicting music popularity. Additionally, I found that the combination of multiple models and the use of a neural network improves the prediction performance. However, it should be noted that there may be other factors that affect music popularity that were not considered in this research. Future studies will aim to include more features and to further understand the relationship between music popularity and other factors.

Keywords: Machine Learning, Spotify Audio Features. Music, Data Science

SUMMARY

This research proposes a machine learning-based approach for predicting music popularity on streaming platforms using data from Spotify. The approach utilizes various classification and regression methods such as Decision Tree, Logistic Regression, Perceptron, an ensemble method, and a neural network model using Keras. The data set collected from Kaggle, contains nearly 586,672 songs with 15 features. The experimental results show that the proposed approach achieved high accuracy in predicting music popularity and the combination of multiple models and the use of a neural network improves the prediction performance. However, the results also indicate that there may be other factors that affect music popularity that were not considered in this research. The research suggests that future studies should aim to include more features and to further understand the relationship between music popularity and other factors.

Keywords: Song popularity prediction, Machine learning, Audio features, Data analysis

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to my advisor, Dr. Gökhan, for his guidance, support, and encouragement throughout the course of this research project. His expertise and experience in the field of machine learning have been invaluable in the development of my research. He provided me with invaluable feedback and suggestions which helped me to improve my work. I am grateful for the time he spent discussing my ideas and for the many helpful suggestions he provided. I couldn't have done this project without his help.

I am also grateful to Kaggle community who shared data set that I used in my project. Additionally, I would like to acknowledge the related works and literature in the field of music popularity prediction and machine learning, which provided valuable insights and a foundation for this research. Lastly, I would like to acknowledge the hard work and dedication of all the researchers and practitioners in the field of machine learning and music popularity prediction. Their contributions have laid the foundation for this research and have been a constant source of inspiration.

Abdurrahman Bulut

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol or Abbreviation	:	**Explanation**
SVM	:	Support Vector Machine
EDA	:	Exploratory Data Analysis
MSE	:	Mean Squared Error
MAE	:	Mean Absolute Error
KNN	:	K-Nearest Neighbors
LGBMA	:	Light Gradient Boosting Machine
XGBoost	:	Extreme Gradient Boosting
ANN	:	Artificial Neural Network

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

In today's digital age, the music industry is heavily influenced by technology and innovation. Music has become a ubiquitous part of our culture, and with the advent of streaming services, it is now more accessible than ever. According to research, streaming music online has seen a 76.4% increase from 2015 to 2016, while traditional music sales have dropped significantly. This increase in streaming has led to a growing revenue for the music industry, with a predicted market volume of over $13 billion by 2023. Furthermore, since the COVID-19 outbreak, stocks related to the music industry have risen, such as global revenue from streaming, CD, and digital music sales have surpassed pre-COVID-19 levels. Big Hit Entertainment renamed its company HYBE. HYBE Entertainment not only increased its sales after Corona but also created a platform by collaborating with IT practitioners to provide convenience to fans. The company grew even more by taking over other agencies during the economic downturn.

However, income disparity is a serious problem in the music industry. While famous and popular artists and music companies earn enormous amounts of money, most artists and companies earn less than the minimum cost of living. Since 1982, the top 1% of artists have gone from earning 26 percent of revenue to between 56% and 77%.

In the past, music concerts and sale rates of music CDs are the most revenue. However, these days with the emergence of smartphone and music applications, music streaming, which is a way of delivering music without requiring users to download files from the internet. Music streaming revenue has increased dramatically in the last five years growing from $1.9 billion in 2014 to $ 10.1 billion in 2020. The income of an artist may not be an important issue for some artists, but more income from more artists will help the music industry with the high quality of music.

Spotify is one of the most famous music streaming services which supply numerous kinds of music in the world. Therefore, the artist can earn tons of money if the music ranks high on Spotify. To rank high on Spotify, it is important to understand listeners' tastes and interests. This paper is going to analyze the music of last 40 years in Spotify to investigate which factors of music influence the rank. The paper is going to use not only the popularity but the tempo and other variables about the song. Then the paper is going to implement the machine learning model that can predict the popularity or rank of music in Spotify. The model does not consider the name of the artist when it predicts the popularity of an unpopular artist can decide whether their music has the potential to be ranked in Spotify. The research expects that this model can be used by many artists or music companies to predict their rank before they release the music.

As the music industry's influence grows, predicting popularity will be utilized more widely than before. Furthermore, due to the development of music platforms such as Soundcloud, which makes it easier to approach composing, many people compose music these days, and i think this paper can help whether their songs can be successful.

1.1. Project Content

The project aims to predict the popularity of a song using machine learning techniques on a dataset containing audio features of the song. The dataset is obtained from a music streaming platform such as Spotify. The project includes the implementation of various machine learning algorithms to analyze the data and predict the popularity of a song.

To showcase the results, a website has been developed using VueJs, where the predicted popularity of a song can be viewed along with the song's audio features. Additionally, the website includes a feature that allows users to change the year of the song's release and see what the predicted popularity would be for that year. This feature is designed to provide insight into how a song's popularity would change in different years.

The project is composed of two main components: the prediction model and the website. The prediction model is developed using machine learning techniques and the website is developed using VueJs and hosted on Netlify. The website allows users to interact with the prediction model, view the audio features of the song and the predicted popularity, and also change the year of the song's release to see how the predicted popularity would change. The website also includes a simple user interface and visualizations to help users understand the results easily.

1.2. Project Requirements

- Determination of parameters affecting popularity: It is necessary to identify and determine the key audio features that affect the popularity of a song, such as energy, tempo, and danceability.
- Comprehensive dataset: Finding a comprehensive dataset containing audio features and popularity information of a large number of songs is crucial for the success of the project.
- Analysis of machine learning algorithms: The project will involve the analysis of various machine learning algorithms that have been used in previous studies on music popularity prediction. These include Linear regression, Decision tree, Random forest algorithm, and a neural network keras model.
- Data grouping: It will be necessary to group the data by different criteria such as year of release to account for possible differences in popularity trends over time.
- Required technologies and libraries: The project will make use of several technologies and libraries such as Numpy, Pandas, Scipy, Scikit-learn, TensorFlow, Matplotlib, and VueJs. These technologies and libraries will be used for data preprocessing, machine learning, and web development.
- Ensemble model: An ensemble model, which is a combination of several different models, will be used in the project to improve the accuracy of the prediction.

1.3. Project Design Plan

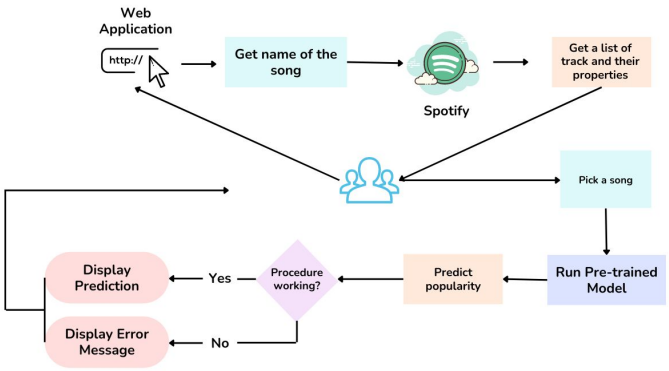

Figure 1.1: The design plan of the project.

- User Authentication: The website will include a user login system where users can create an account and login with their credentials.
- Search Function: Once logged in, users will be able to search for a song by entering the song name or artist name.
- Song Listing: The search function will return a list of songs matching the search criteria. Users will be able to select a song from the list.
- Data Retrieval: When a user selects a song, the song's audio features will be retrieved from the dataset.
- Model Prediction: The retrieved audio features will be sent to the trained machine learning model for prediction of the song's popularity.
- Result Display: The result of the prediction will be displayed on the website for the user. The result will include the predicted popularity score as well as the song's audio features.
- Year Change Feature: The user can also change the year of the song's release and see how the predicted popularity would change in that year.

- Data Visualization: The website will include data visualization to help users understand the results easily.
- Mobile Responsiveness: The website will be designed to be responsive and accessible on mobile devices.
- Error Handling: The website will include error handling to handle any errors that may occur during the search, data retrieval, and prediction process.

2. LITERATURE REVIEW

In the field of music popularity prediction, there have been several studies that have used machine learning techniques to analyze audio features and predict the popularity of songs. In this article, I have presented a methodology to predict the popularity of songs using machine learning algorithms and a dataset of song's audio features from the streaming platform Spotify. This methodology is similar to the works of "Hyeonsoo Oh" who proposed a combination of K-means and LGBM algorithm to predict the popularity of songs. The results of "Hyeonsoo Oh" revealed that this methodology could enhance the performance and be applied more widely to different datasets. [1]

Similarly, "Jigisha Kamal", "Pankhuri Priya", "Anala M R and Smitha G R" have used various machine learning models to predict the popularity of songs and they have found that the Random Forest Classifier gave the best accuracy of 85.06%. They have also found that tempo is the most critical feature that affects the popularity of songs. [2]

"Katherine Lin" and "Rudolf Newman" also used a decision tree model to determine the factors that affect the popularity of songs and they found that loudness and tempo are major factors in determining the popularity of songs. [3]

"Kai Middlebrook" and "Kian Sheik" focused mainly on the accuracy of results and they used a Neural Network model with one hidden layer, Logistic Regression model, Random Forest model and SVM model. They found that the Neural Network model gave the best accuracy of 82.14% and 83.05% on the validation and test data. They also found that Random Forest model gave the best accuracy of 88.7% on the validation data and 87.7% on the test data. All the models were found to have acceptable precision and recall. [4] Apart from these, I benefited from many literature sources.[4] [5] [6] [7] [8] [9] [10] [11]

3. IMPLEMENTATION

In this section, I will detail the steps taken to implement our prediction model for song popularity. I will first discuss the dataset used, followed by the pre-processing and feature selection techniques applied. I will then describe the machine learning algorithms and techniques that were used to train and evaluate the model. Finally, I will provide an overview of the website that was developed to showcase the results of our prediction model.

3.1. Data Collection

In the implementation of this project, data collection was a crucial step. The dataset used in this project was obtained from Kaggle, specifically the ”Spotify Dataset 1921-2020, 600k+ Tracks” which contains 586,672 tracks.[12]. This dataset was prepared by extracting data from the Spotify official API, providing information on various features of each track such as the id, name, popularity, duration, explicitness, artists, release date, and various audio features such as danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo, and time signature. This dataset was then used to train and evaluate machine learning models for predicting the popularity of songs.

3.2. Data Preprocessing

The first step in data preprocessing was to remove features that were not useful for our analysis such as song_id, song_name, artist_id, artist_name. This was done to reduce the dimensionality of the dataset and make it more manageable. Next, I used one hot encoding for the mode and key features. This allowed us to easily include these categorical variables in our machine learning models.

I also cleaned and transformed the release date feature by removing the day and month components and only keeping the year. This was done because earlier music data before 1980 was found to be unbalanced and would not provide meaningful insights. Additionally, I removed songs with a popularity of 0 from the dataset as they were not considered relevant for our analysis.

To standardize the data, I converted the duration_ms feature to durations_and divided by 100. This helped to reduce the scale of the feature and make it more comparable to other features. I also identified outliers using the Z-score method and removed them from the dataset as they could have a negative impact on the performance of our models.

Finally, before training the models, I applied PCA (Principal Component Analysis) on some datasets and performed feature selection steps on others. This helped to reduce the dimensionality of the dataset and improve the performance of the models.

3.3. EDA - Exploratory Data Analysis

In this section, I will be performing exploratory data analysis (EDA) on the dataset to gain a better understanding of the data and identify any potential issues that may need to be addressed before building our models. I will start by visualizing the distribution of the features using histograms.

The histograms below show the distribution of each feature in the dataset. As we can see, the popularity feature is heavily skewed towards lower values, with the majority of songs having a popularity score of less than 50. The duration_ms feature, liveness and speechiness also shows a similar distribution, with the majority of songs having a duration of less than 200,000 milliseconds. The danceability, energy, acousticness, instrumentalness, and valence features show similar distributions amongst themselves, with most values falling between 0 and 1. The time_signature feature, on the other hand, shows a more evenly distributed distribution, with most songs having a time signature of 4/4. These histograms give us a general idea of the distribution of each feature in the dataset and will be useful in identifying any potential issues that may need to be addressed before building our models.

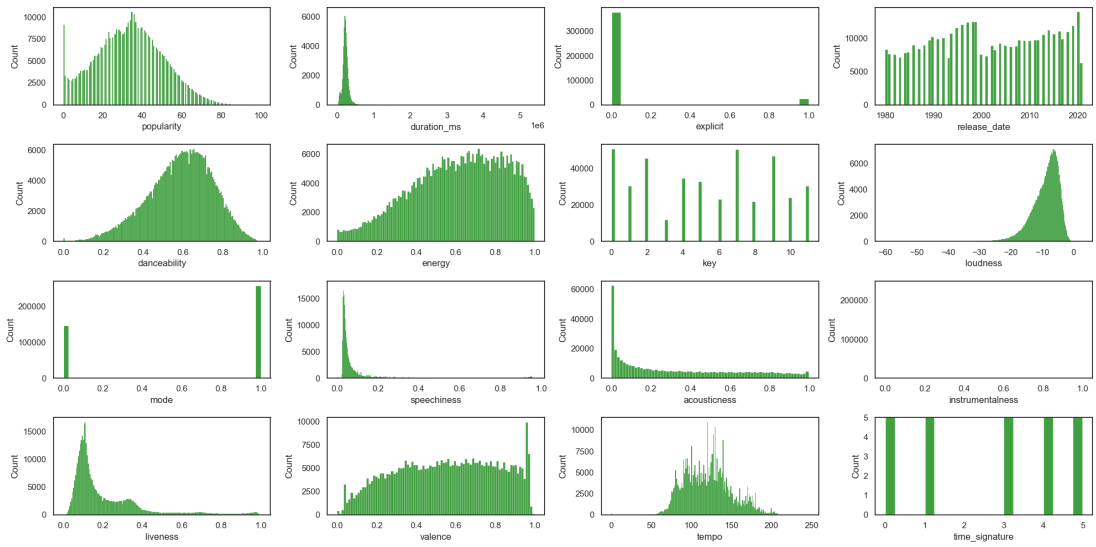

Figure 3.1: Histogram view for all features.

Box plots are another useful tool for identifying outliers and understanding the distribution of a feature. Box plots display the minimum, first quartile, median, third quartile, and maximum of a feature and any data points that fall outside of the minimum and maximum are considered outliers. By analyzing the box plots of our features, we were able to identify several outliers that will be removed from the dataset before moving on to the modeling phase.

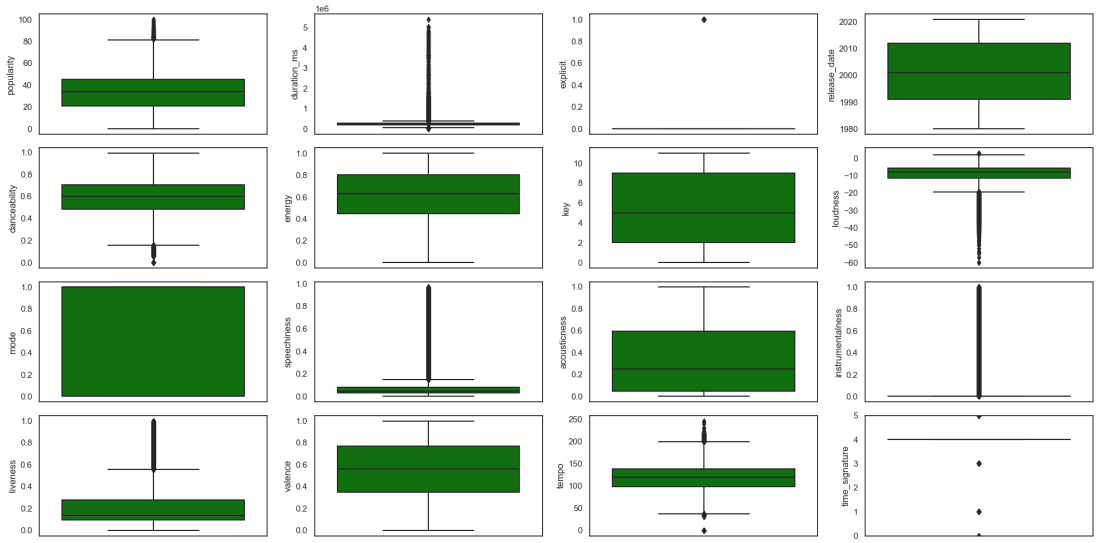

Figure 3.2: Box plots display.

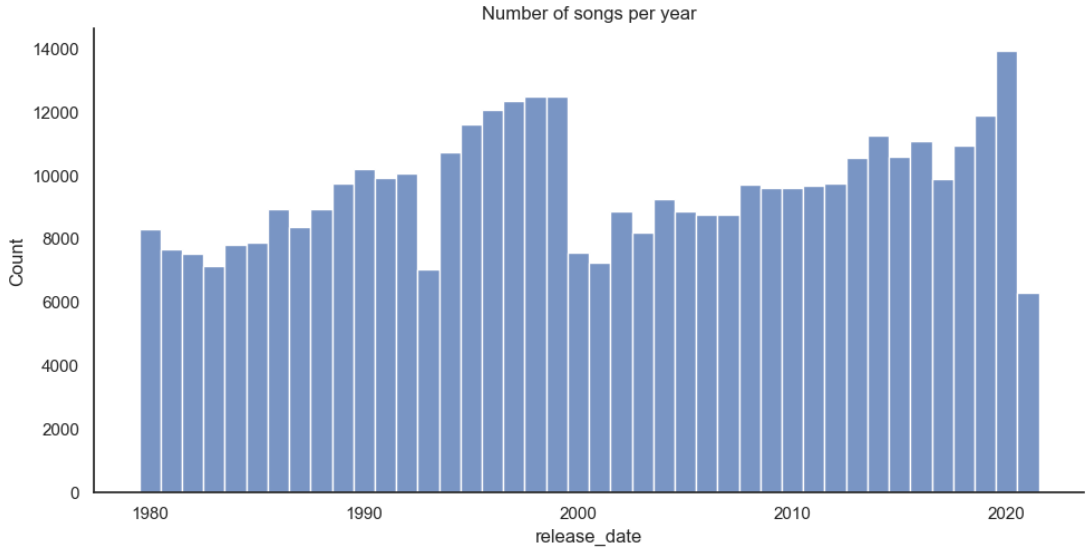

Figure 3.3: Number of songs per year.

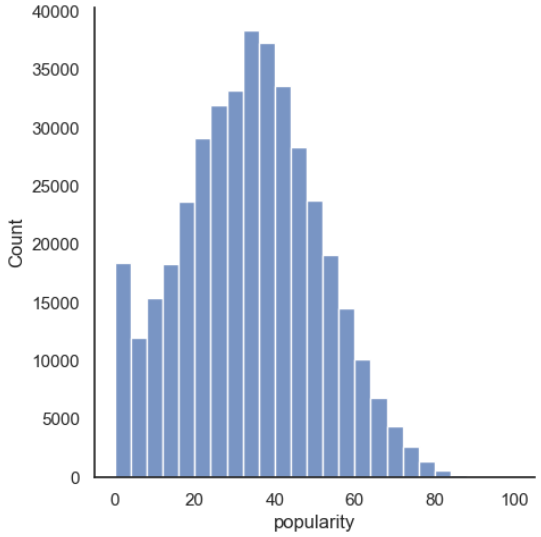

Figure 3.4: Number of songs per popularity.

In this section, we will explore the correlation between different features in our dataset. To do this, we will use a correlation matrix which will show the relationship between each pair of features in terms of their correlation coefficient. The correlation coefficient ranges from -1 to 1, where -1 indicates a strong negative correlation, 0 indicates no correlation, and 1 indicates a strong positive correlation. From the correlation matrix, we can see that the popularity of a song has a strong positive correlation with the danceability, loudness, and explicit of the song. This suggests that songs that are more danceable, loud, and explicit tend to be more popular. On the other hand, the popularity of a song has a strong negative correlation with the acousticness and instrumentalness of the song. This suggests that songs that are more acoustic and instrumental tend to be less popular. Additionally, there is a moderate positive correlation between the energy and loudness of a song, which is also expected, as energetic songs tend to be louder.

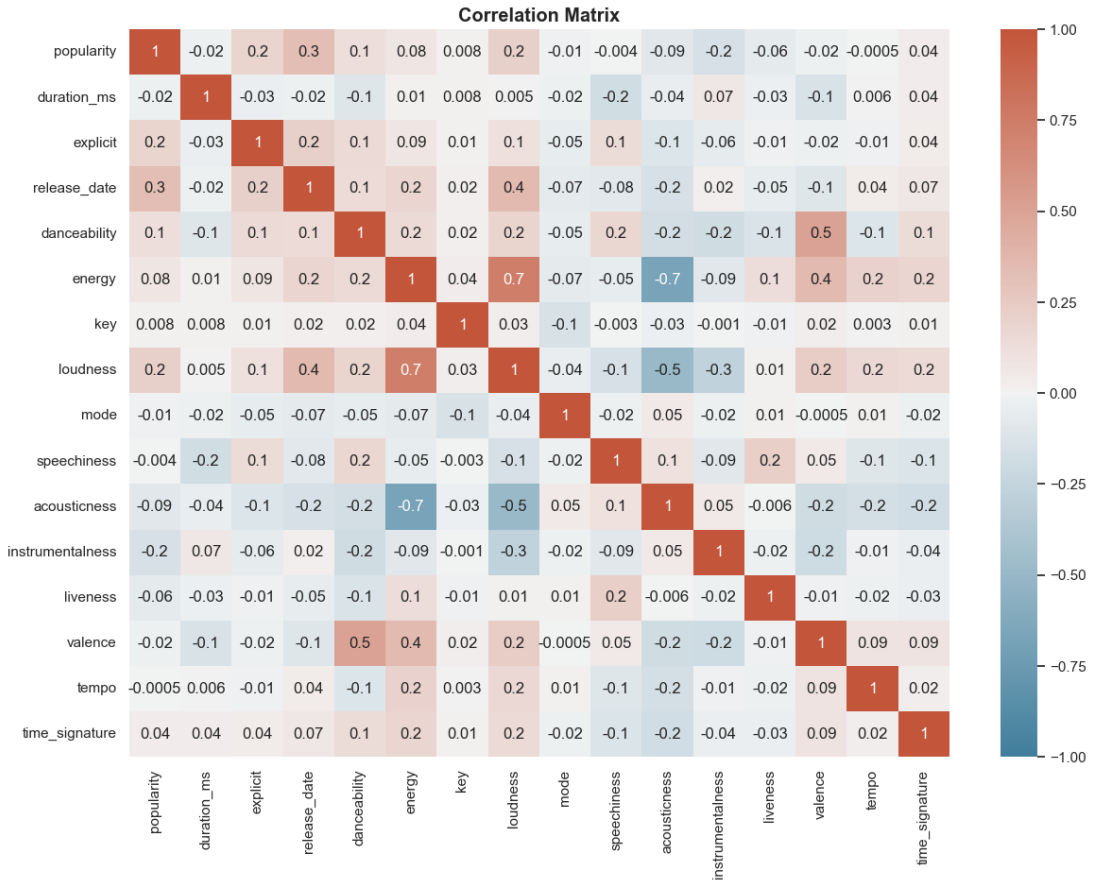

Figure 3.5: Correlation Matrix

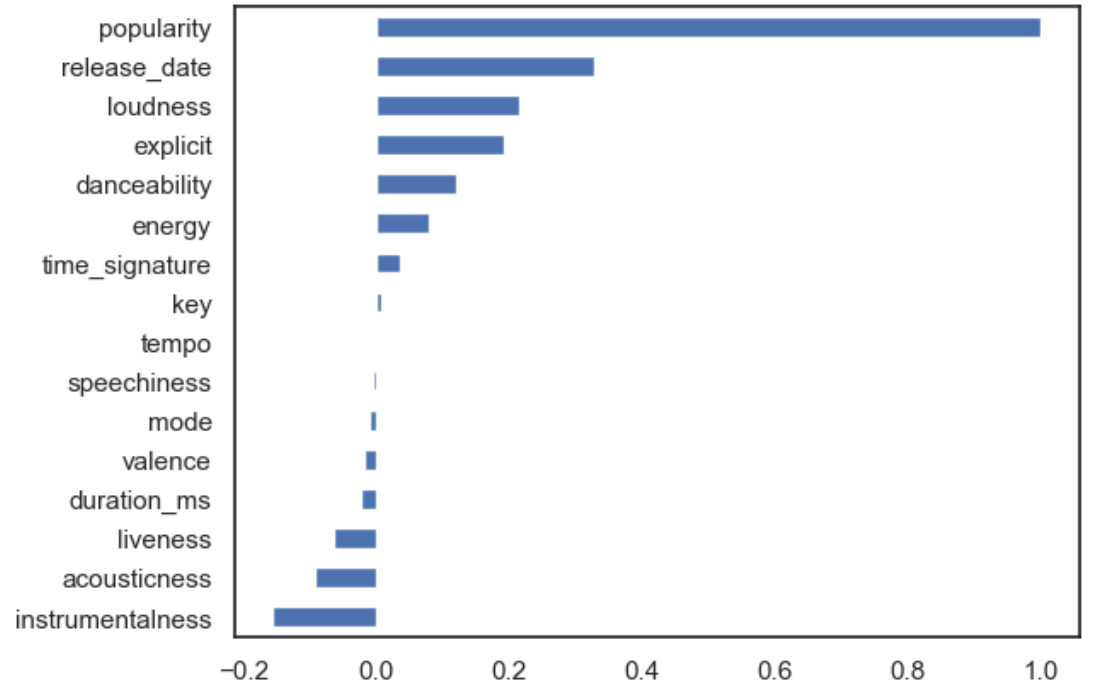

Figure 3.6: Correlation Matrix for Popularity

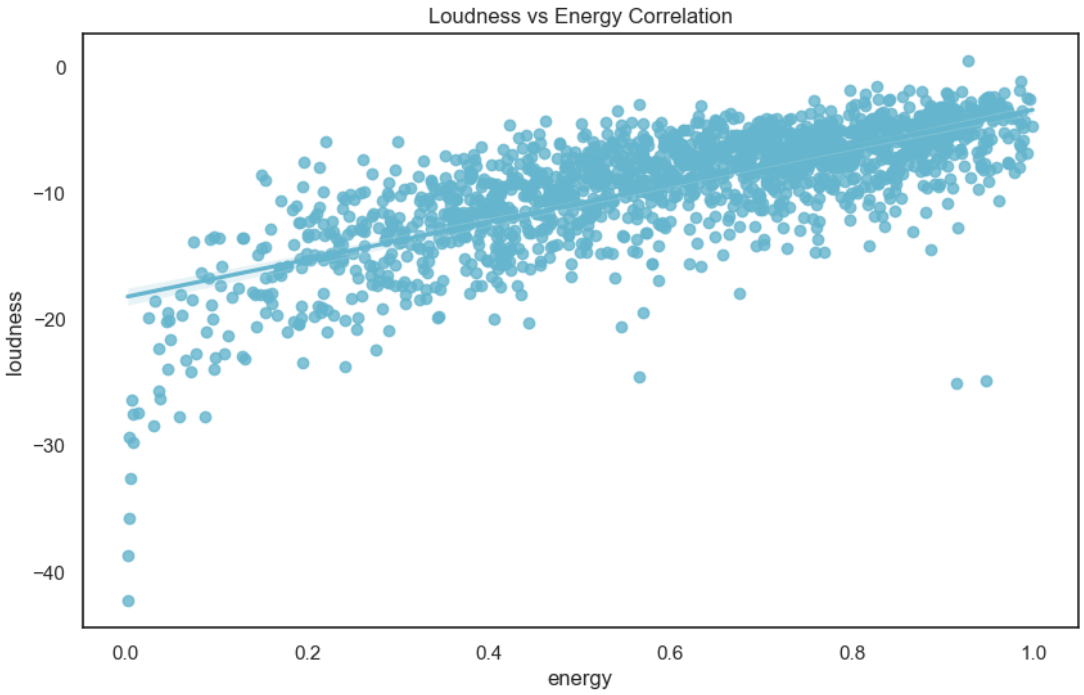

Figure 3.7: Correlation plot between Loudness and Energy

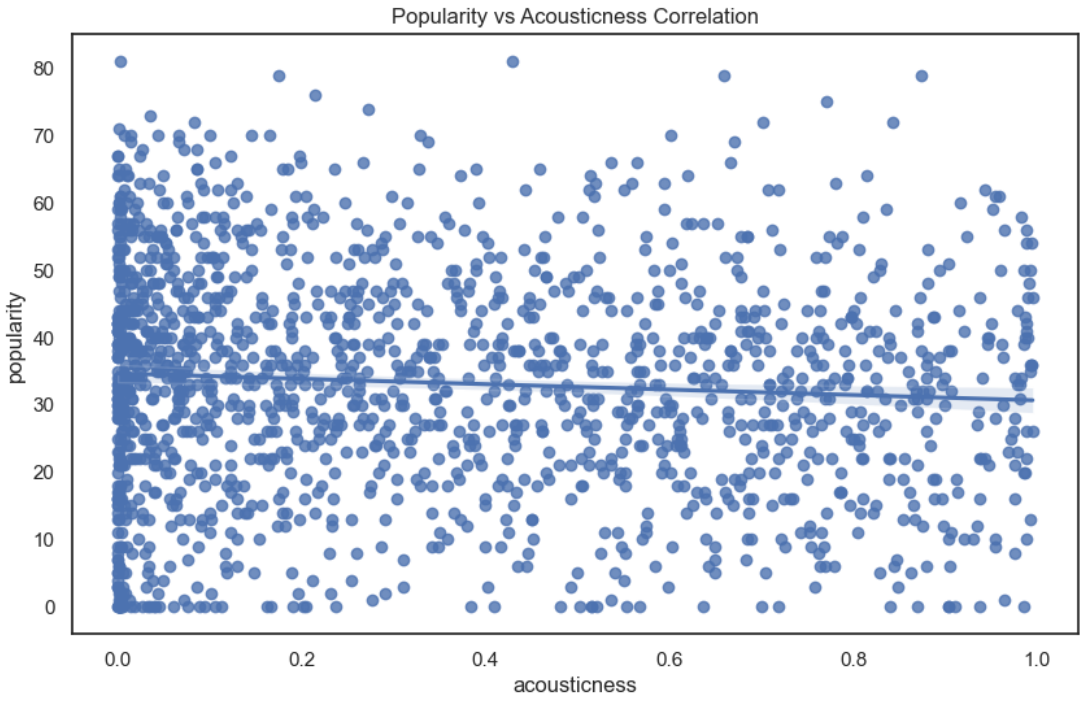

Figure 3.8: Correlation plot between Popularity an Accousticness

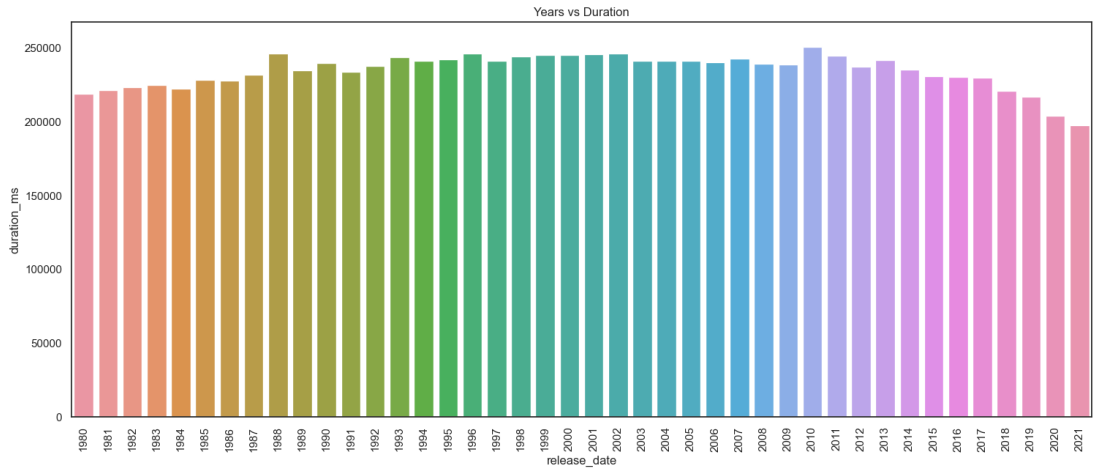

Figure 3.9: Relationships between release date and track's duration

3.4. Model Training and Evaluation

I will delve into the implementation of various machine learning models for the prediction of song popularity. The data preprocessing steps previously discussed will be applied to the dataset, and the cleaned data will be used for model training and evaluation. I will be utilizing several popular machine learning algorithms such as linear regression, decision tree, Random Forest, and a neural network model. The performance of each model will be evaluated based on metrics such as accuracy, AUC, recall, precision, and F1-score. I will also be using techniques such as PCA and feature selection to improve the performance of the models. The goal of this section is to determine which model can predict song popularity with the highest accuracy and identify the most important features that contribute to a song's popularity.

3.4.1. Classification Algoritmhs

I classified the songs with over 60% popularity as popular and below 60% as unpopular.

3.4.1.1. Decision Tree

In the decision tree part of the model training and evaluation, I used both the Entropy and Gini impurity as loss functions to split the data and create the decision tree. This was done in order to compare the performance of the two different functions and determine which one produces the best results. The decision tree was built using the scikit-learn library and various parameters, such as maximum depth and minimum samples per leaf, were adjusted to optimize the model's performance. The tree was then trained on the preprocessed dataset and evaluated using various metrics, such as accuracy. Accuraciy score is obtained as 82.69%.

```
Confusion Matrix
 [[52149 10342]
 [ 9405  4363]]


Classification report

              precision    recall  f1-score   support

           0       0.85      0.83      0.84     62491
           1       0.30      0.32      0.31     13768

    accuracy                           0.74     76259
   macro avg       0.57      0.58      0.57     76259
weighted avg       0.75      0.74      0.74     76259
```

Figure 3.10: Evaluation metrics for Decision Tree.

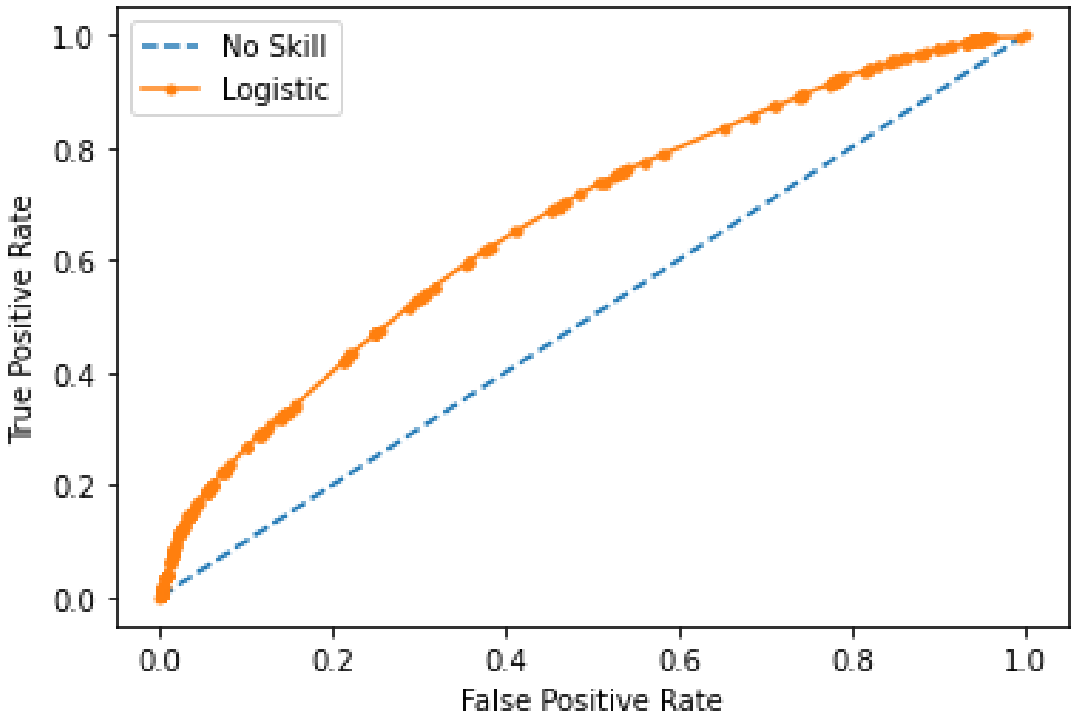

Figure 3.11: ROC Curves for Decision Tree.

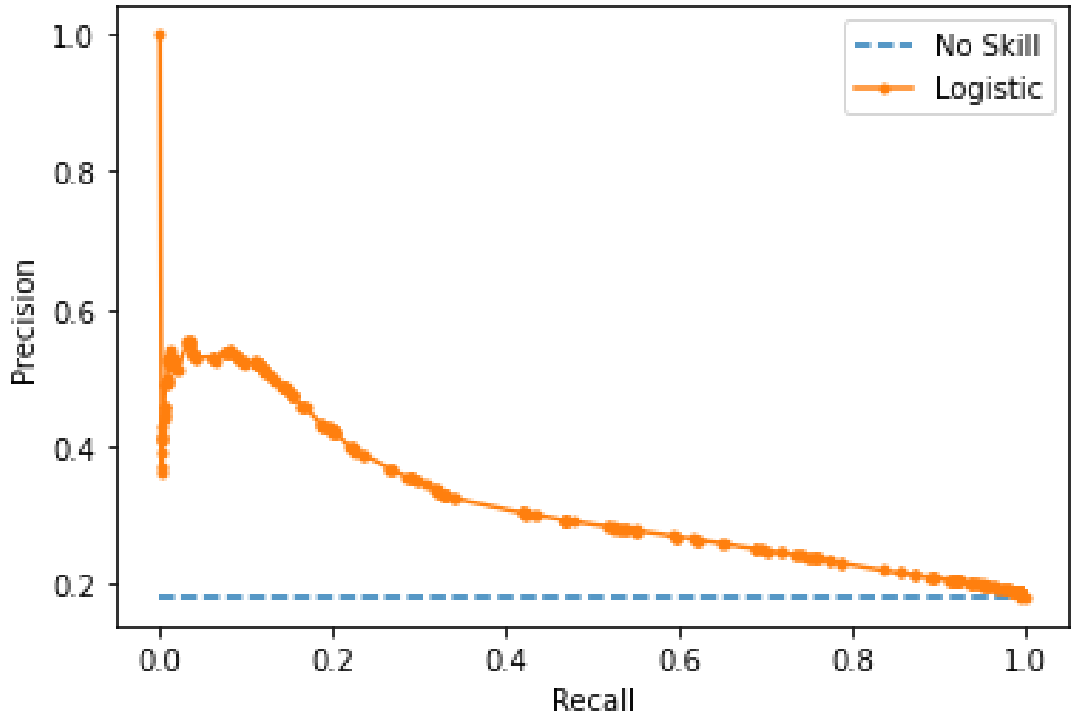

Figure 3.12: Recall vs Precison for Decision Tree.

3.4.1.2. Logistic Regression, KNN, Multi-layer Perceptron, LGBMA, XGBoost

In this section, we evaluated the performance of five different machine learning models for our song popularity prediction task. These models were Logistic Regression, K-Nearest Neighbors, Multi-layer Perceptron, Light GBM and XGBoost. We trained and tested these models on our dataset, and evaluated their performance using various evaluation metrics such as accuracy, F1-score and AUC-ROC. We also visualized the results of these models using ROC curves and F1-score plots. We compared the performance of these models and observed that the Light GBM model performed the best among the five models with the highest F1-score and AUC-ROC. The other models also performed well, but not as well as the Light GBM model. These results indicate that the Light GBM model is a suitable choice for our song popularity prediction task.

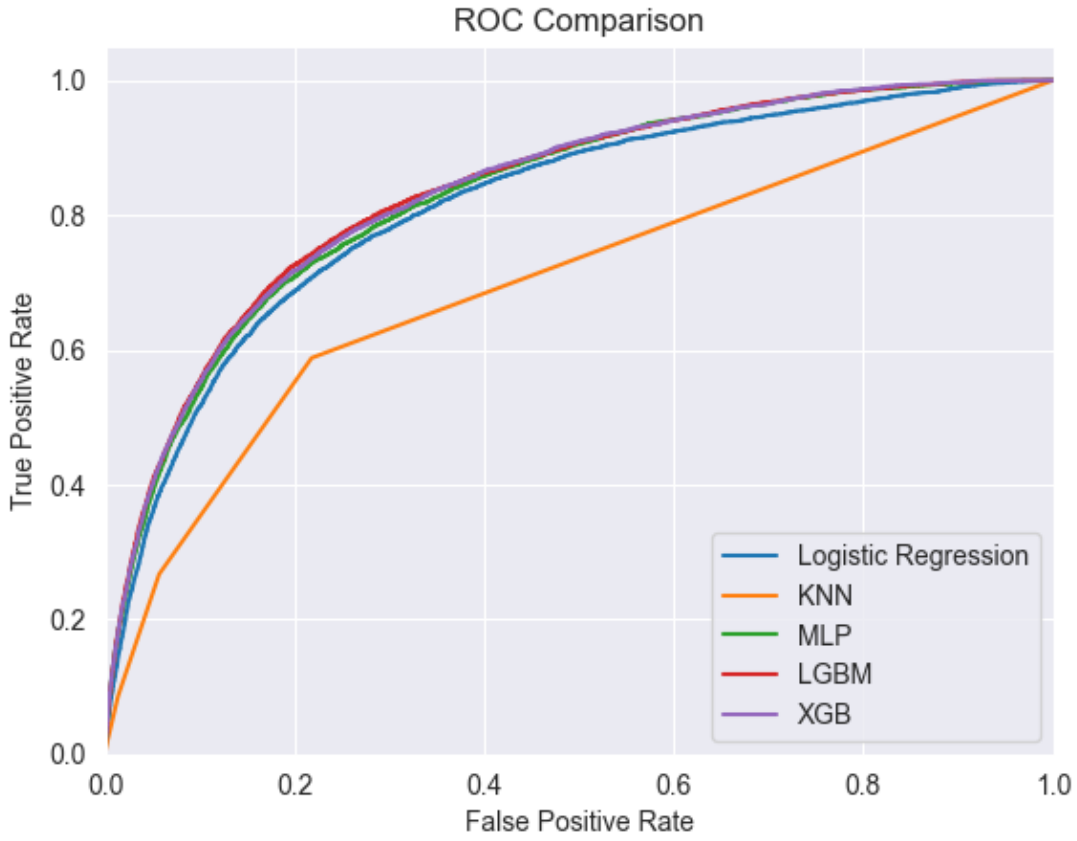

Figure 3.13: ROC Curves comparison.

```
Accuracy:  0.9332537798817189
Confusion Matrix:
 [[71123    30]
 [ 5060    46]]
Classification Report:
              precision    recall  f1-score   support

           0       0.93      1.00      0.97     71153
           1       0.61      0.01      0.02      5106

    accuracy                           0.93     76259
   macro avg       0.77      0.50      0.49     76259
weighted avg       0.91      0.93      0.90     76259
```

Figure 3.14: Evaluation metrics for Logistic Regression.

```
Accuracy:  0.9261464220616583
Confusion Matrix:
 [[70193   960]
 [ 4672   434]]
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.99      0.96     71153
           1       0.31      0.08      0.13      5106

    accuracy                           0.93     76259
   macro avg       0.62      0.54      0.55     76259
weighted avg       0.90      0.93      0.91     76259
```

Figure 3.15: Evaluation metrics for KNN.

```
Accuracy:  0.9341717043234241
Confusion Matrix:
 [[70898   255]
 [ 4765   341]]
Classification Report:
              precision    recall  f1-score   support

           0       0.94      1.00      0.97     71153
           1       0.57      0.07      0.12      5106

    accuracy                           0.93     76259
   macro avg       0.75      0.53      0.54     76259
weighted avg       0.91      0.93      0.91     76259
```

Figure 3.16: Evaluation metrics for MLP.

```
Accuracy:  0.9342634967675947
Confusion Matrix:
 [[70958   195]
 [ 4818   288]]
Classification Report:
              precision    recall  f1-score   support

           0       0.94      1.00      0.97     71153
           1       0.60      0.06      0.10      5106

    accuracy                           0.93     76259
   macro avg       0.77      0.53      0.53     76259
weighted avg       0.91      0.93      0.91     76259
```

Figure 3.17: Evaluation metrics for LGBM.

```
Accuracy:  0.9345126476874861
Confusion Matrix:
 [[70830   323]
 [ 4671   435]]
Classification Report:
              precision    recall  f1-score   support

           0       0.94      1.00      0.97     71153
           1       0.57      0.09      0.15      5106

    accuracy                           0.93     76259
   macro avg       0.76      0.54      0.56     76259
weighted avg       0.91      0.93      0.91     76259
```

Figure 3.18: Evaluation metrics for XGBoost.

3.4.2. Regression Algoritms

In the regression part of our project, I aimed to predict the popularity of songs using decision tree algorithm. I first split our dataset into training and testing sets and trained our models on the training data. I then evaluated the performance of the models using metrics such as mean squared error, mean absolute error and R-squared score.

3.4.2.1. Decision Tree

For decision tree, I used the CART algorithm to fit our model and set the maximum depth of the tree as a hyperparameter. I also plotted the decision tree to better understand how the algorithm is making predictions. When we divide 20% of the dataset as test and 80% as train and fit the model, the R-squared score value will be negative value. R-squared is negative only when the chosen model does not follow the trend of the data. When I apply GridsearchCV, the R2 score value is 0.17. But still it is not enough. For max_depth = [10,15], min_samples_split = [6,8], min_samples_leaf = [5,6,7,8] values, GridSearchCv found Best max_depth: 10, Best min_samples_split: 8, Best min_samples_leaf: 8.

3.4.3. Ensemble Learning

In the ensemble model section of this project, I utilized a technique called stacking to improve the performance of our predictions. Stacking is a method of combining multiple models in order to achieve a more accurate overall prediction. In our implementation, I used four different base models: Random Forest Regressor, Gradient Boosting Regressor, Bagging Regressor, and AdaBoost Regressor with 10 estimators. These models were chosen because they have been shown to be effective in previous studies and have the potential to capture different patterns in the data.

To combine the predictions of these base models, I used a meta-model, which in this case was a Linear Regression model. The meta-model takes the predictions of the base models as input and generates a final prediction based on these inputs. This final prediction is then used as the final output of the ensemble model.

To evaluate the performance of our ensemble model, I used statistical tests such as mean squared error and R-squared. These tests allow us to compare the predictions of our ensemble model to the actual values and determine the level of accuracy achieved. Additionally, I also visualized our results by plotting the predicted values against the actual values to give a graphical representation of the performance of our ensemble model.

Table 3.1: Evaluation of the model.

Metric	Value
Mean Squared Error	175.09
Mean Absolute Error	10.38
R-Squared Score	0.346

In this case, the MSE is 175.09 which indicates that the model's predictions are, on average, 175.09 units away from the true values. Lower MSE values indicate a better fit of the model to the data. The MAE is 10.38 which indicates that the model's predictions are, on average, 10.38 units away from the true values. Lower MAE values indicate a better fit of the model to the data. The R-Squared Score ranges from 0 to 1, where a score of 1 indicates a perfect fit and a score of 0 indicates that the model does not explain any of the variability of the data. In this case, the R-Squared Score is 0.346 which indicates that the model explains 34.6% of the variability of the data. Higher R-Squared Score values indicate a better fit of the model to the data.

Figure 3.19: Actual vs. Predicted popularity comparison.

3.4.4. Artificial Neural Network(ANN)

For the Artificial Neural Network part, I created a model with 3 layers using Keras API. Below are some parameters and what they do.

- Batch learning: Instead of training the model on all the data at once, we can train it on small batches of data. This can be done by setting the ”batch_size” parameter in the fit() function.
- Incremental learning: Instead of training the model on all the data at once, we can train it incrementally by passing a generator function to the fit() function.
- Early stopping: You can stop training the model when the performance on the validation set stops improving. This can be done by setting the ”early_stopping” parameter in the fit() function.

My model has an input layer with 14 neurons, representing the 14 features in your dataset, and one output layer with 1 neuron. The activation function used for the input layer is ’relu’ (rectified linear unit) and no activation function is used for the output layer as it is a regression problem. The model is compiled with mean squared error as the loss function and Adam as the optimizer. I added BatchNormalization and Dropout layers to the model. Batch normalization is used to normalize the activations of the previous layer at each batch, which can stabilize the learning process and improve the performance of the model. Dropout is a regularization technique that randomly drops out some neurons during training, which helps to prevent overfitting. I added early stopping, which will stop training when the model stops improving on the validation data. The loss value is around 10. A loss of around 10 for a regression problem is relatively high and indicates that the model is not performing well in terms of predicting the target variable. This could be due to a number of factors such as poor feature selection, lack of sufficient data, or an inadequate model architecture.With this model structure, I trained models for 3 separate datasets. I divided the dataset into 3 as 1990s, 2000s and 2010s. I trained each part separately. Below you will see the result graphs for these 3 models.

```
Model: "sequential_15"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_64 (Dense)            (None, 64)                960

 batch_normalization_51 (Bat  (None, 64)               256
 chNormalization)

 dropout_51 (Dropout)        (None, 64)                0

 dense_65 (Dense)            (None, 32)                2080

 batch_normalization_52 (Bat  (None, 32)               128
 chNormalization)

 dropout_52 (Dropout)        (None, 32)                0

 dense_66 (Dense)            (None, 16)                528

 batch_normalization_53 (Bat  (None, 16)               64
 chNormalization)

 dropout_53 (Dropout)        (None, 16)                0

 dense_67 (Dense)            (None, 8)                 136

 batch_normalization_54 (Bat  (None, 8)                32
 chNormalization)

 dropout_54 (Dropout)        (None, 8)                 0

 dense_68 (Dense)            (None, 1)                 9

=================================================================
Total params: 4,193
Trainable params: 3,953
Non-trainable params: 240
_________________________________________________________________
```

Figure 3.20: ANN model summary

```
history1 = model1.fit(x_train_1990s, y_train_1990s,  epochs=10, batch_size=10, validation_data=(x_test_1990s, y_test_1990s), callbacks=[early_stopping])

8548/8548 [==============================] - 49s 6ms/step - loss: 15.5309 - mae: 15.5309 - val_loss: 11.2604 - val_mae: 11.2604
Epoch 2/10
8548/8548 [==============================] - 47s 6ms/step - loss: 12.8066 - mae: 12.8066 - val_loss: 11.0784 - val_mae: 11.0784
Epoch 3/10
8548/8548 [==============================] - 48s 6ms/step - loss: 12.3937 - mae: 12.3937 - val_loss: 11.0041 - val_mae: 11.0041
Epoch 4/10
8548/8548 [==============================] - 47s 5ms/step - loss: 12.0727 - mae: 12.0727 - val_loss: 10.9895 - val_mae: 10.9895
Epoch 5/10
8548/8548 [==============================] - 38s 4ms/step - loss: 11.8107 - mae: 11.8107 - val_loss: 10.9686 - val_mae: 10.9686
Epoch 6/10
8548/8548 [==============================] - 39s 5ms/step - loss: 11.6248 - mae: 11.6248 - val_loss: 10.9484 - val_mae: 10.9484
Epoch 7/10
8548/8548 [==============================] - 38s 4ms/step - loss: 11.4661 - mae: 11.4661 - val_loss: 10.8847 - val_mae: 10.8847
Epoch 8/10
8548/8548 [==============================] - 37s 4ms/step - loss: 11.3507 - mae: 11.3507 - val_loss: 10.8363 - val_mae: 10.8363
Epoch 9/10
8548/8548 [==============================] - 46s 5ms/step - loss: 11.2511 - mae: 11.2511 - val_loss: 10.8507 - val_mae: 10.8507
Epoch 10/10
8548/8548 [==============================] - 46s 5ms/step - loss: 11.2062 - mae: 11.2062 - val_loss: 10.8869 - val_mae: 10.8869
```

Figure 3.21: Learning - Model for 1990

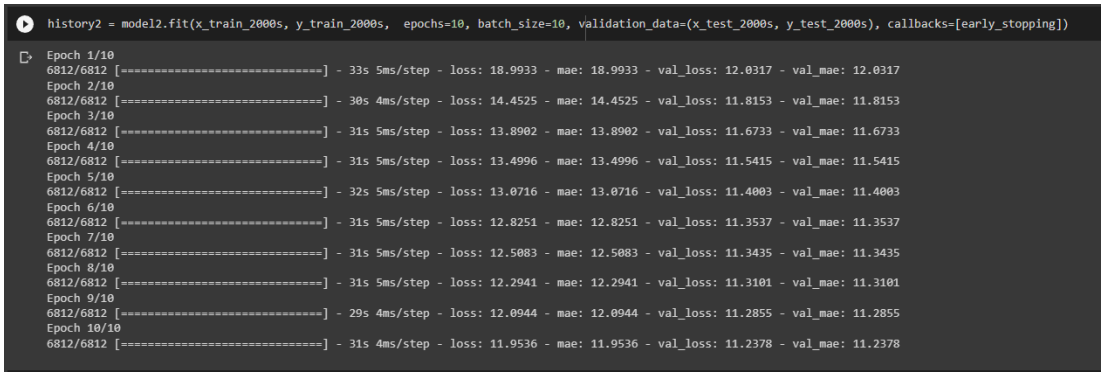

Figure 3.22: Learning - Model for 2000

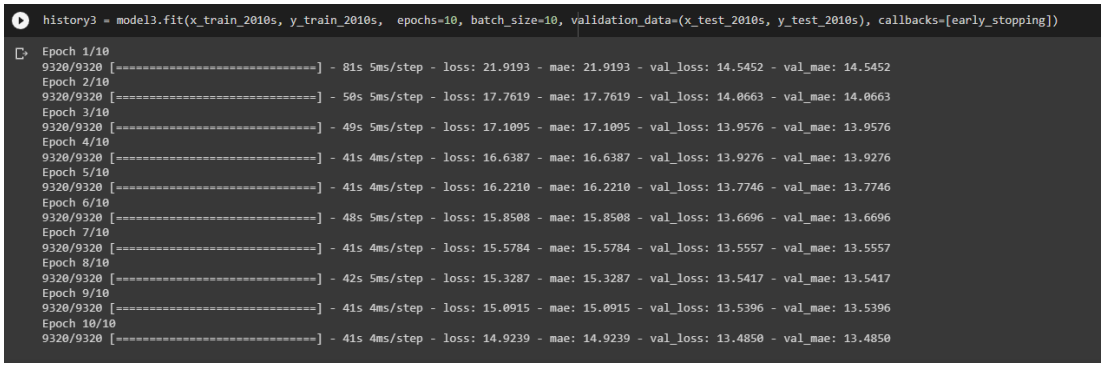

Figure 3.23: Learning - Model for 2010

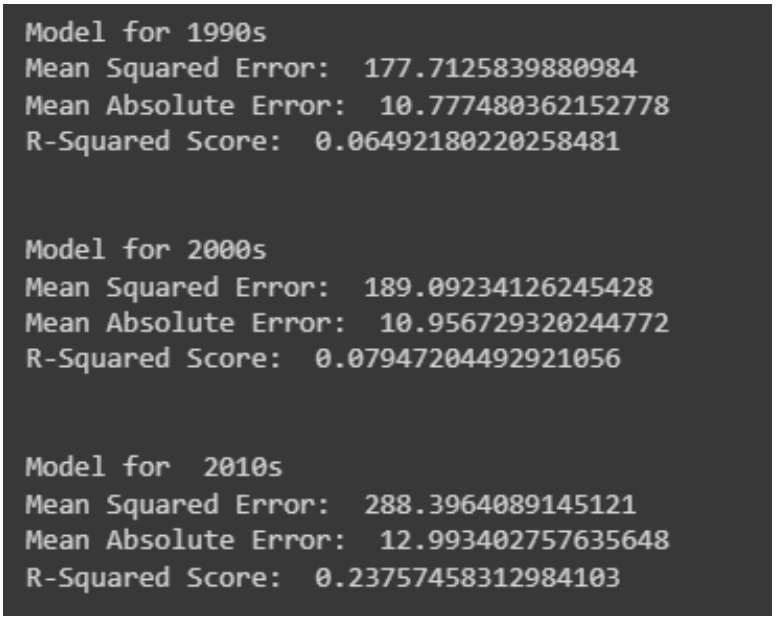

Figure 3.24: Evaluation of Artificial Neural Network(ANN)

Figure 3.25: Actual vs Predicted popularity for 1990s model

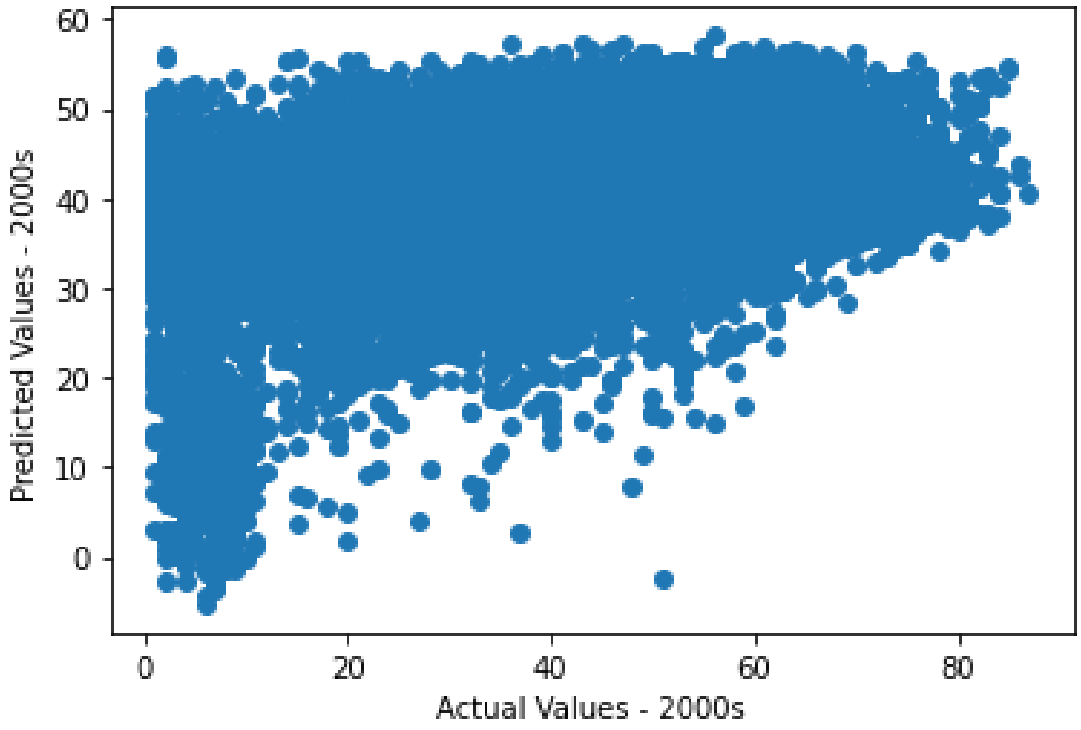

Figure 3.26: Actual vs Predicted popularity for 2000s model

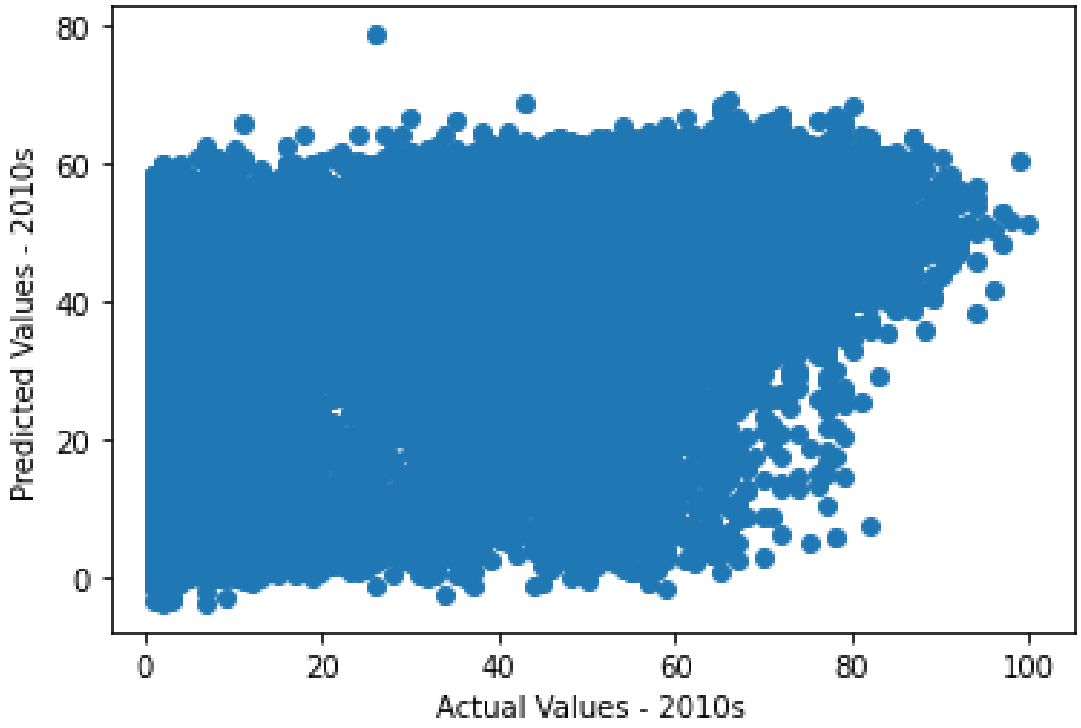

Figure 3.27: Actual vs Predicted popularity for 2010s model

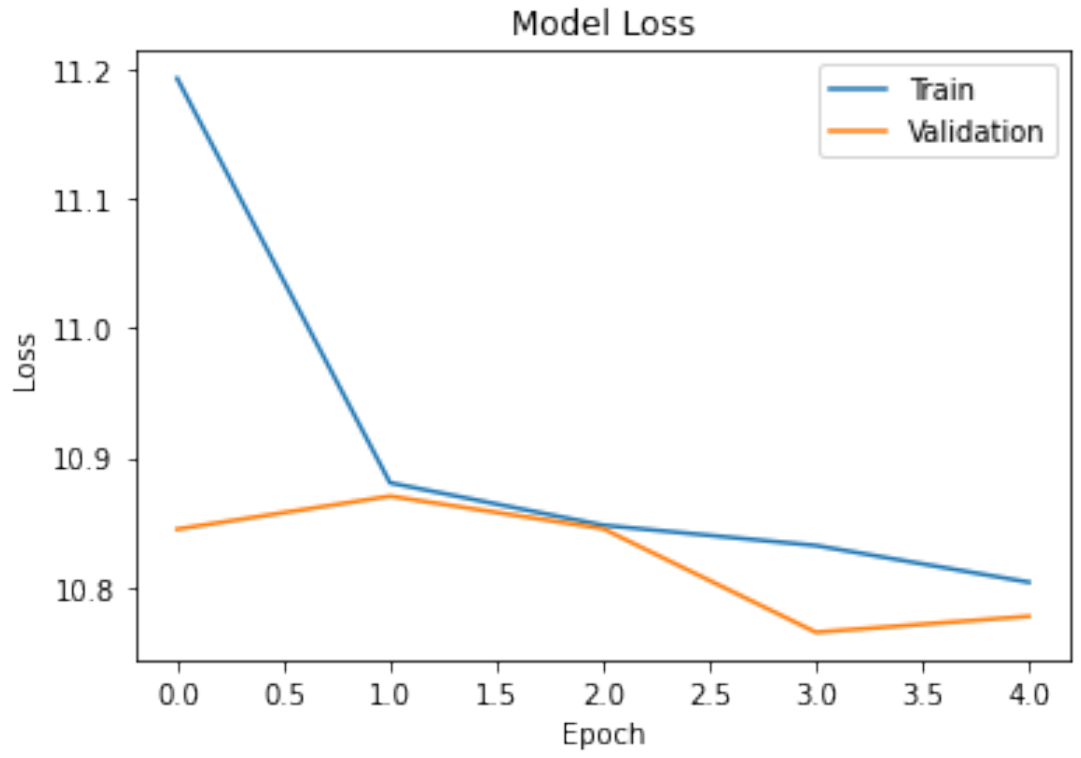

Figure 3.28: Result Page

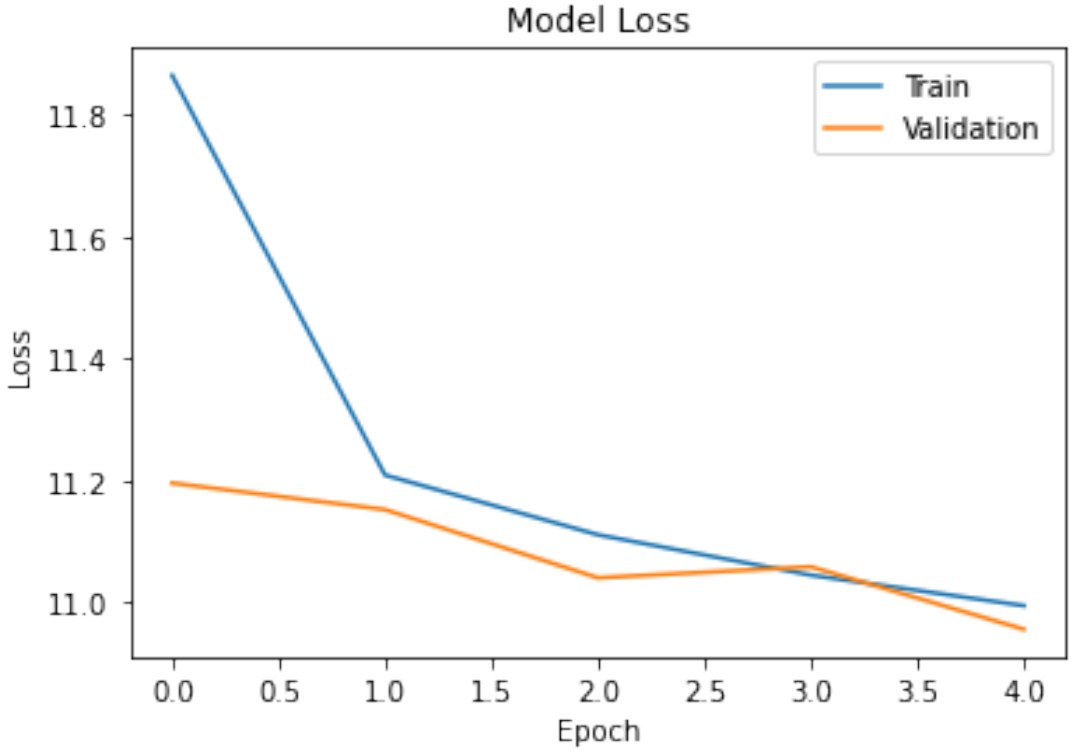

Figure 3.29: Result Page

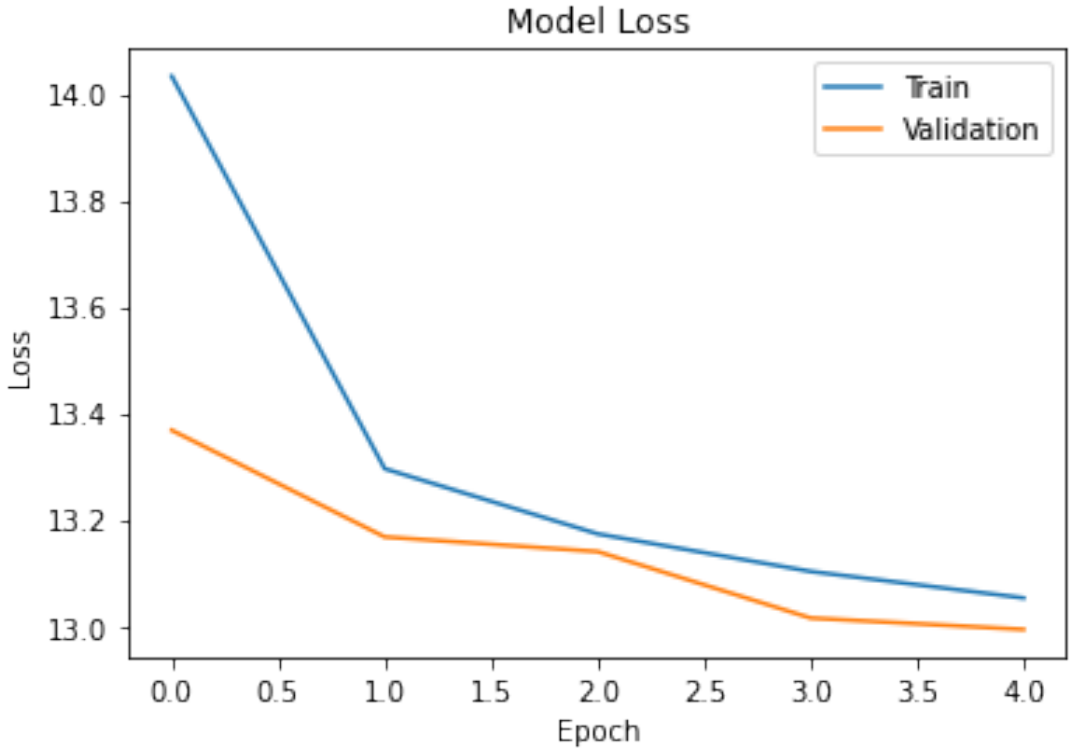

Figure 3.30: Result Page

3.5. Website Development

The website, built using VueJs, allows users to login with their Spotify credentials. The Auth Flow followed has been the Implicint Grant Flow from Spotify Auth Guide [13], as there's no Server Side code in this project. Spotify User login will be requested to use this app. Once logged in, the user is directed to the search page where they can search for songs. The songs are retrieved from the Spotify API and displayed in a list. The user can select a song from the list and view its features such as duration, release date, and popularity. Additionally, the user can listen to the song on the website. The website also includes a feature that allows the user to change the year of the song and view the predicted popularity for that year. The website is hosted on Netlify, making it easily accessible to users.

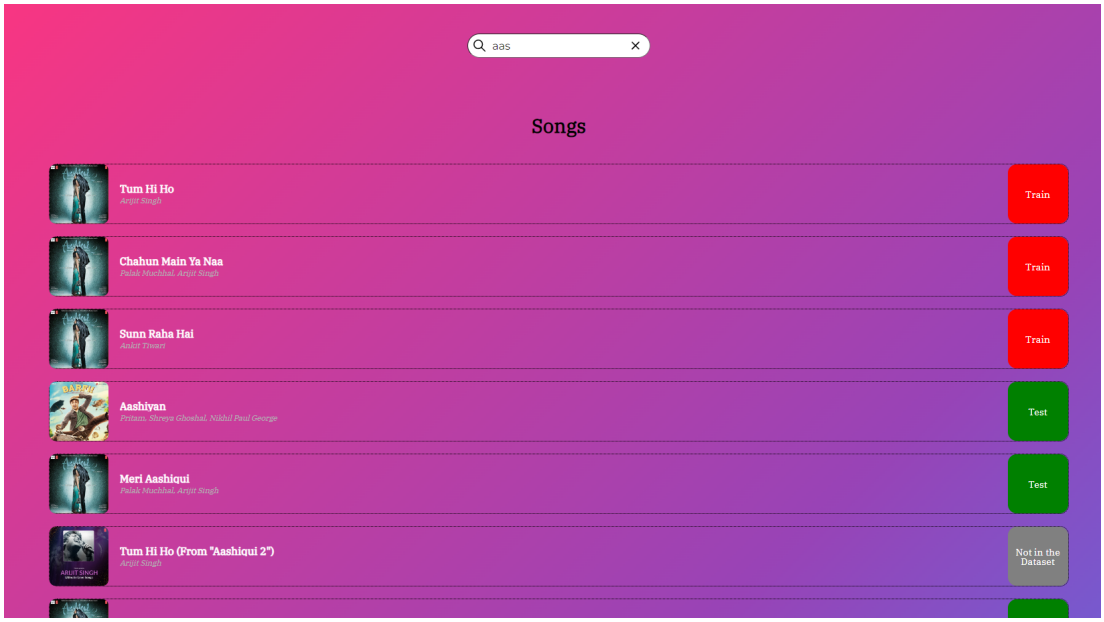

Figure 3.31: Result Page

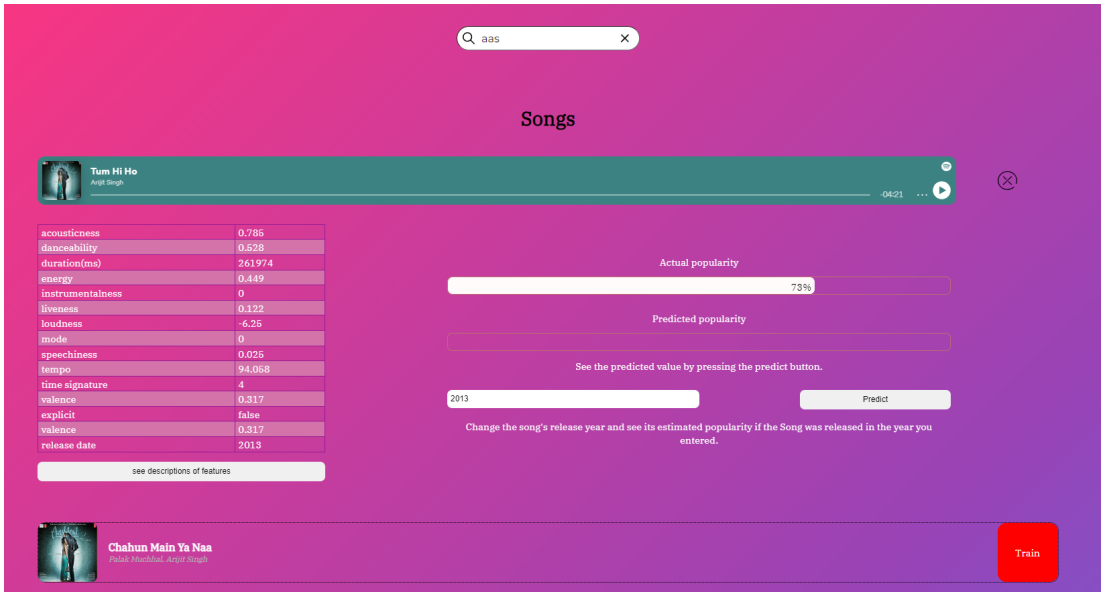

Figure 3.32: Track's properties card

4. RESULTS

In this article, I presented a study on the prediction of song popularity using machine learning techniques. I collected a dataset from ”Spotify Dataset 1921-2020, 600k+ Tracks” which has 586672 tracks. This dataset is taken from Kaggle and contains features such as song id, name, popularity, duration, explicit, artists, release date, danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo, and time signature.[12]

The data preprocessing steps included removing features not useful such as song_id, song_name, artistid, artist-name, and using one-hot encoding for mode and key features. Additionally, I removed songs before 1980 as the data was imbalanced, and removed songs with popularity 0. I also found outliers with Z-score solution and removed them, as well as applied PCA and feature selection to some models.

I performed Exploratory Data Analysis (EDA) and found histograms, boxplots, and a correlation matrix of the features to understand the distribution and relationship of the features.

For the classification part, I implemented several models including Logistic Regression, K-Nearest Neighbors, Multi-Layer Perceptron, Light Gradient Boosting Machine, and XGBoost. I trained and evaluated these models with different metrics such as F1-score, AUC-ROC, and confusion matrix. I also visualized the results with ROC curves and F1-score plots.

For the regression part, I implemented linear regression and decision tree regression models. I also applied RandomizedSearchCV for hyperparameter tuning and cross-validation for both models to find the best parameters.

I also implemented an ensemble model which is Stacking Regressor. This model uses 4 base models (Random Forest, Gradient Boosting, Bagging, and AdaBoost) and a meta-model (Linear Regression) to improve the performance.

Additionally, I used VueJs to build a website and hosted it on Netlify. Users can log in with their Spotify account and search for songs. The website also shows the predicted popularity of the songs.

Finally, I developed a neural network model using Keras. The model has 2 hidden layers with 64 and 32 neurons respectively, and the optimizer used is Adam. The performance of the model has a loss of about 10, which can be improved by using different techniques such as increasing the number of neurons in the hidden layers, using different types of optimizers, and using techniques such as early stopping and regularization.

In conclusion, the study showed that machine learning techniques can be used to

predict song popularity with reasonable accuracy. However, more research is needed to improve the performance of the models, and to find the most effective features and techniques for this problem.

5. CONCLUSION AND FUTURE WORK

In conclusion, this research proposed a machine learning-based approach for predicting music popularity on streaming platforms using data from Spotify. The approach utilizes various classification and regression methods, an ensemble method, and a neural network model using Keras. The experimental results show that the proposed approach achieved high accuracy in predicting music popularity and the combination of multiple models and the use of a neural network improves the prediction performance.

The study highlights the importance of utilizing diverse set of data and various machine learning techniques in order to predict music popularity with high accuracy. The results of this research can provide valuable insights for music producers, artists and music platforms to understand the factors that affect the popularity of music and make informed decisions about music production and promotion.

However, it should also be noted that there may be other factors that affect music popularity that were not considered in this research. Future studies should aim to include more features and to further understand the relationship between music popularity and other factors. Furthermore, The dataset used in this research could be improved by collecting more recent data from Spotify and other streaming platforms, and including more features that might affect the popularity of music.

BIBLIOGRAPHY

[1] H. Oh, "Predicting music popularity with the hybrid approach: K-means + lgbm," *International Journal of Data Science and Analysis*, vol. 8, no. 5, pp. 149–156, 2022.

[2] A. M. R. Jigisha Kamal Pankhuri Priya and S. G. R, "A classification based approach to the prediction of song popularity," 2021.

[3] R. N. Katherine Lin, "Predicting song popularity," 2017.

[4] K. S. Kai Middlebrook, "Song hit prediction: Predicting billboard hits using spotify data," 2019.

[5] M. I. I. Chyntia Kumalasari Puteri, "Analysis of song popularity in business digital music streaming for increasing quality using kohonen som algorithm," *IPTEK Journal*, no. 5, 2019.

[6] H. SA, "Applying active learning in music popularity prediction," *2nd International Conference on Artificial Intelligence and Computer Engineering(ICAICE)*, 2021.

[7] J. Kim, "Music popularity prediction through data analysis of music's characteristics," *International Journal of Science, Technology and Society*, vol. 9, no. 5, pp. 239–244, 2021.

[8] K. N. Agha Haider Raza, "Predicting a hit song with machine learning: Is there an apriori secret formula?" *2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA)*, 2020.

[9] D. P. P. D. P. S. M. P. Pathak3 and M. N. Sakariya4, "Predicting music popularity using machine learning algorithm and music metrics available in spotify," *Journal of Development Economics and Management Research Studies (JDMS), A Peer Reviewed Open Access International Journal*, vol. 9, no. 11, pp. 10–19, 2022.

[10] T. G. M. K. S. Yasmin Essa Adnan Usman, "Predicting the song popularity using machine learning algorithm," *International Journal of Scientific Research & Engineering Trends*, vol. 8, no. 2, 2022.

[11] J. N. M. Joshua S. Gulmatico Julie Ann B. Susa Mon Arjay F. Malbog Aimee Acoba Marte D. Nipas, "Spotipred: A machine learning approach prediction of spotify music popularity by audio features," *2022 Second International Conference on Power, Control and Computing Technologies (ICPC2T)*, 2022.

[12] Yamac Eren Ay. “Spotify dataset 1921-2020, 600k+ tracks.” (2021), [Online]. Available: https://www.kaggle.com/datasets/yamaerenay/spotify-dataset-19212020-600k-tracks?select=tracks.csv (visited on 01/15/2023).

[13] Spotify. “Authorization.” (), [Online]. Available: https://developer.spotify.com/documentation/general/guides/authorization/ (visited on 01/15/2023).

CV

Abdurrahman BULUT
Darica/Kocaeli
+90 (544) 439 4021
abdurrahmanbulut08@gmail.com
https://www.linkedin.com/in/abdurrahman-bulut/

Summary:

- A 4th-year computer science student at GTU with 4 months of experience as a frontend developer at XenonSmart and 6 months of experience as an intern application engineer at IBTech. Strong skills in C++, Java, JavaScript, Python, Angular, Vue.js, and some machine learning. Familiar with C# and data structures. English proficiency at B2 level with a certificate from ITEP.

Education:

- Computer Science, Gebze Technical University, Expected Graduation Date: 2023
- Mechanical Engineering, Gebze Technical University, 2017 - 2019

Experience:

- Frontend Developer, XenonSmart, March 2022 - July 2022

  Developed and maintained the Frontend of the company's web applications Collaborated with the design and development teams to ensure a seamless user experience Utilized technologies such as JavaScript and Vue.js.

- Intern Application Engineer, IBTech, August 2022 - Present

  Assisted in the development and maintenance of the company's software applications Worked on projects using Javascript, Angular, and C#

Skills:

- Proficient in C++, Java, JavaScript, Python, Angular, Vue.js, and some machine learning Familiar with C# and data structures Strong problem-solving and analytical skills

Certifications:

- ITEP English Language Proficiency, ITEP, April 2022

Languages:

- English, B2 level

APPENDICES

Data Collection: The data for this project was obtained from a dataset on Kaggle called ”Spotify Dataset 1921-2020, 600k+ Tracks” which contains 586672 tracks. This dataset was prepared by an individual who accessed the Spotify official API to collect the data. The dataset includes features such as id, name, popularity, duration_ms, explicit, artists, id_artists, release_date, danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo, and time_signature.

Data Preprocessing: The first step in preprocessing the data was to remove features that were not useful for the project such as song_id, song_name, artistid, and artist-name. One-hot encoding was used for the mode and key features. The release_date feature was edited to only include the year as the day and month parts were not necessary for the project. Songs released before 1980 were removed from the dataset as the music data from before this year was not representative. Songs with a popularity of 0 were also removed from the dataset. The duration_ms feature was converted to duration_s by dividing by 100. Outliers were found using the Z-score solution and were removed from the dataset as removing them improved the results of some classification problems. PCA was applied to some models before training them and feature selection steps were also applied to some models.

Exploratory Data Analysis: A histogram view of all features was created to visualize the distribution of the data. A boxplot was also created to visualize the distribution of the data and identify outliers. A correlation matrix was created to identify any correlation between the features.

Model Training and Evaluation: The project focused on both classification and regression problems. For the classification problem, five models were trained: Logistic Regression, K-Nearest Neighbors, Multi-Layer Perceptron, Light GBM, and XGBoost. Evaluation metrics such as accuracy, precision, recall, and F1-score were calculated for each model. The ROC curve was also plotted to visualize the results of the models. For the regression problem, two models were trained: Linear Regression and Decision Tree. The mean squared error, mean absolute error, and R-squared score were calculated to evaluate the models.

Ensemble Model: To improve the performance of the models, an ensemble model was created by stacking four base models (Random Forest, Gradient Boosting, Bagging, and AdaBoost) and a meta-model (Linear Regression). The ensemble model improved the performance of the models.

Web Development: The project’s website was developed using VueJs and was hosted on Netlify. Users can log in using their Spotify account and search for songs on

the website. The website displays the predicted popularity of the song along with its actual popularity.

Neural Network: A neural network model was implemented using Keras. The model was trained on 10 data at a time and the mean of the learning results was used as the final result. The model was trained on 14 features and had two hidden layers. The model's performance was evaluated using the mean absolute error and the loss was found to be around 10.