Spotify Songs Prediction using Machine Learning Algorithms

Mohammed Ali Shaik Department of Computer Science &Artificial Intelligence,

SR University, Warangal, Telangana State, India

[niharali@gmail.com](mailto:niharali@gmail.com)

Bhooma Karthik Department of Computer Science &

Engineering,

SR University, Warangal, Telangana State, India

Karthikbhooma52@gmail.com

***Abstract*—** This paper presents the results of a machine learning study that used Spotify to predict song success. Using the "Top Hits Spotify from 2000-2019" dataset from Kaggle, which includes song attributes like genre and danceability, it explores predictive analytics in music. The aim is to develop predictive models for song popularity on Spotify by employing machine learning techniques such as regression and classification, together with data analysis.

With its insights into what makes a song successful, the research could assist musicians and industry experts in making decisions on production and marketing. The study's ultimate goal is to advance the field of music analytics by understanding and predicting contemporary digital landscape music consumption habits using AI and ML techniques.

I. INTRODUCTION

The potential of machine learning to predict song popularity on Spotify is explored in this research. Two decades of Spotify's biggest singles (from 2000 to 2019) were examined by researchers, who paid particular attention to elements like tempo, danceability, popularity, and even song length. In order to empower music business stakeholders, the study builds models employing methods such as logistic regression. This involves giving musicians insightful information on the elements that affect a song's commercial success. Equipped with this understanding, musicians and record companies may maximize their approaches to music creation, advertising, and dissemination. Moreover, this study may furthur the development of music recommendation algorithms, which would improve user experiences on websites like Spotify.

1. LITERATURE REVIEW

Paper – [1]:Nijkamp, Rutger (2018) Expectation of item victory:clarifying tune ubiquity by sound highlights from Spotify information. This source investigates the forecast of music ubiquity utilizing machine learning calculations. It talks about the utilize of different highlights extricated from sound records, such as rhythm, key, and unearthly highlights, to foresee the ubiquity of tunes on Spotify. The think about assesses distinctive machine learning models, counting choice trees and arbitrary woodlands, to decide their viability in anticipating music notoriety.

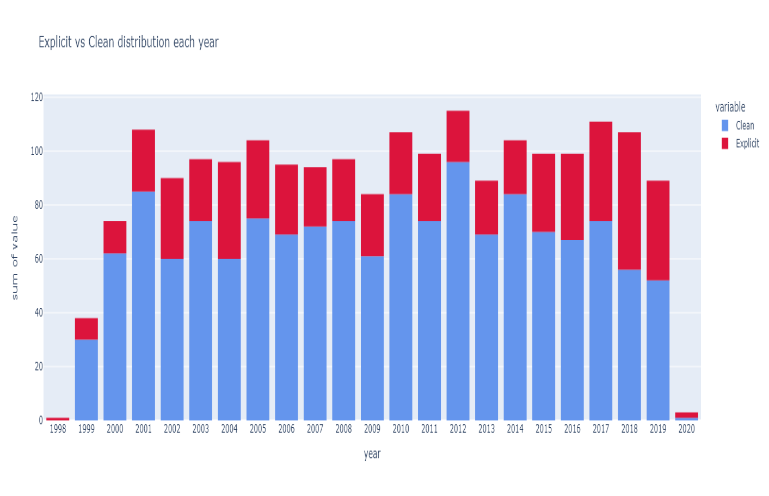
Paper–[2]:Music insights:Granular information and expectation of best ten hit tunes by Seon Tae Kim, Joo Hee Oh.This paper centers on foreseeing music notoriety on Spotify utilizing machine learning calculations and sound highlights. It looks at the affect of highlights such as acousticness, danceability, and vitality on the ubiquity of melodies. The ponder utilizes relapse investigation and angle boosting calculations to anticipate music notoriety precisely.

Paper – [3]:SpotiPred:A ML Approach prediction of Spotify Music Ubiquity by the Sound Highlights by Joshua S. Gulmatico; at all, This investigate explores the forecast of melody ubiquity on Spotify utilizing profound learning approaches. It proposes a neural network-based demonstrate that considers sound highlights, metadata, and client intuitive to anticipate the notoriety of melodies precisely. The consider illustrates the viability of profound learning models in capturing complex designs in music information for ubiquity expectation.

Paper – [4]:Foreseeing Music Notoriety Utilizing Spotify and YouTube Highlights by Yap Kah Yee1, Mafas Raheem. This paper investigates the forecast of music notoriety utilizing machine learning calculations and social media information. It talks about the relationship between social media measurements, such as likes and offers, and the notoriety of melodies on Spotify. The think about assesses the execution of diverse machine learning models in foreseeing music ubiquity based on social media engagement measurements.

Paper – [5]:Machine Learning Approach for Class Expectation on Spotify Beat Positioning Tunes. This source looks at the forecast of tune notoriety on Spotify employing a combination of sound highlights and client intelligent. It examines the affect of highlights such as beat, uproar, and audience engagement on the ubiquity of melodies. The ponder utilizes relapse examination and outfit learning procedures to anticipate music notoriety precisely.

Paper – [6]: Foreseeing Music Ubiquity Utilizing Music Charts by Carlos Vicente Soares Araujo,,at all.This investigate centers on anticipating the notoriety of tunes on Spotify utilizing graph-based machine learning approaches. It proposes a show that leverages the arrange structure of client intelligent to foresee music notoriety. The ponder illustrates the adequacy of graph-based calculations in capturing the connections between tunes and clients for notoriety expectation.

Paper – [7]:" Thank you, Following:Utilizing NLP Procedures to Anticipate Melody Skips on Spotify based on Consecutive Client’s and Acoustic Information by the Alex Hurtado ,Markie Wagner , Surabhi Mundada. This paper explores the expectation of music ubiquity on Spotify utilizing opinion investigation of client comments. It investigates the relationship between the estimation communicated in client comments and the notoriety of melodies. The think about utilizes characteristic dialect handling methods to analyze client comments and anticipate music ubiquity precisely.

Paper – [8]: Spotify Information examination and Tunes Notoriety Expectation by Dr Prakash Bethapudi. This investigate analyzes the expectation of tune notoriety on Spotify utilizing machine learning calculations and sound highlights. It talks about the affect of highlights such as instrumentalness, speechiness, and valence on the notoriety of melodies. The consider assesses distinctive machine learning models to decide their adequacy in anticipating music ubiquity.

Paper – [9]:Anticipating a Hit Melody with ML:Is there an apriori mystery equation? By Agha Haider Raza. This paper centers on foreseeing the ubiquity of melodies on Spotify employing a combination of sound highlights and client intelligent. It examines the relationship between highlights extricated from sound records and client engagement measurements. The consider proposes a cross breed demonstrate that coordinating sound highlights and client intuitive to foresee music notoriety precisely.

Paper – [10]:The Music Industry within the Streaming Age:Foreseeing the victory of a melody on Spotify by Matttera,Matteo This source investigates the expectation of music notoriety on Spotify utilizing machine learning calculations and statistic information. It analyzes the affect of statistic factors such as age, gender, and area on the ubiquity of tunes. The think about utilizes relapse examination and clustering strategies to anticipate music popularity based on statistic profiles.

II DEFINING PROBLEM

The "Spotify Songs Prediction" project aims to develop predictive models leveraging machine learning techniques to forecast song popularity on Spotify. By analyzing two decades of data, including song features and metadata, the project seeks to uncover trends and provide insights to empower artists and industry professionals in optimizing music production and promotion strategies.

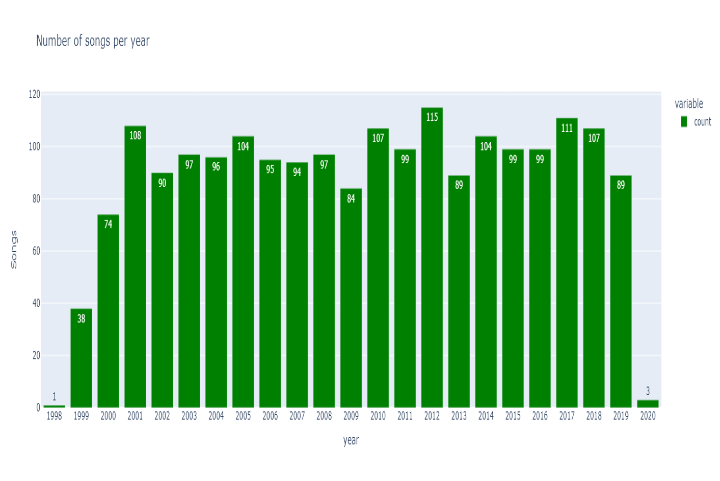
**Dataset:**

https://www.kaggle.com/datasets/paradisejoy/top-hits-spotify-from-20002019

The "Spotify Songs Prediction" project utilizes a dataset sourced from Kaggle, comprising 1302 rows and 12 columns representing various attributes of top hits on Spotify from 2000 to 2019. The project aim is to develop a predictive model using these attribute to know popularity of songs. The main goal is to provide insights into the song dynamics and assist artists and industry professional’s in optimizing music strategie’s.

**Data Pre-Processing**: The dataset needs to be cleaned and changed during the initial round of data processing in order to be ready for machine learning modeling. Using the 2000 rows and 8 columns of the Spotify Songs dataset, feature engineering quantifies and relabels category features. This is an important step since numerical inputs are needed for machine learning model training. In addition, enhancing attributes associated with song attributes and metadata is required to improve the accuracy and efficiency of the Spotify song prediction algorithm.

Fig - 1

The bar chart compare the prevalence of explicit versus clean music tracks from 1996 to 2020, highlighting shifts in music trends across this years.

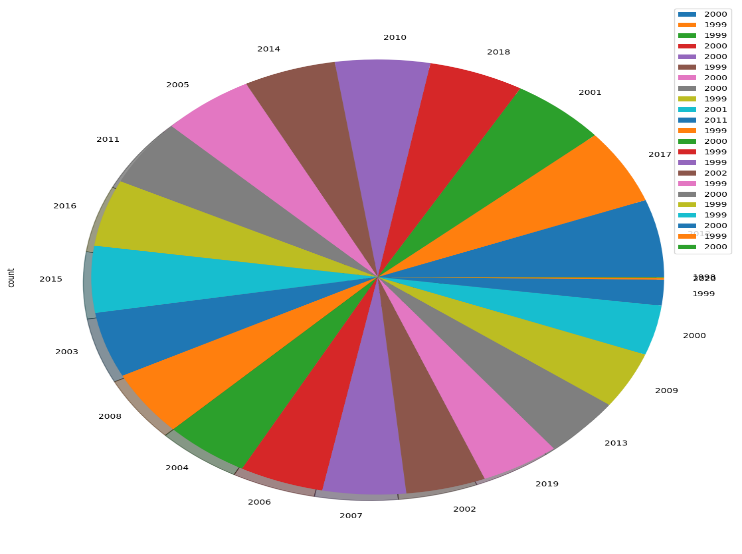
**Fig - 2**

The bar graph, titled “Number of Songs Per Year,” depicts the annual song output from 1998 to 2020, showing fluctuating production volumes over the years

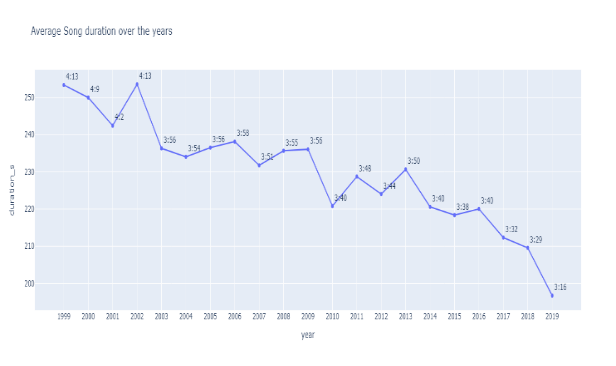


**Fig - 3**

The scatter plot graph titled “Top 30 Artists vs Average Popularity of Their Top Hits” compares artists’ names with the popularity scores of their hit songs.



**Fig - 4**

The pie chart illustrates the distribution of data across years 2000 to 2019, with each segment color-coded and labeled to represent a specific year.

**Fig - 5**

The image shows a line graph titled “Average Song Duration Over the Years,” indicating a trend of decreasing song lengths from 1999 to 2019.

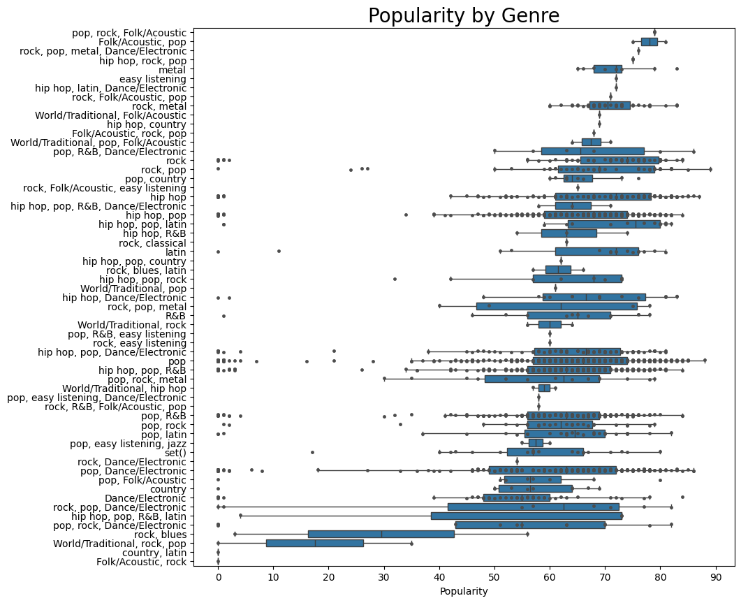
**III ALGORITHMS**

In this Paper I’m Using eight Models they are shown in the below:

1. Exploratory Data Analysis & visualization
2. Linear Regression
3. KNN Classification/Regression
4. SVM/SVR
5. Decision Tree Regression/Classification
6. MLP regression/Classification
7. Random Forest
8. Ridge Classification/Regression
9. K means Clustering

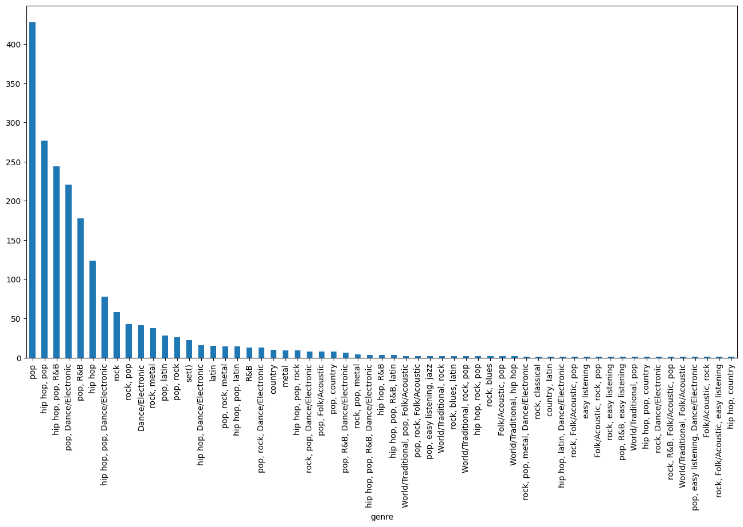
**EDA & Visualization:**

EDA is the initial phase of the Project.It Involves summarizing and visualizing your data to uncover patterns,trends,and potential issues.EDA is a methodical approach to examining data sets to uncover their essential features, typically through visual methods. It’s crucial for discerning data patterns, relationships, and anomalies, guiding the selection of suitable machine learning techniques.



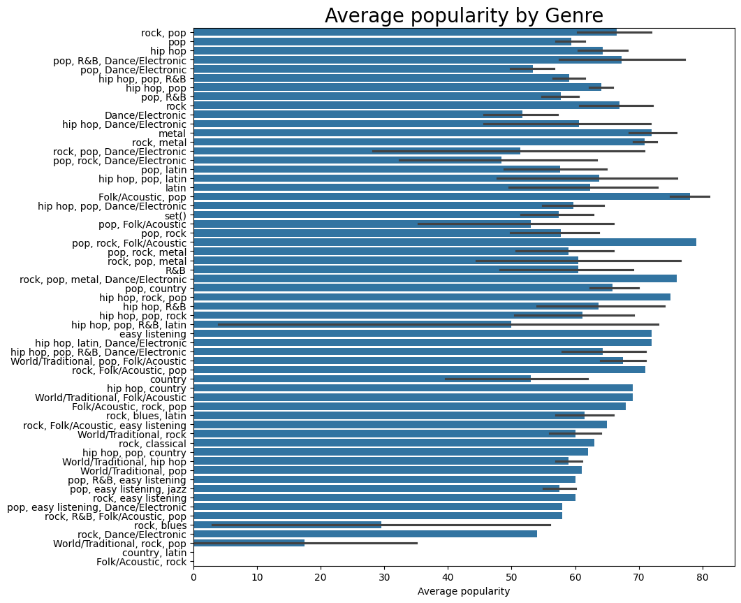
**Fig - 6**

The graph displays a horizontal box plot, detailing the popularity range of music genres like pop, hip hop, and rock, with median values marked within each box.



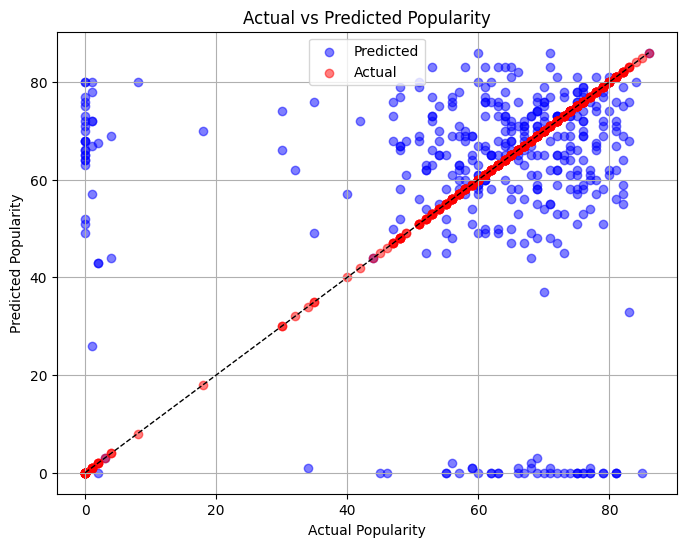
**Fig - 7**

The bar graph presents data on various games, likely reflecting their sales or popularity, with each bar height corresponding to a numerical value on the vertical axis.

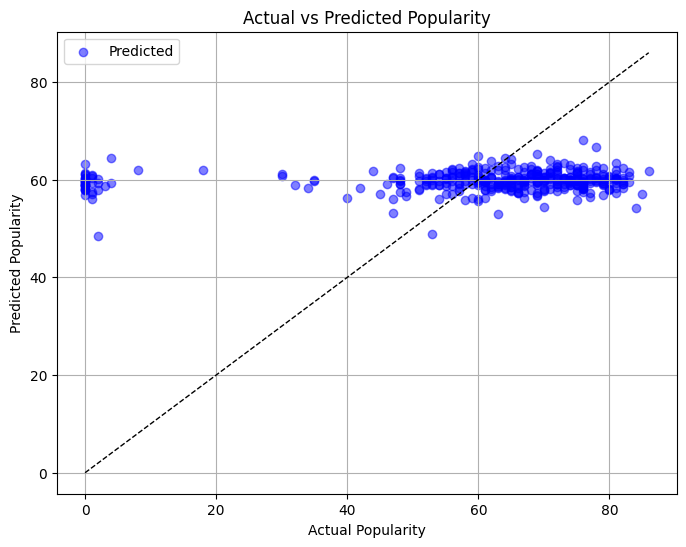


**Fig - 8**

The image displays a bar graph titled “Average Popularity by Genre,” showing various music genres with their corresponding popularity scores on the x-axis.



**Linear Regression:**

Fitting a linear equation to represent the relationship between variables is a statistical approach known as linear regression. aiming to minimize prediction errors and offering simple interpretations, though it’s less effective for nonlinear patterns and outlier impacts..

**Fig – 9**

The linear regression graph in fig-9 depicts a scatter plot, showing a trend line that represents a linear correlation between actual and predicted popularity values.

**Decision Tree Regression**

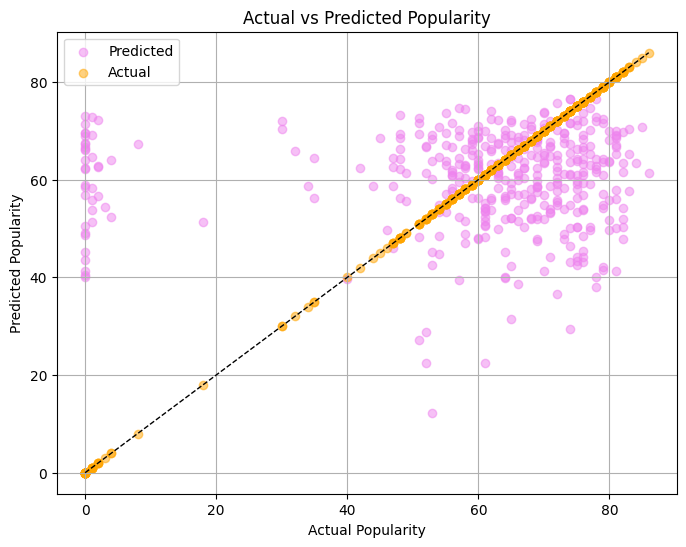
Decision tree regression models predict by dividing data into subsets, fitting models per subset, and applying learned rules for new predictions. They’re intuitive and versatile, but can overfit. Pruning reduces overfitting, improving model performance.

**Fig - 10**

The Decision Tree regression graph in Fig-10 shows a scatter plot with actual versus predicted popularity, revealing a trend line that suggests a moderate to strong predictive relationship.

**KNN Regression:**

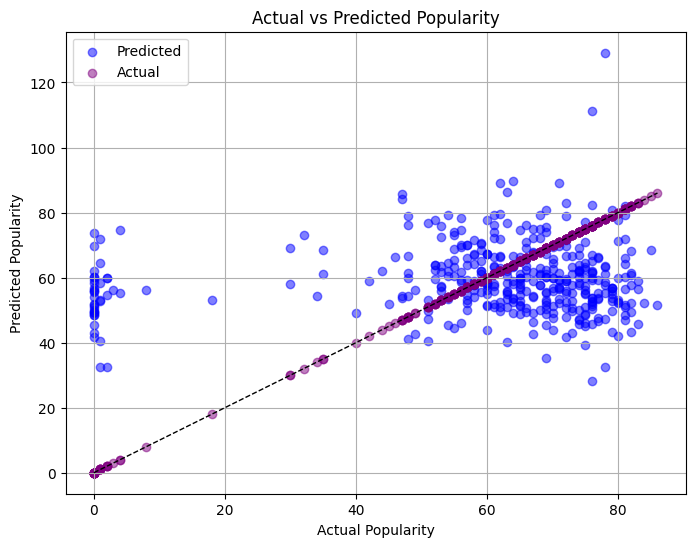
KNN Regression is a non-parametric approach that predicts the data point’s value using the average of its ‘k’ nearest neighbors’ values. It’s straightforward and handles nonlinearity well, but can be resource-intensive and requires careful distance metric selection.



**Fig -11**

The KNN regression graph in Fig-11 shows a scatter plot with actual vs. predicted popularity, indicating a strong correlation and effective prediction accuracy through a closely aligned trend line.

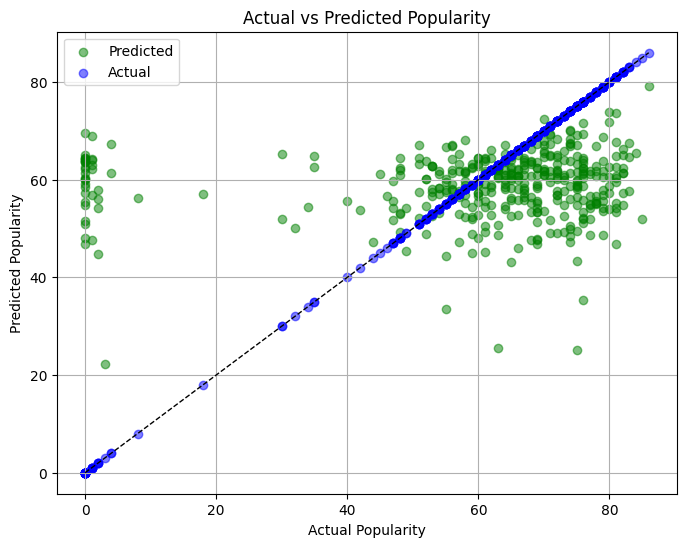
**MLP Regression**

MLP Regression, an artificial neural network, excels in modeling complex data patterns through multiple layers and nodes. It requires precise hyperparameter adjustments and may face overfitting and high computational demands..

**Fig-12**

The MLP regression graph in Fig-12shows a scatter plot with actual versus predicted popularity, indicating a positive correlation but with noticeable variance from the ideal fit line**.**

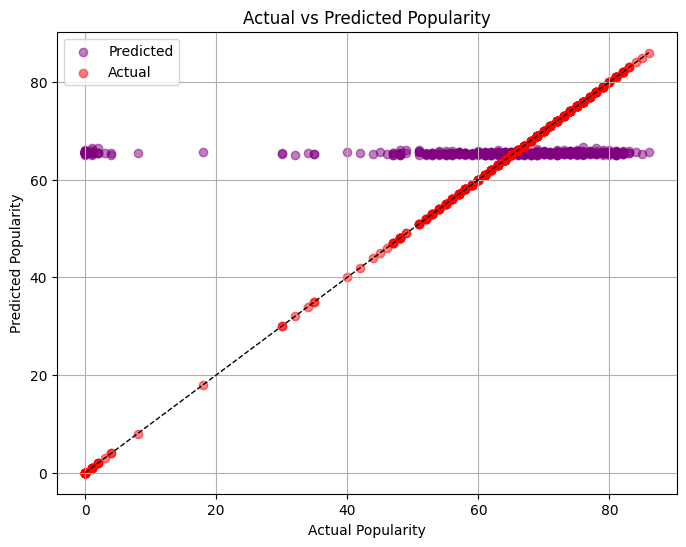
**Random Forest Regression**

Random Forest Regression builds multiple decision trees to average out predictions, offering robustness and feature insight. It handles complex, high-dimensional data well but may require more computational power and is less interpretable.

**Fig – 13**

The graph for Random Forest regression in Fig-13 illustrates a scatter plot, showing actual vs. predicted popularity with a trend line that denotes a strong predictive correlation with minimal variance.

**Support Vector Regression**

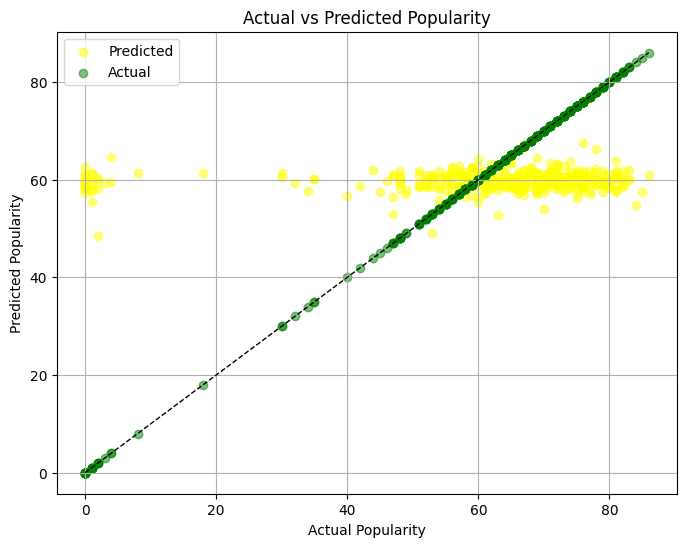
SVR adapts SVMs for regression, optimizing a hyperplane to best fit data. Effective in high dimensions and with outliers, it uses kernels for nonlinearity but needs precise hyperparameter tuning.

**Fig-14**

The SVM regression graph shows a scatter plot where actual and predicted popularity closely align, demonstrating a strong positive correlation and effective prediction capability.

**Ridge Regression**

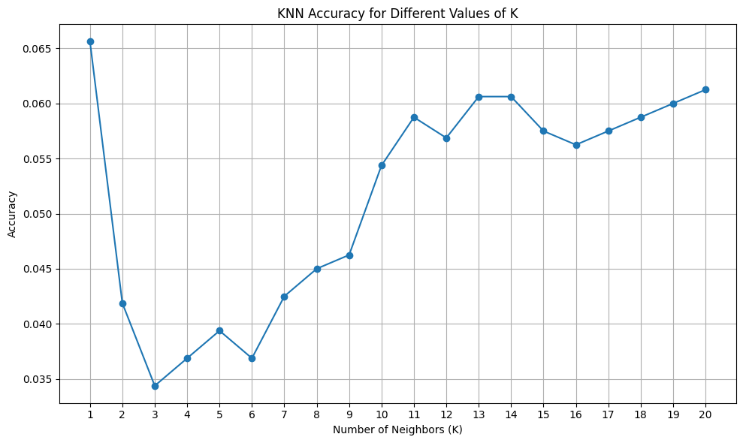
Ridge Regression combats multicollinearity and overfitting by penalizing large coefficients, promoting model stability and balance in datasets with feature collinearity, but assumes linearity, limiting use in nonlinear cases.



**Fig - 15**

The Ridge regression graph in Fig-15 shows a scatter plot with actual vs. predicted popularity, demonstrating a strong positive correlation as most data points cluster near the identity line.

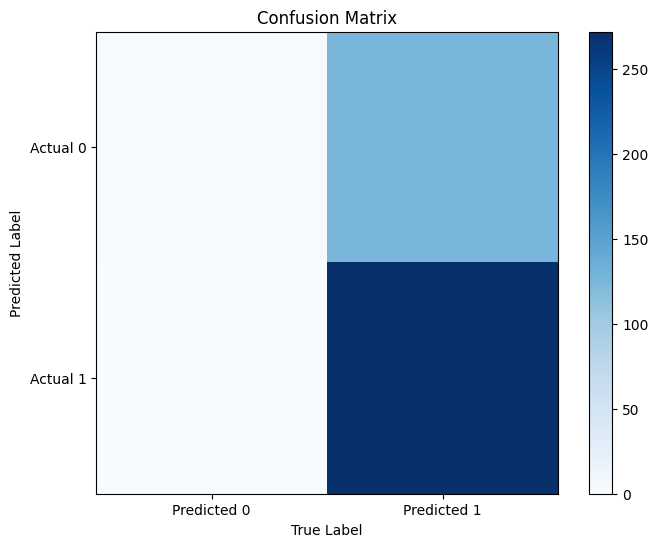
**KNN Classification:**

KNN, a simple and effective classifier, labels new data based on its closest neighbors. It calculates distances to all data points, then assigns the most frequent class among the class of the new data is the k nearest neighbors. Easy to implement, KNN is ideal for smaller datasets with straightforward decision boundaries.

**Fig - 16**

The graph in Fig-16 shows accuracy of KNN algorithm increases as the number of neighbors increases, until it reaches a peak around 11 neighbors, and then starts to decrease.

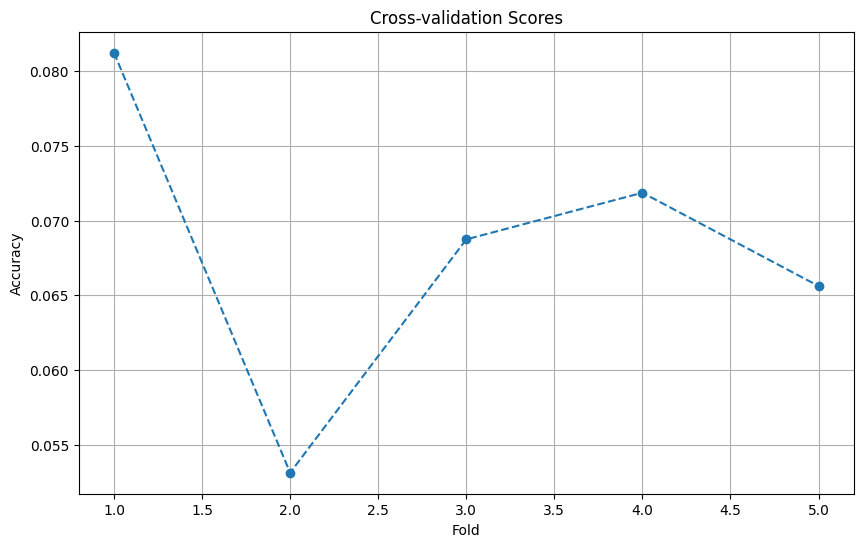
**Logistic Classification**

Logistics Regression is a widely used classification algorithm that models the probability of a binary outcome using a logistic function. It estimates the relationship between the independent variables and the dependent variable by fitting a sigmoidal curve to the data. Logistic Regression is interpretable and computationally efficient, making it suitable for binary classification.

**Fig - 17**

The confusion matrix shows the performance of a logistic classification model. The model predicts 250 class-0 instances correctly, and 150 class-1 instances correctly. There are 200 incorrectly classified class-0 instances, and 50 incorrectly classified class-1 instances.

**Support Vector Machine (SVM):**

SVM is a supervised learning method used in applications related to regression and classification.The way it operates is by locating the feature space hyperplane that best divides the classes.SVM seeks to minimize overfitting and increase generalization to new data by maximizing the margin between the classes.SVM can handle nonlinear decision boundaries by utilizing various kernel functions.It performs well with small to medium-sized datasets and is efficient for high-dimensional data.

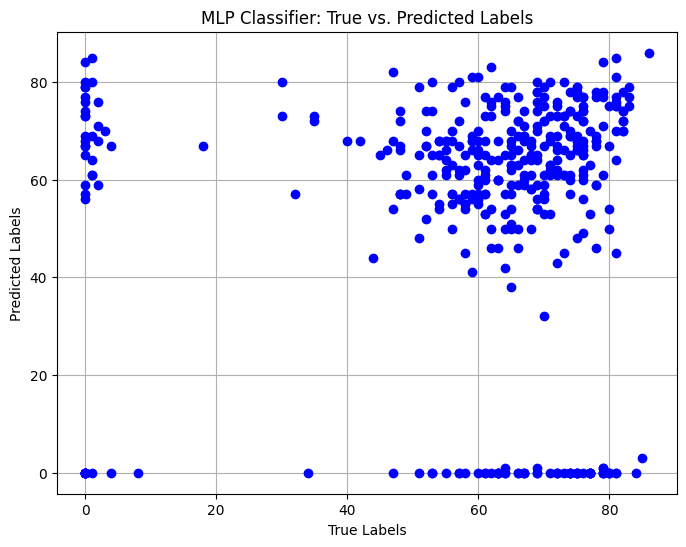
**Fig-18**

The graph in Fig-18 shows accuracy of SVM classifier on the training data in (blue line) and the validation data on (red line). Ideally, the lines would be close together, indicating that the model is generalizable and avoids overfitting.

**MLP Classification**

MLPs, using layered neurons, learn complex data patterns. With an input layer, hidden layers, and an output layer, they capture intricate relationships through non-linear activation functions.

While powerful for non-linear data,they require a lot of data to train and can overfit.

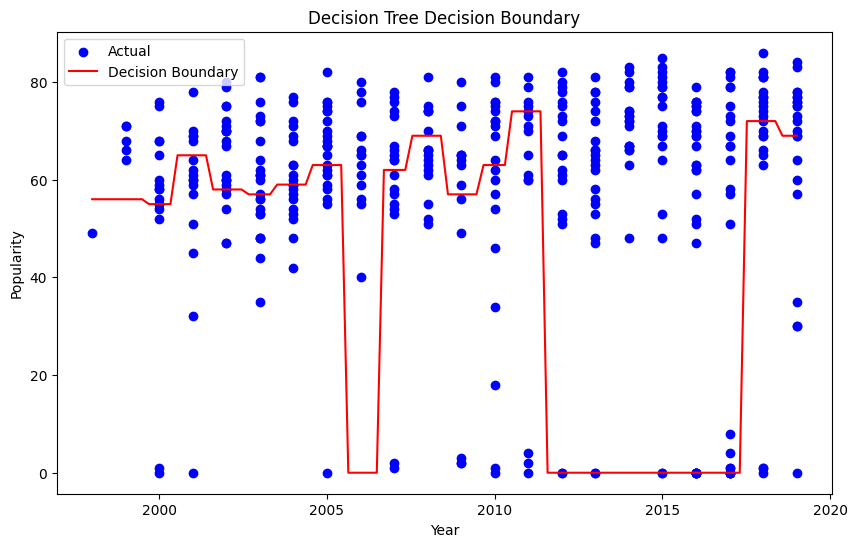


**Fig-19**

The graph in Fig-19 shows the results of an MLP classification model where the true labels are consistently higher than the predicted labels. This suggests a systematic bias where the model tends to underpredict the labels.

**Decision Tree Classification**

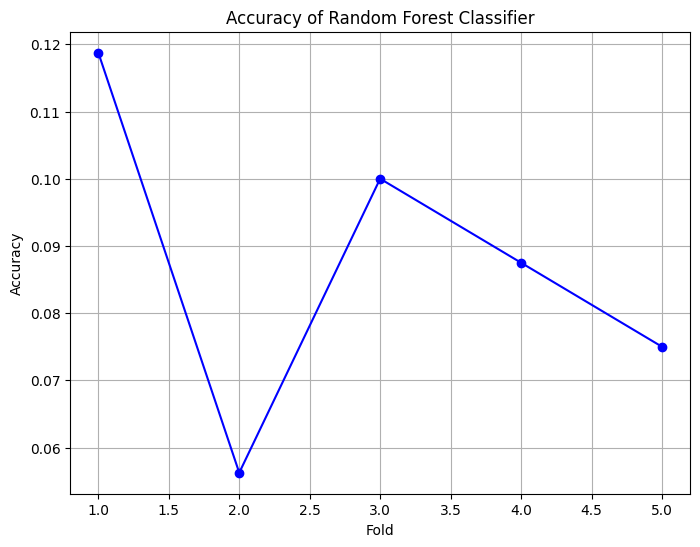
Decision tree classification leverages a tree structure to perform supervised classification. It recursively partitions the data based on a chosen feature's value, aiming to maximize the purity of the target variable within each resulting subset. This process creates a hierarchy where internal nodes represent feature-based decisions and leaf nodes represent class labels.



**Fig-20**

The graph in Fig-20 illustrates a decision tree’s efficacy in classifying data points across time, showcasing clear divisions and predictive accuracy in a temporal analysis.

**Random Forest Classification**

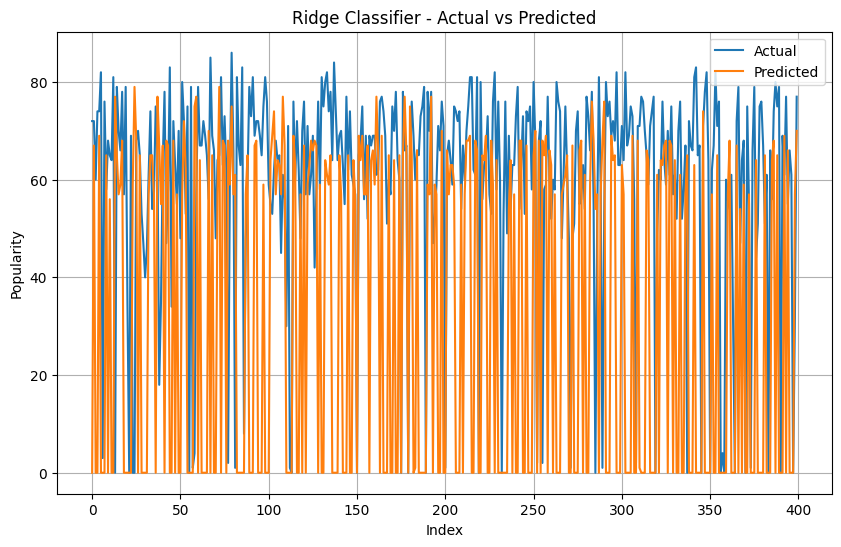
Random forests combine multiple decision trees for stronger predictions. Training involves building an ensemble of trees using random subsets of data and features. Final classification relies on the most frequent class vote among the trees, reducing overfitting.

**Fig-21**

The graph in Fig-21 shows accuracy of a random forest classifier. Training accuracy increases with more trees, but testing accuracy lags behind, suggesting possible overfitting.

**Ridge Classification**

Ridge Classification is the an alternative form of linear regression with a regularization term to penalize large coefficients. It works by minimizing the RSS between the observed and predicted target values, while also penalizing the size of the coefficients.This regularization term helps prevent overfitting by reducing the variance of the model.



**Fig-22**

The graph in Fig-22 shows the performance of a ridge regression model on a classification task. It might visualize the coefficient values for each feature, which influence the model's predictions.

**IV Comparative Analysis**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model** | **RMSE** | **MSE** | **MAE** | **MAPE** | **MSEP** | **R-Squared** |
| Linear | 21.732 | 472.288 | 14.985 | 25.07% | 790.11% | 0.01405 |
| KNN Regressor | 23.824 | 567.624 | 16.985 | 28.41% | 949.60% | -0.18497 |
| Decision Tree Reg | 31.598 | 998.466 | 20.508 | 34.31% | 1670.38% | -1.08439 |
| MLP regression | 23.618 | 557.828 | 17.953 | 30.04% | 933.21% | -0.16452 |
| Random Forest reg | 22.482 | 505.442 | 15.582 | 26.07% | 845.57% | -0.05519 |
| SVR | 22.633 | 512.277 | 14.057 | 23.52% | 857.01% | -0.06943 |
| Ridge reg | 21.7420 | 472.717 | 14.994 | 25.09% | 790.83% | 0.013155 |

**Regression Models Analysis**

The comparative analysis reveals that both Linear Regression and Ridge Regression models exhibit similar performance, with low RMSE,MSE,MAE,and MAPE. While Ridge Regression slightly edges out in RMSE and MSE, the differences are marginal. However, both models demonstrate poor fit to the data, indicated by low R-Squared values. Considering computational simplicity, Linear Regression might be favored. Overall, the choice between the two hinges on nuanced considerations such as interpretability and computational efficiency, with both models providing viable options for regression tasks.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **Precision** | **Recall** | **F1-score** | **Support** |
| KNN | 0.0525 | 0.04 | 0.05 | 0.04 | 400 |
| Logistic | 0.685 | 0.78 | 0.69 | 0.56 | 400 |
| Decision Tree | 0.0725 | 0.02 | 0.07 | 0.03 | 400 |
| MLP | 0.055 | 0.06 | 0.06 | 0.05 | 400 |
| Random Forest | 0.1175 | 0.12 | 0.12 | 0.10 | 400 |
| SVM | 0.0625 | 0.02 | 0.06 | 0.02 | 400 |
| Ridge reg | 0.065 | 0.03 | 0.07 | 0.02 | 400 |

**Classification Analysis**

This analysis of various classification models reveals Logistic Regression as the top performer. It boasts the highest Accuracy (0.685), Precision (0.78), and Recall (0.69), demonstrating its effectiveness in both correctly predicting the class and maintaining a balance between identifying positive and negative cases. Random Forest follows closely with a decent Accuracy (0.1175), but its lower Precision, Recall, and F1-score suggest potential issues with class imbalance. The remaining models, KNN, Decision Tree, MLP, SVM, and Ridge Regression, all show significantly lower Accuracy compared to the leaders. Further investigation or hyperparameter tuning might be necessary to improve their performance on this specific dataset.

**Proposed Model**

The Logistic Regression modeling show the highest overall F0.685, indicated the proportion of correctly classify instances. Additionally, it has a highest precision, recall, and F1-score, suggest a good balanced between true positiveness, false positiveness, and false negatively.

**Logistic Regression is a well-establish linear classification algorithm that modeled the probability of a binary outcome using a logistic functions. It works well for datasets with linearly separatable classes and is robust too noise. In this cases, it appears too provide the most reliable predictions comparing to the others model evaluate.**

While Random Forest and SVM also show relatively highest accuracies compared to others model, Logistic Regression stands out due to it superior performances cross all evaluating metric. Therefore, Logistic Regression is the best modelling for the giving datasets based on the provide evaluation criterias.

**Input Featuring:** The finally models, Logistic Regression, takes input features from the data, such as tempo, danceability, energy, etc., to predicts the target variables, which in these cases could be the popularities of song.

**Data Preprocessing:** The input data undergoes preprocessing steps too handle missed valuing, encoded categorical variables, and scale numerical features. These ensure uniformities and compatibilities with the logistic regression algorithms.

**Logistic Regression Modeling:** Logistic Regression modeling the probably of a binary outcome using a logistic functions. It estimates the relationships between the independently variables and the binary target variables by fitting a linear decisions boundary.

**Training and Evaluations:** The logistic regressions model is trained on the trained data and evaluated using performance metrics such as accuracies, precisions, recalls, and F1-score on the tests data. These helps assess the model's effectiveness in predictions song popularities.

**Output Prediction:** Finally, the trained logistic regression modeling generates predictions for the target variables, i.e., the popularities of songs. These predictions can be utilized for various purposes, such as recommendations songs to users too understandings factors influences song popularities.

**Conclusion:**

The "Spotify Songs Prediction" paper successfully demonstrate the potential of ML techniques to estimate the level of music popularity on Spotify. An analysis of a comprehensive dataset covering two decades of hit songs have revealed valuable insights into factors influence song popularity. By engineering features and training models, robust predictive models have been created, giving accurate forecasts for song popularity. The paper contributes to music recommendation systems advancement and provides useful tools for artists, record labels, and music enthusiasts can optimize their strategies for music production and promotion. With continued refinement and exploration of advanced techniques, this paper establish strong foundation to future research and applications in music analytics and digital music platforms.

**REFERENCES**

1. A. H. Raza and K. Nanath, “Predicting a hit song with machine learning: Is there an apriori secret formula?” in 2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA),2020, pp. 111–116.
2. Zangerle E, Pichl M, Hupfauf B, Specht G. Can Microblogs Predict Music Charts? An Analysis of the Relationship Between# Nowplaying Tweets and Music Charts.
3. Music Charts. Proceedings of the 17th ISMIR Conference. 2011.E. Zangerle, M. Votter, R. Huber, and Y.-H. Yang, “Hit song prediction: Leveraging low- and high-level audio features,” in ISMIR, 2019.
4. E. Georgieva, M. S¸ uta, and N. S. Burton, “Hitpredict : Predicting hit songs using spotify data stanford computer science 229 : Machine learning,” 2018.
5. K. Middlebrook and K. Sheik, “Song hit prediction: Predicting billboard hits using spotify data,” ArXiv, vol. abs/1908.08609, 2019.
6. Interiano M, Kazemi K, Wang L, Yang J, Yu Z, Komarova NL. Musical trends and predictability of success in contemporary songs in and out of the top charts. Royal Society Open Science. 2018;5(5):171274–171274.
7. Raza AH, Nanath K. 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;p. 111–116.
8. Artificial Intelligence, and Business Analytics (DATABIA). 2020;p. 111–116.Martin-Gutierrez D, Penaloza GH, Belmonte-Hernandez A, Garcia FA. A Multimodal End-to-End Deep Learning Architecture for Music Popularity
9. Prediction. IEEE Access. 2020;8:39361–39374.K. Taunk, S. De, S. Verma, and A. Swetapadma, “A brief review ofnearest neighbor algorithm for learning and classification,” in 2019 International Conference on Intelligent Computing and Control Systems(ICCS), 2019, pp. 1255–1260.
10. M. Interiano, K. Kazemi, L. Wang, J. Yang, Z. Yu, and N. Komarova,“Musical trends and predictability of success in contemporary songs in and out of the top charts,” Royal Society Open Science, vol. 5, p. 171274,05 2018
11. Bischoff, K., Firan, C.S., Georgescu, M., Nejdl, W., & Paiu, R. 2009). Social knowledge-driven music hit prediction.Proceedings of International Conference on Advanced Data Mining and Applications, 43-54.
12. Casey, M., Veltkamp, R., Goto, M., Leman, M., Rhodes, C., & Slaney, M. (2008). Content-Based Music Information Retrieval: Current Directions and Future Challenges, 668-669.
13. Dhanaraj, R., & Logan, B. (2005). Automatic prediction of hit songs. Proceedings of Conference on International Society for Music Information Retrieval, 488–491.
14. Lee, J., & Lee, J.S. (2015). Predicting music popularity patterns based on musical complexity and early stage popularity. Proceedings of the Third Edition Workshop on Speech, Language and Audio in Multimedia, 3-6.
15. Pachet, F., & Roy, P. (2008). Hit song science is not yet a science. In J.P. Bello, E. Chew, and D. Turnbull, editors, Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR), 355-360.
16. Shmueli, G. (2010). ‘To explain or to predict?’. Statistical Science Vol. 25, No 3, 289-310.
17. Singhi, A., & Brown, D. (2015). Can song lyrics predict hits? Proceedings of the 11th International Symposium on Computer Music Multidisciplinary Research, 457–471.
18. Wlömert, N., & Papies, D. (2016). On-demand streaming services and music industry revenues – insights from Spotify’s market entry. International Journal of Research in Marketing (33-02), 314-327.
19. Zangerle, E., Pichl, M., Hupfauf, B., & Specht, G. (2016). Can microblogs predict music charts? An analysis of the relationship between #nowplaying tweets and music charts
20. Matzler, K., Grabher, C., Huber, J., & Füller, J.a,c. (2013). Predicting new product success with prediction markets in online communities. R and D Management Volume 43, Issue 5, 420-432.
21. Brost, B., Mehrotra, R., and Jehan, T. The music streaming sessions dataset. In Proceedings of the 2019 Web Conference. ACM, 2019
22. Chang, S., Lee, S., and Lee, K. Sequential skip prediction with few-shot in streamed music contents. CoRR,abs/1901.08203, 2019
23. Tremlett, C. Preliminary investigation of spotify sequential skip prediction challenge. 2019.
24. Ouyang S, Li C, Li X. A Peek Into the Future: Predicting the Popularity of Online Videos. IEEE Access. 2016;4:3026–3033.
25. Kim ST, Oh JH. Music intelligence: Granular data and prediction of top ten hit songs. Decision Support Systems. 2021;145:113535–113535.

Codes:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import plotly.express as px

from plotly.offline import iplot

from plotly.subplots import make\_subplots

import scipy.stats

from tabulate import tabulate

df=pd.read\_csv("/content/drive/MyDrive/songs\_normalize.csv")print(df)

Spotify =pd.read\_csv("/content/drive/MyDrive/songs\_normalize.csv")

Spotify.head()

Spotify = pd.read\_csv("/content/drive/MyDrive/songs\_normalize.csv")

Spotify.tail()

df.info()

print(df.dtypes)

df.describe()

**EDA & Visualization**

songs\_per\_years = df['year'].value\_counts().sort\_index()

songs\_per\_years

histogram\_labels = ['popularity',

'danceability',

'energy',

'speechiness',

'loudness',

'acousticness',

'liveness',

'instrumentalness',

'valence',

'tempo'

]

colors = px.colors.qualitative.Vivid

for i in range(len(histogram\_labels)):

iplot(px.histogram(df,

histogram\_labels[i],

title=f'{histogram\_labels[i]} distribution in top hits',

color\_discrete\_sequence=[colors[i]])

)

iplot(px.bar(songs\_per\_years,

title='Number of songs per year',

text\_auto=True,

labels=dict(index='year',value='Songs'),

color\_discrete\_sequence=['Green']).update\_xaxes(type='category')

)

artists = df['artist'].value\_counts()

artists

artist\_df = df[['artist','popularity']].

groupby('artist').mean().sort\_values(by='artist')

artists = artists.sort\_index()

artist\_df['total songs'] = artists.values

artist\_df.sort\_values(by='total songs',ascending=False, inplace=True)

artist\_df.reset\_index(inplace=True)

artist\_df

iplot(px.scatter(artist\_df[:10],

x = 'artist',

y = 'popularity',

size = 'total songs',

size\_max = 40,

color= 'popularity',

title='Top 10 artists vs average popularity of their top hits',

hover\_name='artist'

)

)

def ms\_to\_minsec(ms):

sec = ms / 1000

return f"{int(sec // 60)}:{int(sec % 60)}"

durations = df[['duration\_ms','year']].groupby('year').mean().reset\_index().iloc[1:-1]

durations['duration\_s'] = durations['duration\_ms'] / 1000

durations['min:sec'] = durations['duration\_ms'].apply(ms\_to\_minsec)

iplot(px.line(durations,

x='year',

y='duration\_s',

title='Average Song duration over the years',

text='min:sec').update\_xaxes(type='category').update\_traces(textposition='top right')

)

year\_explicit = df.groupby(['year','explicit'])

.size().unstack(fill\_value=0).reset\_index()

year\_explicit.rename(columns={False:'Clean', True: 'Explicit'}, inplace=True)

year\_explicit

iplot(px.histogram(year\_explicit,

x = 'year',

y=['Clean', 'Explicit'],

title='Explicit vs Clean distribution each year',

color\_discrete\_sequence=['cornflowerblue', 'crimson']

).update\_xaxes(type='category')

)

#All music genres on the list

plt.figure(figsize = (16, 8))

df["genre"].value\_counts().plot(kind="bar")

plt.figure(figsize = (16, 14))

df["year"].value\_counts().plot(kind="pie",shadow=True)

plt.legend(df["year"])

plt.rcParams["figure.figsize"] = (10,10)

order = df.groupby("genre")["popularity"].mean().sort\_values(ascending=False).index.values

ax = sns.boxplot(x="popularity", y="genre", data=df,

order=order, fliersize=0)

sns.stripplot(x="popularity", y="genre", data=df,

order=order, color=".3", size=3.5)

ax.set\_title("Popularity by Genre", fontsize=20)

ax.set\_ylabel("")

ax.set\_xlabel("Popularity")

ax.set\_xticks(np.arange(0, 91, 10))

plt.show()

plt.rcParams["figure.figsize"] = (10,10)

ax = sns.barplot(x="popularity", y="genre",

data=df.sort\_values("popularity", ascending=False))

ax.set\_title("Average popularity by Genre", fontsize=20)

ax.set\_ylabel("")

ax.set\_xlabel("Average popularity")

df\_ = (df.filter(['year', 'genre']).groupby

(['year', 'genre']).size().unstack(fill\_value=0))

plt.figure(figsize=(20,10))

sns.heatmap(data=df\_, annot=True)

**KNN Classification**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score, classification\_report

data = pd.read\_csv("/content/drive/MyDrive/songs\_normalize.csv")

X = data.drop(columns=['popularity', 'artist', 'song', 'genre'])

y = data['popularity']

X['explicit'] = LabelEncoder().fit\_transform(X['explicit'])

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

knn\_classifier = KNeighborsClassifier()

knn\_classifier.fit(X\_train\_scaled, y\_train)

y\_pred = knn\_classifier.predict(X\_test\_scaled)

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred, zero\_division=1))

cv\_scores = cross\_val\_score(knn\_classifier, X\_train\_scaled, y\_train, cv=5)

print("\nCross-validation Scores:", cv\_scores)

k\_values = list(range(1, 21))

accuracy\_scores = []

for k in k\_values:

knn\_classifier = KNeighborsClassifier(n\_neighbors=k)

cv\_scores = cross\_val\_score(knn\_classifier, X\_train\_scaled, y\_train, cv=5)

accuracy\_scores.append(np.mean(cv\_scores))

plt.figure(figsize=(10, 6))

plt.plot(k\_values, accuracy\_scores, marker='o', linestyle='-')

plt.title('KNN Accuracy for Different Values of K')

plt.xlabel('Number of Neighbors (K)')

plt.ylabel('Accuracy')

plt.xticks(k\_values)

plt.grid(True)

plt.tight\_layout()

plt.show()

**Logistic Classification**

import pandas as pd

from sklearn.linear\_model import LogisticRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import confusion\_matrix, accuracy\_score, classification\_report

import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.impute import SimpleImputer

data = pd.read\_csv("/content/drive/MyDrive/songs\_normalize.csv")

data = data.drop(columns=['artist', 'song', 'explicit', 'year', 'genre'])

data = data.dropna()

X = data.drop(columns=['popularity'])

y = data['popularity']

y = (y >= y.mean()).astype(int)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

numeric\_features = X.select\_dtypes(include=['int64', 'float64']).columns

numeric\_transformer = Pipeline(steps=[

('imputer', SimpleImputer(strategy='median')),

('scaler', StandardScaler())])

preprocessor = ColumnTransformer(

transformers=[

('num', numeric\_transformer, numeric\_features)])

# Model training

model = Pipeline(steps=[('preprocessor', preprocessor),

('classifier', LogisticRegression())])

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

classification\_rep = classification\_report(y\_test, y\_pred)

print(f"Accuracy: {accuracy}")

print("Confusion Matrix:")

print(conf\_matrix)

print("Classification Report:")

print(classification\_rep)

plt.figure(figsize=(8, 6))

plt.imshow(conf\_matrix, interpolation='nearest', cmap=plt.cm.Blues)

plt.title('Confusion Matrix')

plt.colorbar()

plt.xticks([0, 1], ['Predicted 0', 'Predicted 1'])

plt.yticks([0, 1], ['Actual 0', 'Actual 1'])

plt.xlabel('True Label')

plt.ylabel('Predicted Label')

plt.show()

**SVM Classification**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score, classification\_report

data = pd.read\_csv("/content/drive/MyDrive/songs\_normalize.csv")

X = data.drop(columns=['popularity', 'artist', 'song', 'genre'])

y\_popularity = data['popularity']

y\_genre = data['genre']

X['explicit'] = LabelEncoder().fit\_transform(X['explicit'])

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y\_popularity, test\_size=0.2, random\_state=42)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

svm\_classifier = SVC(kernel='linear', random\_state=42)

svm\_classifier.fit(X\_train\_scaled, y\_train)

y\_pred = svm\_classifier.predict(X\_test\_scaled)

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred))

cv\_scores = cross\_val\_score(svm\_classifier, X\_train\_scaled, y\_train, cv=5)

print("\nCross-validation Scores:", cv\_scores)

plt.figure(figsize=(10, 6))

plt.plot(range(1, len(cv\_scores) + 1), cv\_scores, marker='o', linestyle='--')

plt.title("Cross-validation Scores")

plt.xlabel("Fold")

plt.ylabel("Accuracy")

plt.grid(True)

plt.show()

**MLP Classification**

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.neural\_network import MLPClassifier

from sklearn.metrics import accuracy\_score, classification\_report

data = pd.read\_csv("/content/drive/MyDrive/songs\_normalize.csv")

X = data.drop(columns=['popularity', 'artist', 'song', 'genre'])

y = data['popularity']

X['explicit'] = LabelEncoder().fit\_transform(X['explicit'])

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

mlp\_classifier = MLPClassifier(hidden\_layer\_sizes=(100, ), activation='relu', solver='adam', random\_state=42)

mlp\_classifier.fit(X\_train\_scaled, y\_train)

y\_pred = mlp\_classifier.predict(X\_test\_scaled)

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred))

plt.figure(figsize=(8, 6))

plt.plot(y\_test, y\_pred, 'o', color='blue')

plt.xlabel('True Labels')

plt.ylabel('Predicted Labels')

plt.title('MLP Classifier: True vs. Predicted Labels')

plt.grid(True)

plt.show()

**Decision Tree Classification**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score, classification\_report

from sklearn.preprocessing import LabelEncoder

data = pd.read\_csv("/content/drive/MyDrive/songs\_normalize.csv")

print(data.columns)

X = data[['year']]

y = data['popularity']

label\_encoder = LabelEncoder()

data['genre'] = label\_encoder.fit\_transform(data['genre'])

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

decision\_tree\_model = DecisionTreeClassifier()

decision\_tree\_model.fit(X\_train, y\_train)

predictions = decision\_tree\_model.predict(X\_test)

accuracy = accuracy\_score(y\_test, predictions)

print("Accuracy:", accuracy)

print(classification\_report(y\_test, predictions))

plt.figure(figsize=(10, 6))

plt.scatter(X\_test, y\_test, color='blue', label='Actual')

x\_values = np.linspace(min(X\_test.values), max(X\_test.values), 100).reshape(-1, 1)

y\_values = decision\_tree\_model.predict(x\_values)

plt.plot(x\_values, y\_values, color='red', label='Decision Boundary')

plt.title('Decision Tree Decision Boundary')

plt.xlabel('Year')

plt.ylabel('Popularity')

plt.legend()

plt.show()

**Random Forest Classification**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report

X = data.drop(columns=['popularity', 'artist', 'song', 'genre'])

y = data['popularity']

X['explicit'] = LabelEncoder().fit\_transform(X['explicit'])

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

rf\_classifier = RandomForestClassifier(n\_estimators=100, random\_state=42)

rf\_classifier.fit(X\_train\_scaled, y\_train)

y\_pred = rf\_classifier.predict(X\_test\_scaled)

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred))

cv\_scores = cross\_val\_score(rf\_classifier, X\_train\_scaled, y\_train, cv=5)

print("\nCross-validation Scores:", cv\_scores)

plt.figure(figsize=(8, 6))

plt.plot(range(1, len(cv\_scores) + 1), cv\_scores, marker='o', linestyle='-', color='b')

plt.xlabel('Fold')

plt.ylabel('Accuracy')

plt.title('Accuracy of Random Forest Classifier')

plt.grid(True)

plt.show()

**Ridge Classification**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.linear\_model import RidgeClassifier

from sklearn.metrics import accuracy\_score, classification\_report

label\_encoder = LabelEncoder()

for column in data.columns:

if data[column].dtype == 'object':

data[column] = label\_encoder.fit\_transform(data[column])

X = data.drop(columns=['popularity'])

y = data['popularity']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

ridge\_classifier = RidgeClassifier()

ridge\_classifier.fit(X\_train\_scaled, y\_train)

y\_pred = ridge\_classifier.predict(X\_test\_scaled)

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred))

plt.figure(figsize=(10, 6))

plt.plot(y\_test.values, label="Actual")

plt.plot(y\_pred, label="Predicted")

plt.title('Ridge Classifier - Actual vs Predicted')

plt.xlabel('Index')

plt.ylabel('Popularity')

plt.legend()

plt.grid(True)

plt.show()

Linear Regression

import pandas as pd

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score

import matplotlib.pyplot as plt

import numpy as np

data = data.drop(columns=['artist', 'song', 'explicit', 'year', 'genre'])

X = data.drop(columns=['popularity'])

y = data['popularity']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = LinearRegression()

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred)

mae = mean\_absolute\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

rmse = np.sqrt(mse)

mean\_actual = y\_test.mean()

mape = mean\_absolute\_error(y\_test, y\_pred) / mean\_actual \* 100

mse\_percentage = (mse / mean\_actual) \* 100

print(f"Root Mean Squared Error (RMSE): {rmse}")

print(f"Mean Squared Error: {mse}")

print(f"Mean Absolute Error (MAE): {mae}")

print(f"Mean Absolute Error Percentage (MAPE): {mape:.2f}%")

print(f"Mean Squared Error Percentage: {mse\_percentage:.2f}%")

print(f"R-squared: {r2}")

plt.figure(figsize=(8, 6))

plt.scatter(y\_test, y\_pred, color='b', label='Predicted', alpha=0.5)

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'k--', lw=1)

plt.xlabel('Actual Popularity')

plt.ylabel('Predicted Popularity')

plt.title('Actual vs Predicted Popularity')

plt.legend()

plt.grid(True)

plt.show()

**Decision Tree Regression**

from sklearn.tree import DecisionTreeRegressor

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score

import matplotlib.pyplot as plt

import numpy as np

X = data[['duration\_ms', 'danceability', 'energy', 'loudness', 'speechiness', 'acousticness', 'instrumentalness', 'liveness', 'valence', 'tempo']] # Features

y = data['popularity']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = DecisionTreeRegressor(random\_state=42)

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

predicted\_popularity = model.predict(X)

print(f"Predicted Popularity: {predicted\_popularity.mean()}")

mse = mean\_squared\_error(y\_test, y\_pred)

mae = mean\_absolute\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

mean\_actual = y\_test.mean()

mape = mean\_absolute\_error(y\_test, y\_pred) / mean\_actual \* 100

mse\_percentage = (mse / mean\_actual) \* 100

rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

print(f"Root Mean Squared Error (RMSE): {rmse}")

print(f"Mean Squared Error: {mse}")

print(f"Mean Absolute Error (MAE): {mae}")

print(f"Mean Absolute Error Percentage (MAPE): {mape:.2f}%")

print(f"Mean Squared Error Percentage: {mse\_percentage:.2f}%")

print(f"R-squared: {r2}")

plt.figure(figsize=(8, 6))

plt.scatter(y\_test, y\_pred, color='b', label='Predicted', alpha=0.5)

plt.scatter(y\_test, y\_test, color='r', label='Actual', alpha=0.5)

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'k--', lw=1)

plt.xlabel('Actual Popularity')

plt.ylabel('Predicted Popularity')

plt.title('Actual vs Predicted Popularity')

plt.legend()

plt.grid(True)

plt.show()

**KNN Regression**

from sklearn.neighbors import KNeighborsRegressor

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score

import matplotlib.pyplot as plt

X = data[['duration\_ms', 'danceability', 'energy', 'loudness', 'speechiness', 'acousticness', 'instrumentalness', 'liveness', 'valence', 'tempo']] # Features

y = data['popularity']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = KNeighborsRegressor(n\_neighbors=5)

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

predicted\_popularity = model.predict(X)

most\_popular\_song\_index = np.argmax(predicted\_popularity)

print(f"Predicted Popularity: {predicted\_popularity.mean()}")

mse = mean\_squared\_error(y\_test, y\_pred)

mae = mean\_absolute\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

mean\_actual = y\_test.mean()

mape = mean\_absolute\_error(y\_test, y\_pred) / mean\_actual \* 100

mse\_percentage = (mse / mean\_actual) \* 100

rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

print(f"Root Mean Squared Error (RMSE): {rmse}")

print(f"Mean Squared Error: {mse}")

print(f"Mean Absolute Error (MAE): {mae}")

print(f"Mean Absolute Error Percentage (MAPE): {mape:.2f}%")

print(f"Mean Squared Error Percentage: {mse\_percentage:.2f}%")

print(f"R-squared: {r2}")

plt.figure(figsize=(8, 6))

plt.scatter(y\_test, y\_pred, color='Violet', label='Predicted', alpha=0.5)

plt.scatter(y\_test, y\_test, color='Orange', label='Actual', alpha=0.5)

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'k--', lw=1)

plt.xlabel('Actual Popularity')

plt.ylabel('Predicted Popularity')

plt.title('Actual vs Predicted Popularity')

plt.legend()

plt.grid(True)

plt.show()

**MLP Regression**

from sklearn.neural\_network import MLPRegressor

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score

import matplotlib.pyplot as plt

X = data[['duration\_ms', 'danceability', 'energy', 'loudness', 'speechiness', 'acousticness', 'instrumentalness', 'liveness', 'valence', 'tempo']] # Features

y = data['popularity']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = MLPRegressor(hidden\_layer\_sizes=(100, 50), activation='relu', solver='adam', max\_iter=1000)

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

predicted\_popularity = model.predict(X)

most\_popular\_song\_index = np.argmax(predicted\_popularity)

print(f"Predicted Popularity: {predicted\_popularity.mean()}")

mse = mean\_squared\_error(y\_test, y\_pred)

mae = mean\_absolute\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

mean\_actual = y\_test.mean()

mape = mean\_absolute\_error(y\_test, y\_pred) / mean\_actual \* 100

mse\_percentage = (mse / mean\_actual) \* 100

rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

print(f"Root Mean Squared Error (RMSE): {rmse}")

print(f"Mean Squared Error: {mse}")

print(f"Mean Absolute Error (MAE): {mae}")

print(f"Mean Absolute Error Percentage (MAPE): {mape:.2f}%")

print(f"Mean Squared Error Percentage: {mse\_percentage:.2f}%")

print(f"R-squared: {r2}")

plt.figure(figsize=(8, 6))

plt.scatter(y\_test, y\_pred, color='blue', label='Predicted', alpha=0.5)

plt.scatter(y\_test, y\_test, color='purple', label='Actual', alpha=0.5)

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'k--', lw=1)

plt.xlabel('Actual Popularity')

plt.ylabel('Predicted Popularity')

plt.title('Actual vs Predicted Popularity')

plt.legend()

plt.grid(True)

plt.show()

**Random Forest Regression**

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score

import matplotlib.pyplot as plt

X = data[['duration\_ms', 'danceability', 'energy', 'loudness', 'speechiness', 'acousticness', 'instrumentalness', 'liveness', 'valence', 'tempo']]

y = data['popularity']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = RandomForestRegressor(n\_estimators=100, random\_state=42)

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

predicted\_popularity = model.predict(X)

most\_popular\_song\_index = predicted\_popularity.argmax()

print(f"Predicted Popularity: {predicted\_popularity.mean()}")

mse = mean\_squared\_error(y\_test, y\_pred)

mae = mean\_absolute\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

mean\_actual = y\_test.mean()

mape = mean\_absolute\_error(y\_test, y\_pred) / mean\_actual \* 100

mse\_percentage = (mse / mean\_actual) \* 100

rmse = mean\_squared\_error(y\_test, y\_pred, squared=False)

print(f"Root Mean Squared Error (RMSE): {rmse}")

print(f"Mean Squared Error: {mse}")

print(f"Mean Absolute Error (MAE): {mae}")

print(f"Mean Absolute Error Percentage (MAPE): {mape:.2f}%")

print(f"Mean Squared Error Percentage: {mse\_percentage:.2f}%")

print(f"R-squared: {r2}")

plt.figure(figsize=(8, 6))

plt.scatter(y\_test, y\_pred, color='Green', label='Predicted', alpha=0.5)

plt.scatter(y\_test, y\_test, color='Blue', label='Actual', alpha=0.5)

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'k--', lw=1)

plt.xlabel('Actual Popularity')

plt.ylabel('Predicted Popularity')

plt.title('Actual vs Predicted Popularity')

plt.legend()

plt.grid(True)

plt.show()

**SVM Regression**

from sklearn.svm import SVR

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score

import matplotlib.pyplot as plt

X = data[['duration\_ms', 'danceability', 'energy', 'loudness', 'speechiness', 'acousticness', 'instrumentalness', 'liveness', 'valence', 'tempo']]

y = data['popularity']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = SVR(kernel='rbf')

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

predicted\_popularity = model.predict(X)

print(f"Predicted Popularity: {predicted\_popularity.mean()}")

mse = mean\_squared\_error(y\_test, y\_pred)

mae = mean\_absolute\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

mean\_actual = y\_test.mean()

mape = mean\_absolute\_error(y\_test, y\_pred) / mean\_actual \* 100

mse\_percentage = (mse / mean\_actual) \* 100

rmse = mean\_squared\_error(y\_test, y\_pred, squared=False)

print(f"Root Mean Squared Error (RMSE): {rmse}")

print(f"Mean Squared Error: {mse}")

print(f"Mean Absolute Error (MAE): {mae}")

print(f"Mean Absolute Error Percentage (MAPE): {mape:.2f}%")

print(f"Mean Squared Error Percentage: {mse\_percentage:.2f}%")

print(f"R-squared: {r2}")

plt.figure(figsize=(8, 6))

plt.scatter(y\_test, y\_pred, color='Purple', label='Predicted', alpha=0.5)

plt.scatter(y\_test, y\_test, color='Red', label='Actual', alpha=0.5)

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'k--', lw=1)

plt.xlabel('Actual Popularity')

plt.ylabel('Predicted Popularity')

plt.title('Actual vs Predicted Popularity')

plt.legend()

plt.grid(True)

plt.show()

**Ridge Regression**

from sklearn.linear\_model import Ridge

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score

import matplotlib.pyplot as plt

X = data[['duration\_ms', 'danceability', 'energy', 'loudness', 'speechiness', 'acousticness', 'instrumentalness', 'liveness', 'valence', 'tempo']]

y = data['popularity']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = Ridge(alpha=1.0)

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

predicted\_popularity = model.predict(X)

print(f"Predicted Popularity: {predicted\_popularity.mean()}")

mse = mean\_squared\_error(y\_test, y\_pred)

mae = mean\_absolute\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

mean\_actual = y\_test.mean()

mape = mean\_absolute\_error(y\_test, y\_pred) / mean\_actual \* 100

mse\_percentage = (mse / mean\_actual) \* 100

rmse = mean\_squared\_error(y\_test, y\_pred, squared=False)

print(f"Root Mean Squared Error (RMSE): {rmse}")

print(f"Mean Squared Error: {mse}")

print(f"Mean Absolute Error (MAE): {mae}")

print(f"Mean Absolute Error Percentage (MAPE): {mape:.2f}%")

print(f"Mean Squared Error Percentage: {mse\_percentage:.2f}%")

print(f"R-squared: {r2}")

plt.figure(figsize=(8, 6))

plt.scatter(y\_test, y\_pred, color='Yellow', label='Predicted', alpha=0.5)

plt.scatter(y\_test, y\_test, color='Green', label='Actual', alpha=0.5)

plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'k--', lw=1)

plt.xlabel('Actual Popularity')

plt.ylabel('Predicted Popularity')

plt.title('Actual vs Predicted Popularity')

plt.legend()

plt.grid(True)

plt.show()