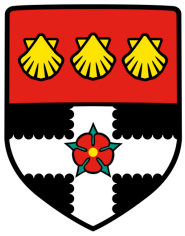

University of Reading

Department of Computer Science

Predicting *Spotify* Song Popularity using AI

Adam Smith

Supervisor: Dr. Udeni Jayasinghe

A report submitted in partial fulfilment of the requirements of
the University of Reading for the degree of
Bachelor of Science in *Computer Science*

May 30, 2025

Declaration

I, Adam Smith, of the Department of Computer Science, University of Reading, confirm that this is my own work and figures, tables, equations, code snippets, artworks, and illustrations in this report are original and have not been taken from any other person's work, except where the works of others have been explicitly acknowledged, quoted, and referenced. I understand that if failing to do so will be considered a case of plagiarism. Plagiarism is a form of academic misconduct and will be penalised accordingly.

I give consent to a copy of my report being shared with future students as an exemplar.

I give consent for my work to be made available more widely to members of UoR and public with interest in teaching, learning and research.

Adam Smith
May 30, 2025

Abstract

In the streaming era, musical popularity is increasingly shaped by algorithmic platforms and digital consumption patterns. While many existing systems rely on engagement metrics such as plays, skips, or playlist inclusion, this project investigates whether a song's internal characteristics alone can predict its popularity. Drawing from Music Information Retrieval (MIR), machine learning, and structured audio metadata, the study analyses over one million Spotify tracks released between 2000 and 2022. A comprehensive preprocessing pipeline was developed to produce clean, normalised inputs, including both raw audio-derived features and engineered metadata such as artist average popularity and genre frequency. Four regression models – Linear Regression, Random Forest, XGBoost, and a Deep Neural Network – were trained and evaluated using standard performance metrics (R^2, MAE, RMSE). Results show that internal features can produce meaningful predictions, with tree-based and neural network models outperforming linear benchmarks. A real-time interactive interface was also developed to allow users to adjust feature values and explore predicted popularity outcomes. The project highlights the potential of transparent, structure-driven modelling in music prediction, while reflecting critically on ethical issues such as representation bias and creative autonomy.

Keywords: music prediction, Spotify, machine learning, audio features, interpretability

Report's total word count:~ 14,000–15,000

Acknowledgements

I'd like to thank and dedicate this report to my Mother for her everlasting persistence and care; and to Dr. Jayasinghe for supporting me throughout this project.

Contents

List of Figures

List of Tables

List of Abbreviations

MIR	Music Information Retrieval
MAE	Mean Absolute Error
RMSE	Root Mean Squared Error
DL	Deep Learning
DNN	Deep Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
df	DataFrame
API	Application Programming Interface
KDE	Kernel Density Estimate
AI	Artificial Intelligence
RMSProp	Root Mean Squared Propagation
ReLu	Rectified Linear Unit
RF	Random Forest
XGBoost	Extreme Gradient Boosting
UI	User Interface
EDA	Exploratory Data Analysis
PII	Personally Identifiable Information
CSV	Comma Separated Values
GPU	Graphics Processing Unit
BPM	Beats Per Minute
dB	Decibels
OHE	One-Hot Encoding
NLP	Natural Language Processing

Chapter 1

Introduction

1.1 Background

In today's music landscape, streaming platforms like Spotify have transformed the mechanics of musical success. Songs are no longer promoted solely through physical sales or radio rotation, but through algorithmically curated playlists and personalised recommendations. This shift raises an important technical and cultural question: what makes a song popular in the digital age? And more specifically, can popularity be predicted from the internal structure of a song – its audio characteristics and metadata – without relying on user engagement metrics?

This project addresses that question through the lens of Music Information Retrieval (MIR), machine learning, and structured data analysis. MIR allows researchers to extract quantifiable features from audio signals, such as danceability, energy, valence, and tempo. When these are combined with contextual metadata – like genre, release year, or artist popularity – they form a rich input space for computational modelling. This work isolates those internal features to evaluate their individual and collective capacity to explain and predict song popularity.

1.2 Problem statement

In the current landscape of digital music, success is often mediated less by the sound of a song and more by the systems that promote it. Streaming platforms like Spotify rely heavily on engagement metrics—streams, skips, playlist placements—to surface tracks to listeners. These behaviours, however, may reflect visibility more than value. As a result, it becomes difficult to determine whether a song is popular because of its musical qualities, or simply because it was algorithmically favoured or attached to a highly marketable persona. This project critically examines that divide by stripping away listener data and focusing exclusively on intrinsic audio and metadata features. In doing so, it not only tests the standalone predictive power of musical content, but also explores whether music, in its purest form, still holds influence in an industry increasingly driven by commercial optics.

1.3 Aims and objectives

Aim: The overarching aim of this project is to evaluate whether internal audio and metadata features can meaningfully predict the popularity of songs, and to explore how these features influence popularity through interpretable modelling, visual analysis, and an interactive user interface.

Objectives:

- Preprocess and clean a large-scale Spotify dataset (1M tracks), ensuring that all features are normalised and categorically encoded where appropriate.
- Perform exploratory data analysis (EDA) across multiple data views (standard, pruned, temporally-aware) to uncover feature trends and correlations.
- Train and evaluate four regression models: Linear Regression (as a baseline), Random Forest, XGBoost, and a Deep Neural Network.
- Implement a real-time, interactive user interface within a Jupyter Notebook, allowing manual and automatic experimentation with feature configurations.
- Reflect on the ethical implications of such modelling, including representational bias, opacity, and the trade-offs between optimisation and creativity.

1.4 Solution approach

Data Preprocessing

Includes null handling, one-hot encoding, Min-Max scaling, and dataset filtering to produce multiple variants for analysis.

Exploratory Analysis

Uses correlation matrices, temporal slicing, and statistical visualisation to interpret key relationships and support feature selection.

Modelling and Evaluation

All four models are trained on the same engineered dataset. Model outputs are compared using R^2, MAE, RMSE, and bespoke visual diagnostics. Linear regression is used to benchmark how much performance improves through more complex methods.

User Interface

A user-controlled interface enables user experimentation with feature values and real-time predictions.

1.5 Summary of contributions and achievements

The project offers several contributions:

- A cleaned and engineered dataset of over one million songs (2000–2022), structured for high-resolution analysis.
- Validation that internal, quantifiable intrinsic audio features – including those derived from metadata – can be used to estimate popularity with meaningful accuracy.
- A suite of models that vary in complexity and interpretability, providing comparative insights into algorithm performance.

- A dynamic UI that helps users visualise how individual features affect model predictions, increasing transparency and exploratory value.
- An ethical approach to modelling that prioritises natural distribution, fairness, depth, nuance, clarity, and generalisation.

These outcomes support a wide range of use cases, from academic research in MIR to creative experimentation by artists or producers. They demonstrate that popularity prediction need not rely on social metrics and can be constructed around the structure and context of music itself.

1.6 Organisation of the report

This report is structured into seven chapters, each addressing a distinct component of the project and collectively guiding the reader through the full lifecycle of the study — from theoretical grounding to practical implementation and critical evaluation.

- Chapter 2 presents a comprehensive literature review, situating this work within the domains of Music Information Retrieval (MIR), machine learning, dataset design, predictive modelling, and ethical considerations.
- Chapter 3 outlines the project methodology, detailing all preprocessing steps, including data cleaning, feature engineering, pruning, normalisation, and the construction of analytical datasets.
- Chapter 4 presents the results of the study, including model evaluations, prediction performance, and exploratory data visualisations using structured datasets.
- Chapter 5 interprets and critiques the key findings of the project, connecting results to the original research questions and discussing implications, limitations, and alternative interpretations.
- Chapter 6 addresses ethical and interpretability concerns, particularly with regard to fairness, transparency, and the influence of external biases in popularity prediction and concludes the report.
- Chapter 7 concludes the report by summarising the project's contributions, reflecting on the core research question, and outlining directions for future development.

This structure is intended to provide a logical, transparent, and academically rigorous narrative that mirrors the chronological development of the project itself.

Chapter 2

Literature Review

2.1 Introduction and Scope of Literature

Predicting the popularity of a song has become a prominent research focus in the age of streaming, where musical consumption is shaped by a complex blend of algorithmic curation, social trends, and internal track attributes. This shift away from traditional radio or physical sales metrics has led to the emergence of large-scale digital datasets, such as those available through Spotify's public API, that allow for granular exploration of the factors underpinning a track's success.

While it is well established that popularity is influenced by external and often immeasurable elements such as artist reputation or marketing exposure, recent research has investigated the extent to which popularity can be modelled using internal song characteristics alone. Features like danceability, energy, valence, tempo, and acousticness – all of which are derived through audio signal processing – are now central to modern music information retrieval (MIR) workflows. These features form the backbone of data-driven analysis in this space, enabling researchers to move beyond subjective interpretation towards measurable correlations.

Projects such as Veritas AI's *Tuning into Trends* illustrate that machine learning models trained solely on these audio features can predict Spotify popularity scores with surprising consistency, without the need for user interaction metrics(Veritas AI, 2024). Similarly, data-driven studies have shown that combining these features with contextual information – such as genre or release year – enhances predictive stability across a wide range of track types(Joseph, 2023; Devgenius, 2023).

Finally, broader reviews of MIR research confirm the academic relevance of popularity prediction as an applied task within the field, one that intersects data science, audio engineering, and cultural analytics(Schedl et al., 2024).

2.2 Audio Feature Extraction and Music Information Retrieval (MIR)

Music Information Retrieval (MIR) refers to the field dedicated to extracting structured information from raw audio signals and metadata. In machine learning contexts, MIR provides the framework for converting songs into feature vectors that can be interpreted and processed by algorithms. These features are often engineered to reflect perceptual properties, such as rhythm, harmony, timbre, and dynamics, enabling data-driven approaches to classification, recommendation, and prediction.

Spotify's publicly accessible audio features represent a blend of temporal, spectral, and

perceptual data. Core attributes such as danceability, valence, and energy are computed using a combination of signal processing techniques and proprietary models. While Spotify does not fully disclose the computation of these values, independent studies have validated their behavioural relevance(Miu and Balteș, 2022).

Other work has applied MIR features directly to popularity modelling. In Fang et al. (2024), audio-derived features were combined with visual data in a hybrid architecture, demonstrating improved performance over models using either modality alone(Fang et al., 2024). While this project does not use visual inputs, it adopts the same principle: relying on core audio features to represent a song's internal structure.

This approach aligns with a growing trend in MIR research that favours interpretable, feature-rich representations over opaque or socially entangled models(Devgenius, 2023).

2.3 Feature Engineering and Popularity Prediction

Feature engineering in this context refers to the creation of additional variables derived from broader group-level statistics – such as artist output volume, genre saturation, or temporal trends – which may influence a song's trajectory independently of its sonic attributes.

In this project, a core strategy involves computing features such as artist average popularity, genre song count, and year average popularity. These engineered values act as contextual anchors, capturing momentum, saturation, and temporal shift respectively.

Studies such as Oladokun (2023) and TU Dublin (2021) support this approach(Joseph, 2023; TU Dublin, 2021), indicating that engineered metadata enhances interpretability, predictive strength, and fairness.

2.4 Regression and Tree-Based Models in Music Prediction

Early work often relied on regression models like linear and ridge regression for their interpretability. However, their limitations in capturing non-linear, high-dimensional interactions limit their predictive utility.

Tree-based ensemble models like Random Forest and XGBoost have since emerged as preferred alternatives. These methods handle interactions and heterogeneity effectively, and also provide importance metrics(*What Makes a Song Popular: Analysis Using Various Machine Learning Algorithms*, 2023).

This project adopts a tiered approach: Linear Regression is used as a baseline, while Random Forest and XGBoost serve as the primary predictive models. This setup balances transparency and accuracy while enabling performance benchmarking across methodologies.

2.5 Deep Learning for Music Popularity Prediction

Deep learning offers an expressive modelling framework for uncovering subtle, complex patterns in feature-rich datasets. Recent studies using dense and convolutional neural networks have outperformed traditional methods on various popularity prediction tasks(Rukavina et al., 2024; UvT, 2022).

The model proposed for this project is a feedforward neural network trained on structured inputs. While interpretability is reduced, this model complements tree-based approaches by acting as a performance benchmark.

2.6 Dataset Design and Temporal Limitations

Dataset scope plays a crucial role in generalisability. To control for inconsistencies and concept drift, the dataset used in this project is temporally constrained from 2000 to 2022. This range aligns with Spotify feature reliability and provides a stable basis for modelling(Putri et al., 2023).

The year_decimal variable adds float noise to release years, improving temporal resolution for trend analysis(Kumar et al., 2021).

2.7 UI and Model Operationalisation

A user-facing Jupyter interface was developed to support interpretability. It enables users to modify input features via sliders and view predictions in real-time. This transparency promotes critical thinking and avoids the ethical issues associated with automated optimisation or recommendation(*Social Music Discovery: An Ethical Recommendation System Based on Friends' Preferred Songs*, 2024).

The UI acts both as a pedagogical tool and a bridge to applied use, reflecting growing interest in user-controllable systems within MIR(*The Impact of Playlist Characteristics on Coherence in User-Curated Music Playlists*, 2025).

2.8 Social, Ethical, and Legal Considerations

Models in this domain risk reproducing bias and cultural homogeneity. Overrepresentation of dominant genres and artists may skew results. This project mitigates such risks through denoising strategies and interactive transparency(NHSJS, 2025; *Music as a Tool for Ethics*, 2022).

Although SHAP and LIME were not implemented, the project favours an interface-led interpretability model, aligning with best practices in AI ethics and fairness.

2.9 Critique of Existing Work and Positioning

Most existing studies rely on engagement-based metrics or classification tasks. This project diverges by:

- Avoiding user engagement signals;
- Focusing on continuous modelling;
- Introducing a real-time interface for interpretability;
- Prioritising internal features and engineered metadata.

It is positioned not as a production recommender, but as a critically reflective analysis tool.

2.10 Summary

This chapter reviewed the literature relevant to modelling song popularity using audio and metadata features. It justified a multi-model strategy using Linear Regression, tree-based methods, and deep learning within a structured and temporally coherent dataset.

The focus on interpretability, fairness, and internal features situates this project within current ethical AI trends, offering a transparent alternative to opaque commercial systems.

References

Chmiel, M., Prałat, P., Klamut, P. and Szymański, B. K. (2022), 'Analyzing and predicting success of professional musicians', *Scientific Reports* **12**(1), 21914.
URL: *https://www.nature.com/articles/s41598-022-25430-9*

Devgenius (2023), 'Mini ml project — predicting spotify songs' popularity'.
URL: *https://blog.devgenius.io/mini-ml-project-predicting-spotify-songs-popularity-part-1-ec1c906b8ff8*

Fang, Y., Wang, Y. and Jin, X. (2024), 'Predicting hit songs using audio and visual features', *MDPI Proceedings* **89**(1), 43.
URL: *https://www.mdpi.com/2673-4591/89/1/43*

Joseph, O. (2023), 'Using machine learning to predict song popularity based on their audio features', *Medium* .
URL: *https://medium.com/@oladokunjoseph2/using-machine-learning-to-predict-song-popularity-based-on-their-audio-features-7c64429af853*

Kumar, S., Chang, Y. and Liu, D. (2021), 'Modelling time-decay and timestamp uncertainty in behavioural event sequences', *Scientific Reports* **11**, Article 94229.
URL: *https://www.nature.com/articles/s41598-021-94229-1*

Miu, A. C. and Baltes, F. R. (2022), 'Music we move to: Spotify audio features and reasons for listening', *PubMed* .
URL: *https://pubmed.ncbi.nlm.nih.gov/36174020/*

Music as a Tool for Ethics (2022).
URL: *https://www.researchgate.net/publication/364130560*$_{M}usic_{a}s_{a}T ool_{f}or_{E}thics$

NHSJS (2025), 'The impact of artificial intelligence on music production: Creative potential, ethical dilemmas, and the future of the industry'.
URL: *https://nhsjs.com/2025/the-impact-of-artificial-intelligence-on-music-production-creative-potential-ethical-dilemmas-and-the-future-of-the-industry*

Putri, M. D., Santoso, H. B. and Purwandari, B. (2023), 'Predicting song popularity based on spotify's audio features: Insights from the indonesian streaming users', *Journal of Information and Telecommunication* **7**(3).
URL: *https://www.tandfonline.com/doi/full/10.1080/23270012.2023.2239824*

Rukavina, T., Zhao, C. and Li, F. (2024), 'Predicting music track popularity by convolutional neural networks on spotify features and spectrogram of audio waveform', *arXiv* .
URL: *https://arxiv.org/pdf/2505.07280*

Schedl, M., Gómez, E. and Serra, X. (2024), 'Surveying more than two decades of music information retrieval research on playlists', *ACM Computing Surveys* .
URL: *https://dl.acm.org/doi/10.1145/3688398*

Social Music Discovery: An Ethical Recommendation System Based on Friends' Preferred Songs (2024), *Multimedia Tools and Applications* .
URL: *https://link.springer.com/article/10.1007/s11042-024-19505-0*

The Impact of Playlist Characteristics on Coherence in User-Curated Music Playlists (2025), *EPJ Data Science* .
URL: *https://epjdatascience.springeropen.com/articles/10.1140/epjds/s13688-025-00531-3*

TU Dublin (2021), 'Feature engineering vs feature selection vs hyperparameter optimization in the spotify song popularity dataset'.
URL: *https://arrow.tudublin.ie/cgi/viewcontent.cgi?article=1021context=ittscicon*

UvT (2022), 'Predicting the popularity of spotify songs using a neural network'.
URL: *https://arno.uvt.nl/show.cgi?fid=171864*

Veritas AI (2024), 'Tuning into trends: Machine learning models for song popularity prediction on spotify'.
URL: *https://nhsjs.com/2024/veritas-ai-tuning-into-trends-machine-learning-models-for-song-popularity-prediction-on-spotify*

What Makes a Song Popular: Analysis Using Various Machine Learning Algorithms (2023), *Studies in Science of Science* .
URL: *https://sciencejournal.re/index.php/studies-in-science-of-science/article/view/754*

Chapter 3

Methodology

3.1 Introduction

This chapter presents a full account of the methodology implemented throughout the project, from problem formulation to experimental setup. It outlines the pipeline design, technical tools used, dataset processing stages, and ethical considerations. Each decision is grounded in reproducibility, transparency, and relevance to the aim of predicting Spotify song popularity using only internal, structured features.

The methodology can almost entirely be explained by the below flow diagram – it displays the cyclical trial-and-error nature of the methodology. Chronology can be seen by dint of the Notebook – whereas the final implications and choices that define this methodology and therefore the results are fabricated by virtue of Fig. 3.1.

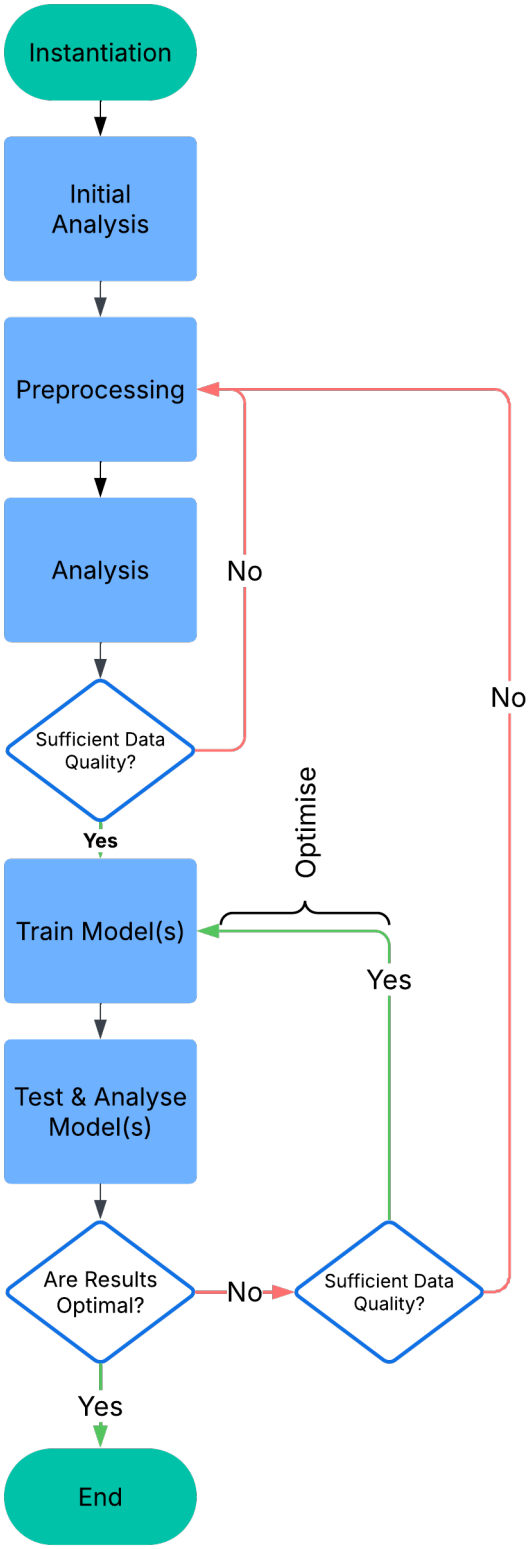

Figure 3.1: a flow chart defining the process of the Methodology.

3.1.1 Investigating the Problem

The aim of this project is to determine whether a song's popularity on Spotify can be accurately predicted using only internal features—specifically, audio-derived characteristics (such as danceability, energy, and valence) and structured metadata (including genre, release year, key, and mode). Critically, the modelling process excludes any form of user engagement data (e.g. stream counts, likes, playlist additions), ensuring that the predictions are based solely on internal attributes.

This task is framed as a supervised regression problem, with the Spotify popularity score—ranging from 0 to 100—used as a continuous target variable. This framing enables nuanced modelling of success beyond binary thresholds typical of traditional hit prediction tasks. By using a continuous score, the system can capture marginal popularity variations and provide richer insights into the drivers of commercial success.

The rationale for excluding engagement metrics stems from three main motivations:

- Interpretability: Internal features offer clear and traceable connections to musical structure, facilitating insight into what makes a song popular.
- Fairness: Engagement data often reflects feedback loops and algorithmic bias, especially in favour of well-established or high-profile artists.
- Reproducibility: Structural features are stable across time and platforms, and can be consistently extracted via APIs—unlike behaviourally driven metrics, which are volatile and platform-dependent.

To support different analytical and modelling tasks while preserving methodological rigour, three distinct versions of the dataset were constructed:

- `train_df`, `pruned_df` and `temporal_df`

These datasets were produced through a structured pipeline that begins with a unified processing stage, as defined in the Sections **1.3.1** through **1.3.5** of the accompanying notebook (see appendix). This initial pipeline comprises:

- **Data Cleaning** (*Section 1.3.1*): Removal of null entries, filtering of invalid genres and time signatures, and dropping of extraneous index columns.
- **Feature Engineering** (*Section 1.3.2*): Computation of group-level metadata including artist, genre, and year average popularity and track counts.
- **Data Shaping and Filtering** (*Sections 1.3.3 and 1.3.4*): Conversion and clipping of variables such as tempo, loudness and duration to bounded, interpretable formats. Removal of low-quality genre entries.
- **Data Separation** (*Section 1.3.5*): Segregation of the dataset into descriptive, categorical, and numerical components to improve modular processing.

Following these stages, the unified dataset branches into three specialised forms in **Notebook Section 1.3.6**, each supporting a different analytical trajectory:

- `train_df` – a fully one-hot encoded and normalised dataset used across all supervised learning tasks. This version forms the input for all predictive models described in Chapter 4.

- `pruned_df` – a filtered subset where entries too closely aligned with artist average popularity are removed, thus reducing the dominant influence of artist momentum in correlation analysis – filtering out data that is predictable. This leaves only data with more dominant and thus higher-value features.
- `temporal_df` – a time-respective variant with fractional release year (`year_decimal`) and running averages computed via `.expanding().mean().shift(1)`. This version enables temporally-aware trend analysis and reduces the risk of information leakage across time.

Each of these downstream datasets supports specific project components:

- **Correlation and Pattern Discovery** – with `pruned_df` and `temporal_df`,
- **Predictive Modelling** – with `train_df` across Linear Regression, XGBoost, Random Forest, and Deep Neural Network architectures,
- **User Interface Integration** – through model-ready structures in `train_df` and interpretability enhancements supported by the modular pipeline design.

This multi-path pipeline ensures robust foundations, transparent transformations, and reproducible structures across all facets of the analysis. It reflects both the chronological construction documented in the notebook and the modular philosophy underpinning this project's reproducibility and interpretability goals.

3.2 Tools and Technologies

All implementation was conducted in Python using a Google Colab environment. This platform was selected due to its cloud-based consistency, embedded GPU access, standard access to TensorFlow and seamless notebook interface that supports modular code development and interactive visualisation. The environment also simplifies distribution and reproducibility, as every phase of the project – from data preparation to model evaluation – occurs within a single organised, documented and executable document.

3.2.1 Libraries and Frameworks

The following libraries and tools formed the backbone of the development process:

- **NumPy** and **Pandas** were used for all numeric processing and data wrangling. Pandas dataframes enabled efficient handling of large-scale tabular data and simplified merging, filtering, and transformation steps.
- **Matplotlib** and **Seaborn** were employed for all visual analysis, including distribution plots, correlation heatmaps, and residual diagnostics. These plots were instrumental in Section 2, which guided feature selection and identified the need for normalisation and filtering.
- **Scikit-learn** was used for preprocessing, feature scaling, and classic ML models. Linear Regression, Random Forest, and evaluation metrics such as R^2 and MSE were sourced from this library, offering standardised interfaces for model training.

- **XGBoost**, a gradient boosting framework optimised for structured data, was selected for its performance and interpretability advantages. Its feature importance scores helped validate metadata signal strength.
- **TensorFlow** was used to build and train the project's deep learning model. The Sequential API enabled flexible definition of a multi-layer dense network with dropout layers for regularisation. The model was trained using RMSprop and Mean Squared Error loss to accommodate the regression framing.
- **TQDM** was employed for progress tracking during training and prediction loops, enhancing usability during iterative processes such as the artist popularity sweep.
- **ipywidgets** powered the construction of a user interface that allows users to interact with the model by adjusting audio and metadata inputs in real time. This interface is documented in Section 5 of the notebook and supports exploratory learning.

3.2.2 Programming Environment

As aforementioned, this project was developed entirely using Python 3.9+ within a Jupyter Notebook environment, hosted in Google Colab. The combination of Colab's free cloud compute resources and Python's scientific ecosystem enabled efficient manipulation of over 1 million tracks and integration of machine learning and deep learning workflows – all in a reproducible, modular context.

The notebook was organised into structured sections using markdown headers that correspond to the key phases of the data pipeline: data cleaning (1.3.1), feature engineering (1.3.2), data shaping (1.3.3), filtering (1.3.4), separation (1.3.5), and dataset branching (1.3.6) as explained in the prior 3.1.1. This structure mirrored the chronological development of the project and ensured clarity across all stages of implementation. The rest of the notebook continues chronologically, providing a diverse array of various figures, information, models and tests to be viewed throughout this report.

This layered pipeline structure ensures that foundational preprocessing is consistent across all experiments while allowing specialised forms of the data to meet targeted analytical needs. The approach prioritises transparency, reproducibility, and modularity, aligning closely with the overarching goals of the project.

Due to Google Colab's 12-hour session cap and memory limits (~12GB RAM), code was optimised to reduce redundant copies of dataframes, and models were batch-trained with small epochs during development to prevent disconnection during DNN training and to ensure that the time frame of the project is realistic. GPU acceleration was sometimes utilised via TensorFlow's built-in support to support this. The entire notebook is runnable end-to-end without dependency on external files (excluding the dataset, this must be loaded into the instance) or APIs, supporting the open science aim of this project.

3.3 Implementation

3.3.1 Data Extraction and Initial Profiling

The dataset, comprising over 1.15 million Spotify tracks from 2000–2022, was sourced via Kaggle and loaded using Pandas into a single DataFrame. Upon import, early data exploration steps were employed to validate the structure and distribution of key variables. These included type-checking all columns, confirming the range of the popularity variable (0–100), and verifying the presence of all expected feature columns: audio-derived attributes

such as danceability, energy, and valence, as well as metadata including artist_name, track_name, genre, key, and year.

Initial visualisation using distribution plots, histograms, and scatter matrices provided insight into the internal variance and potential data quality issues. For instance, it was observed that some fields like speechiness and instrumentalness had heavy right-skewed distributions, and variables like loudness and tempo included significant outliers that risked distorting model scale. Moreover, certain categories such as time_signature included clearly invalid values (e.g. 0 and 1), which informed the need for early-stage filtering. These inspection steps not only informed the cleaning logic but provided a baseline for understanding how audio and categorical features were distributed across the dataset.

3.3.2 Data Cleaning and Filtering

To prepare the dataset for any downstream analysis, a robust cleaning procedure was implemented. This began with the removal of null entries—fewer than 20 in total, primarily from artist_name and track_name—followed by the dropping of a redundant index column included in the CSV file export. Lastly, invalid time_signature rows, which constituted a small but non-negligible subset, were removed entirely (rows indicating inherently erroneous time signatures of 0 or 1).

Genre-level filtering was conducted next. A statistical audit revealed that many of the original genre labels were either sparsely populated or highly ambiguous (e.g. 'anime', 'gospel', 'podcast').

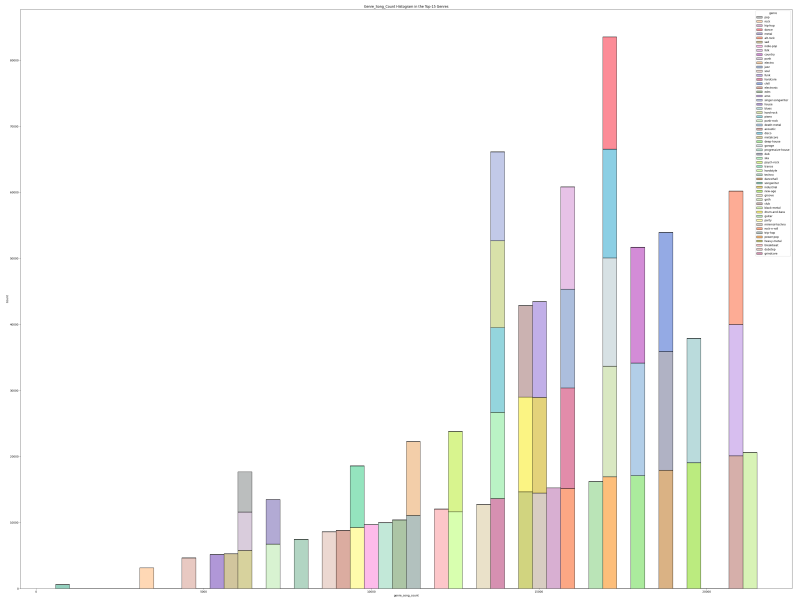

Figure 3.2: a genre-colour-coded histogram (genre colours ordered by descending average popularity) of genre_song_count showing the contrast of representation and under-representation for all genres, chiefly showing the contrast lack of representation for ambiguous, completely culture-based or niche genres.

These genres were excluded to reduce both modelling noise and cultural bias, resulting in a cleaner and more consistent feature space for genre-based correlation and predictive analysis.

Numeric variables were clipped to empirically justifiable ranges based on exploratory statistics and literature benchmarks.

=== Unnormalised Min. and Max. Values ===

	popularity	tempo	artist_avg_popularity	artist_song_count	genre_avg_popularity	genre_song_count	year_avg_popularity	year_song_count	duration_mins
count	746748.000	746748.000	746748.000	746748.000	746748.000	746748.000	746748.000	746748.000	746748.000
mean	19.716	124.152	19.678	87.685	19.766	15170.145	18.439	46899.399	4.132
std	16.167	28.783	13.617	164.140	11.106	4069.155	6.004	3386.969	1.800
min	0.000	50.000	0.000	1.000	2.699	581.000	10.770	39832.000	0.250
25%	6.000	102.190	9.124	21.000	11.478	13285.000	13.473	43492.000	3.067
50%	17.000	124.477	17.894	50.000	17.102	15753.000	16.394	46911.000	3.800
75%	30.000	140.145	28.000	100.000	25.904	17878.000	23.126	50000.000	4.826
max	94.000	220.000	85.000	2278.000	55.677	21503.000	31.115	50000.000	20.000

Figure 3.3: Summary statistics for unnormalised features in `df` including the minimum and maximum values from each. These values specifically are imperative for determining the ranges for clipping and identifying the need for normalisation.

loudness was restricted to the [–60, 0] dB range to capture meaningful musical dynamics. Similarly, tempo was trimmed to the [50, 220] BPM range, and `duration_mins` to [0, 20] minutes. Outlier tracks beyond these thresholds were rare and likely to represent noise, live albums, or metadata errors. Artist frequency variables such as `artist_song_count` were capped at 3000 (encloses all actual values - just a formal boundary), while all popularity features were clipped to the [0, 100] range.

3.3.3 Feature Engineering

To capture broader contextual signals, six key engineered features were created:

`artist_avg_popularity`, `genre_avg_popularity`, `year_avg_popularity`, and their corresponding song count metrics; `artist_song_count`, `genre_song_count` and `year_song_count`. These were computed using grouped aggregations—e.g. taking the mean popularity per artist, genre, or year—as well as counts of tracks per artist and genre.

The motivation for these features was twofold. First, they introduce global contextual knowledge about the historical performance of each categorical grouping. Second, they help model the latent platform-level popularity bias that exists for certain genres or artists, even in the absence of direct engagement metrics. These group-level signals can help proxy for influence, familiarity, or production quality, all of which are known to affect listener exposure.

A correlation matrix revealed that `artist_avg_popularity` was by far the strongest single predictor of popularity, with a Pearson correlation coefficient of approximately 0.84. While this introduces the risk of momentum-based bias, its presence is consistent with Spotify's feedback loops and ranking mechanisms. To mitigate over-reliance on this feature in interpretation, later sections introduce a pruned dataset (`pruned_df`) where high-alignment entries are excluded.

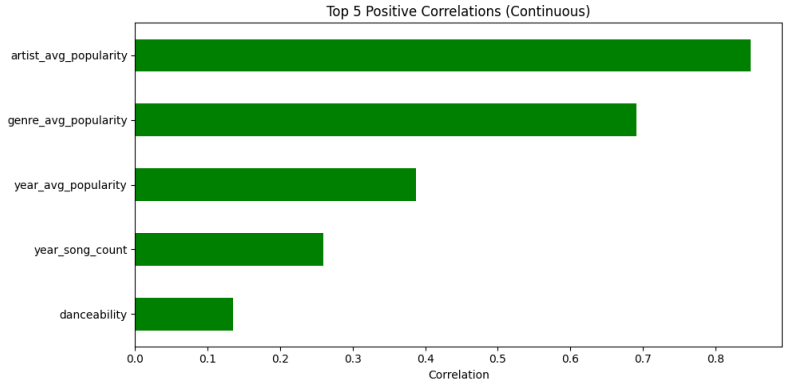

Figure 3.4: Top 5 Positive Correlation Variables from `df`.

3.3.4 Data Shaping

To facilitate modular preprocessing and ensure clear boundaries between functional components of the pipeline, the dataset was systematically separated into three distinct subsets:

- **`descriptors_df`** – contains non-numeric metadata such as `artist_name`, `track_name`, and `track_id`. These identifiers are excluded from any modelling but are retained for visualisation, UI display, and interpretation of predictions, and of course quantified during feature engineering.
- **`categorical_df`** – isolates discrete categorical features including `genre`, `year`, `key`, `mode`, and `time_signature`. These fields require separate handling via one-hot encoding and are processed independently before being reintegrated.
- **`df`** – the core working dataset that includes all continuous features and derived statistics intended for model training. This subset excludes textual or categorical fields and forms the foundation for normalisation and regression input (`train_df`).

This structured decomposition was essential for three reasons. Firstly, it reduced the likelihood of data leakage by separating identifiers from numeric inputs. Secondly, it enabled targeted preprocessing operations, such as applying one-hot encoding only to categorical fields or normalisation only to continuous features. Finally, it improved pipeline maintainability and interpretability, allowing each stage of transformation to be reasoned about in isolation.

Following this segmentation, categorical variables were processed through one-hot encoding (see Section 3.3.5), and continuous variables were scaled to a common range (see Section 3.3.6). All resulting columns were later concatenated into a single unified structure: `train_df`, which served as the standardised dataset for all supervised learning tasks from Section 3 onwards.

This shaping process reflects a deliberate design ethos prioritising clarity, modularity, and reproducibility—key principles when working with large-scale structured datasets and complex modelling workflows.

3.3.5 One-Hot Encoding (OHE)

Occuring in Section 1.3.6 of the Notebook uniquely for `train_df`, a critical step in preparing the dataset for supervised learning was the transformation of categorical variables into a fully numeric format via one-hot encoding (OHE). This encoding strategy was necessary for ensuring compatibility with machine learning algorithms, which require fixed-length, numerically

encoded inputs. The approach was tailored manually for most variables to preserve naming clarity and structural interpretability throughout the pipeline.

The encoding process was applied to four categorical variables: `key`, `mode`, `time_signature`, and `genre/year`. The treatment of each depended on its cardinality, naming convention, and interpretive role in the interface.

Manual Encoding. For `key`, `mode`, and `time_signature`, one-hot encoding was implemented manually. This decision was based on two key motivations:

1. **Controlled Naming.** Manual encoding ensured that resulting columns had clearly interpretable names (e.g., `keyOfC`, `major`) rather than ambiguous or numerical identifiers such as `key_0` or `mode_1`. This was essential for downstream tasks including feature importance analysis, UI labelling, and user-driven prediction in Section 5.

2. **Structural Transparency.** Manual construction provided full visibility into how each categorical input was mapped to numerical space. This avoided the black-box abstraction typical of automated encoders and ensured that each encoded feature retained an intuitive, musically meaningful representation.

Each row was iterated over explicitly, and for each category present, a corresponding binary flag was created and appended as a new column. This upheld the interpretability of the final dataset schema.

Automatic Encoding – In contrast, high-cardinality categorical fields such as `genre` and `year` were encoded using `OneHotEncoder` from `scikit-learn`. This was a practical decision, as both variables contained a large number of unique values that were already appropriately labelled for interpretation. Automatic encoding streamlined this step without sacrificing clarity, and allowed for rapid conversion and reintegration into the master training set.

Outcome – The combination of manual and automated one-hot encoding produced a clean, fully numeric dataset with unambiguous column names and uniform dimensionality. This hybrid strategy balanced interpretability with scalability, and ensured compatibility with all modelling workflows from linear regression to deep learning. Additionally, the naming structure directly supported the later development of the UI system (Notebook Section 5), where encoded values had to be mapped back to human-readable dropdown labels.

All encoded features were integrated into the primary modelling dataset `train_df`, which contained no raw categorical variables post-encoding. This consistent, flat structure enabled efficient training, easier debugging, and smooth deployment across the project pipeline.

3.3.6 Normalisation of Continuous Features

Normalisation was applied to all continuous variables across the final training dataset `train_df`, with the aim of standardising feature magnitudes and ensuring consistent model behaviour across different learning algorithms. The selected method was min-max scaling, which maps all feature values to the $[0, 1]$ range based on predefined minimum and maximum bounds. These bounds were manually determined from exploratory statistics and domain-appropriate clipping ranges (see Section 1.3.4 from the Notebook or 3.3.2), rather than dynamically learned from the data. This static bounding strategy ensures that future transformations remain consistent and interpretable, even when data is streamed or sampled non-randomly.

The necessity of this normalisation arises from the structure of the dataset itself. Several key features—such as `loudness`, `tempo`, `duration_mins`, and various popularity aggregates—span disparate numerical scales and contain long-tailed distributions (see Notebook Sections 2.1.1 for individual feature histogram distributions). Without scaling, features with large numeric ranges or extreme outliers could disproportionately dominate model learning, particularly in the deep learning pipeline described in Section 4 of the Notebook.

Features with high skewness:

	mean	std	min	25%	50%	75%	max	skewness	kurtosis
loudness	-8.014	4.515	-47.283	-9.798	-6.948	-5.003	0.000	-1.733	4.476
speechiness	0.085	0.084	0.022	0.037	0.053	0.092	0.967	3.050	13.126
acousticness	0.223	0.314	0.000	0.002	0.043	0.362	0.996	1.290	0.220
liveness	0.212	0.185	0.006	0.097	0.132	0.283	1.000	2.038	4.275
artist_song_count	87.685	164.140	1.000	21.000	50.000	100.000	2278.000	8.633	100.904
duration_mins	4.132	1.800	0.250	3.067	3.800	4.826	20.000	2.041	10.167

Features with high kurtosis:

	mean	std	min	25%	50%	75%	max	skewness	kurtosis
loudness	-8.014	4.515	-47.283	-9.798	-6.948	-5.003	0.000	-1.733	4.476
speechiness	0.085	0.084	0.022	0.037	0.053	0.092	0.967	3.050	13.126
liveness	0.212	0.185	0.006	0.097	0.132	0.283	1.000	2.038	4.275
artist_song_count	87.685	164.140	1.000	21.000	50.000	100.000	2278.000	8.633	100.904
duration_mins	4.132	1.800	0.250	3.067	3.800	4.826	20.000	2.041	10.167

Figure 3.5: This table displays statistics on features with high **skewness** – quantifying distributional asymmetry – and high **kurtosis**, which reflects how extreme or concentrated the tails are compared to a normal distribution for each feature. These feature statistics directly reflect why normalisation will be imperative for modelling down-the-line.

Although ensemble models like Random Forest and XGBoost are relatively insensitive to feature scaling, the choice to apply min-max scaling uniformly was made to support interoperability across all model types—linear, tree-based, and neural—as well as for consistency in UI design (see Notebook Section 5). Specifically, this allowed the front-end interface to accept bounded slider inputs for each feature that are not bound by seemingly random numbers to be clean boundaries for user-friendliness, whilst still ensuring that user-driven predictions remained within the learned scope of the data for this project.

Importantly, scaling functions are explicitly identified to allow inverse transformation after predictions. This enables all model outputs, particularly those from the neural network in Section 4.3 of the notebook, to be descaled for presentation in real-world units (e.g. popularity on a 0–100 scale). This enhances interpretability and avoids misleading results during deployment.

The following features, that are not normalised (not between 0 and 1) – were scaled using manual min-max normalisation:

- **Loudness**: Scaled from clipped bounds $[-60, 0]$ dB.
- **Tempo**: Scaled from $[50, 220]$ BPM.

- **Duration (minutes)**: Scaled from $[0, 20]$ minutes.
- **Artist Average Popularity**: Scaled from $[0, 100]$.
- **Artist Song Count**: Scaled from $[1, 3000]$.
- **Genre Average Popularity**: Scaled from $[0, 100]$.
- **Genre Song Count**: Scaled from $[0, 30, 000]$.
- **Year Average Popularity**: Scaled from $[0, 100]$.
- **Year Song Count**: Scaled from $[0, 50, 000]$.
- **Popularity (target)**: Scaled from $[0, 100]$.

These transformations ensured that all modelling occurred within a standardised, bounded numeric space. Crucially, the normalised values were retained throughout both model training and evaluation, with inverse scaling only applied for final display or export. This design choice prevents hidden leakage or misinterpretation during internal computations, while ensuring output remains meaningful to users and interpreters of the system.

3.3.7 Data Pruning

The dataset `pruned_df` is a derived subset engineered specifically to address the issue of **artist momentum** bias—a phenomenon where a song's popularity is predominantly explained by the prior success of its artist, rather than by its own musical or structural qualities. While such alignment reflects real-world dynamics on streaming platforms like Spotify, and is desirable for modelling, it poses a challenge for fair and clear visual analysis, as artist-momentum is so heavily dominant. In particular, it skews correlation measures, model interpretation, and visualisation by over-representing fame-driven popularity patterns.

The custom pruning function is derived from the logic that – if any given data point has a popularity score that is unaligned far enough from the dominant `artist_avg_popualrity` feature, then it must be explained by some other feature. By filtering using this logic, we are left with only high-value data, which is very useful in analysis - especially visually.

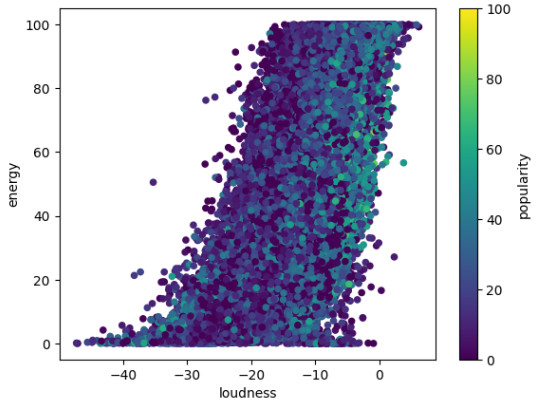

Figure 3.6: Scatter Plot of `energy` against `loudness` coloured by `popularity` (*Not Pruned*)

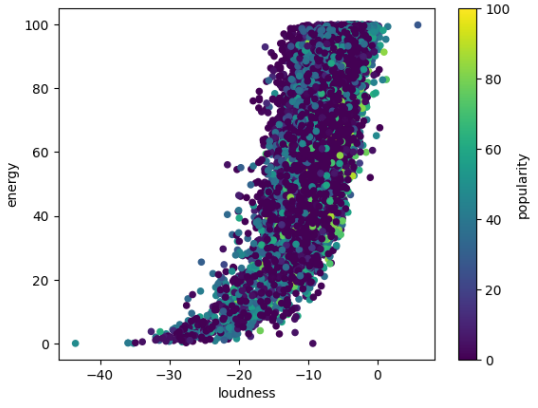

Figure 3.7: Scatter Plot of `energy` against `loudness` coloured by `popularity` (*Pruned*)

From Fig. 4.10 (not pruned) to Fig. 3.7 (pruned), we can immediately see an emergent refined shape appear through the noise – elegantly highlighting the analytical benefit of our pruning function in visualising feature relationships.

Mathematical Formulation of the Pruning Filter

To filter overly momentum-aligned entries, a custom pruning function was devised to eliminate songs whose popularity (p) aligns too closely with their artist's average popularity ($\bar{p}_a$). This method avoids removing statistical outliers; instead, it targets *inliers* that exhibit uninformative consistency with artist momentum.

Let $p_{\max}$ and $\bar{p}_{a,\max}$ be the values of popularity and artist average popularity at the most extreme joint composite point, defined as:

$$x_{\max} = \arg\max_i (p_i + \bar{p}_{a_i}) \tag{3.1}$$

This anchors a reference for the maximum observed compound popularity. For any given song, the deviation Δ_i is given by:

$$\Delta_i = |p_i - \bar{p}_{a_i}| \tag{3.2}$$

The retention condition applies two asymmetric thresholds. A song i is retained if it satisfies either:

$$\Delta_i > \frac{1}{2} \cdot \left| \frac{1}{2} \cdot \alpha \cdot p_{\max} - \left| \frac{1}{2} \cdot p_{\max} - p_i \right| \right| \tag{3.3}$$

or

$$\Delta_i > \frac{1}{2} \cdot \left| \frac{1}{2} \cdot \alpha \cdot p_{\max} - \left| \frac{1}{2} \cdot \bar{p}_{a,\max} - \bar{p}_{a_i} \right| \right| \tag{3.4}$$

The scalar parameter α is the pruning scale, generally set between 1.55 and 1.75 during visual tuning. It defines the size of a symmetrical, diamond-shaped boundary anchored at the top-right extremity of the correlation space between popularity and artist momentum. Songs falling *within* this zone are considered to exhibit excessive alignment and are thus removed. Only entries breaching one or both of these soft boundaries are retained, ensuring meaningful variance and reducing momentum-driven skew.

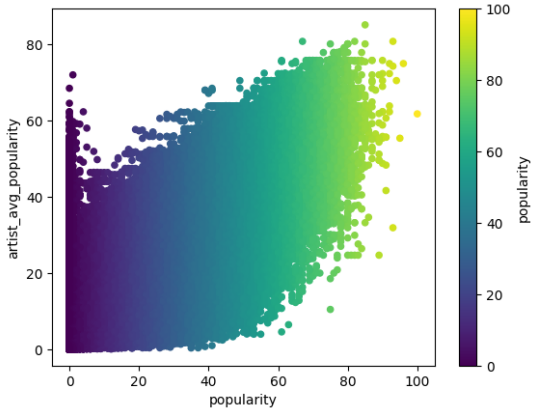

Figure 3.8: Scatter Plot of `artist_avg_popularity` against `popularity` (*Before Pruning*)

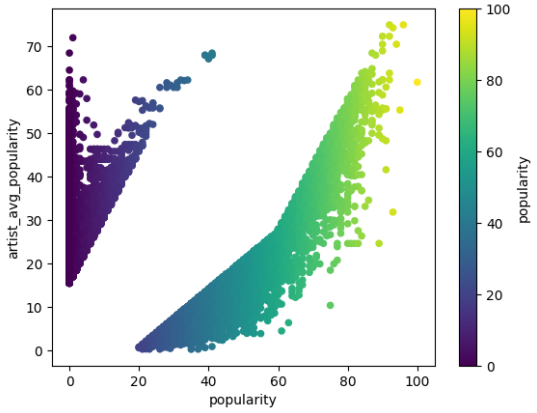

Figure 3.9: Scatter Plot of `artist_avg_popularity` against `popularity` (*After Pruning*)

This asymmetric, double-threshold structure allows the pruning to dynamically respond to both the song's and the artist's position within the broader popularity distribution, adapting based on their respective distances from global maxima. The result, `pruned_df`, serves as a filtered foundation for fairer correlation analysis in Section 2, and is excluded from modelling tasks to avoid signal distortion due to aggressive culling.

The scale parameter α is chosen empirically by iteratively visualising the correlation space and selecting a value that successfully removed the most heavily aligned entries without sacrificing diversity. It can also be done when investigating relationships visually – adjusting the scale until patterns emerge proficiently. In practice, a scale value of $scale \approx 1.55$ yielded the generally clearest boundary between biased and independent samples, going up to $scale \approx 1.75$ as seen in the scatter plots in Section 2.3.2 through 2.3.7 of the Notebook.

3.3.8 Imputing and Integrating Temporal Respect

To prevent temporal data leakage – where models inadvertently learn from future information not available at prediction time – a dedicated temporally-aware dataset variant, `temporal_df`, was constructed. This version experimentally incorporated imputed temporal structure, particularly imputing random release dates spread throughout the year of each song, to enable causal and interpretable analyses of feature behaviour over time.

Due to gaps in the raw data, release year metadata was not directly available. Missing values were thus imputed using heuristic methods, this including partial metadata cross-referencing and approximation based on related tracks. While this allowed for the derivation of year-dependent features such as `year_avg_popularity` and `genre_song_count`, the uncertainty introduced by imputation posed a risk to modelling validity.

As such, `temporal_df` was used strictly for data analysis and feature exploration – not for training or evaluating predictive models. This design choice shed light onto misrepresentation within the dataset – temporally-leaked features distort the learning process or falsely inflate performance metrics. This is because in `train_df`; for example, `artist_avg_popularity` is static, and so for e.g. a popular artist today, their whole career is represented in the dataset

by the overall average of their career – not the actual value that would've been at play at release.

Nonetheless, integrating accurate temporal structure into the training pipeline remains a compelling direction for future work. With access to reliably timestamped data, future models could capture genuine time-dependent trends and shifts in musical popularity. This would allow for more causally grounded predictions, improved generalisability across time periods, and the potential for forecasting emerging music trends far more accurately in a temporally consistent framework.

Temporal Imputation Process

To simulate realistic release patterns and avoid uniform clustering within years, each track was assigned a pseudo-random decimal component to its release year. This was achieved by drawing a value from a uniform distribution (essentially each decimal component is a factor of $\frac{1}{365}$ to roughly represent the days of the year) across the calendar year and adding it to the integer year label. The result, stored in the `year_decimal` column, approximated a continuous temporal axis suitable for expanding-window analysis.

To compute time-respecting historical aggregates, a custom `calculate_running_average()` function was implemented. This function produced expanding means and song counts over time, using the `year_decimal` value to order tracks chronologically. Crucially, the current song was excluded from all calculations via a `.shift(1)` operation, ensuring that only past information was used.

The function was applied globally for complete running values, and then separately within `artist`, `genre`, and `year` groups. For each context, it computed:

- *<context>*`_avg_popularity` – the mean popularity of earlier songs within that group
- *<context>*`_song_count` – the number of previous songs released by the artist/genre/year

For example, the `artist_avg_popularity` feature represented the average popularity of a given artist's previous tracks at the time of a new release. Initial values in each group (i.e., first song per artist or genre) were explicitly set to zero to avoid NaNs and maintain logical consistency.

This process provided a strong foundation for temporally consistent correlation analysis while strictly avoiding leakage from future data.

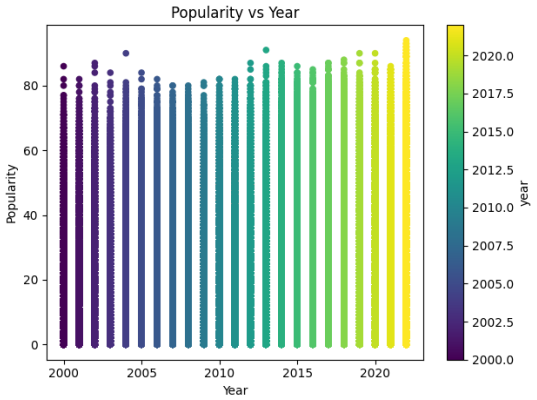

Figure 3.10: Scatter Plot of `year_avg_popularity` against `popularity` (*Before Temporal Imputation*)

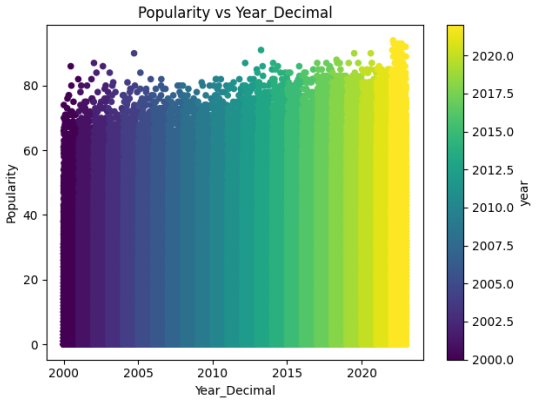

Figure 3.11: Scatter Plot of `year_avg_popularity` against `popularity` (*After Temporal Imputation*)

3.3.9 Experiments Design and Setup

This section outlines the experiments conducted to validate the performance and interpretability of the models used in this project. These experiments were designed not only to measure statistical accuracy but also to understand the human relevance and potential for real-world applicability of internal-feature-based popularity prediction.

Model Setup and Validation Strategy

The central machine learning task was framed as a supervised regression problem. The input space consisted of fully encoded, normalised features, while the target variable was Spotify's popularity metric, scaled from 0 to 1 for training purposes. The `train_df` dataset was used for all training and testing, having been prepared for such an advent.

Each model was evaluated using an 80:20 train–test split, stratified to ensure a similar popularity distribution across subsets. For interpretability, evaluation was conducted not only via numeric metrics but also through visual diagnostics such as scatter plots and residual distributions.

Four different regression models were implemented:

- Linear Regression served as a transparent baseline.
- XGBoost Regressor added regularisation first and boosted learning capacity.
- Random Forest (RF) Regressor captured non-linear feature interactions with efficacy, displaying the effectiveness of a tree-based approach.
- Deep Neural Network (DNN) – implemented via the *Keras* library with ReLU activation, dropouts (0.2), batch regularisations, one-hot and continuous separated inputs and RMSprop optimiser – provided a more expressive, learnable model.

The structural choices for the DNN were made through iterative manual performance metric testing.

Model complexity, architecture and hyperparameters were selected through literature and/or manual tuning. For Tree-based models, the number of estimators was tested between 30 and 500 depending on computation time for the model (Random Forest with 30-60 estimators is

extremely slow compared to e.g. XGBoost easily handling 500). Tuning for these models was done with limitation unfortunately due to the limited timeframe of this project – however they still serve as suitable and sufficient models for analysis. The DNN was evaluated extensively under different dropout rates and layer configurations. Results across model classes were benchmarked against each other using standard regression metrics.

Performance Metrics and Interpretability

Three primary evaluation metrics were used:

- **R^2 Score:** Measures the proportion of variance explained.
- **RMSE (Root Mean Squared Error):** Captures the magnitude of prediction error.
- **Custom Proportional Accuracy:** The mean percentage closeness between each prediction and its actual value, computed as the smaller value divided by the larger, scaled to $[0, 100]$.

The choice of metrics reflects both statistical and perceptual concerns. While an R^2 score indicates goodness-of-fit, RMSE helps identify models that minimise impactful errors. Proportional accuracy offers a human-readable benchmark for how often predictions feel 'close enough' to reality.

Scatter plots of predicted vs. actual values showed clustered diagonals for the best-performing models, with systematic underprediction for high-popularity tracks in linear models and outliers across all models. Residuals were plotted by popularity decile to detect model drift, revealing slight overprediction bias in the lower quartile.

Artist Average Popularity Sweep Test

As part of the model testing process, an experimental sweep was conducted to evaluate how the model's predictions scale in response to changes in artist-level metadata, specifically `artist_avg_popularity`. This feature is known to be highly predictive but can also obscure the role of intrinsic audio features. The goal of this test was to simulate controlled conditions and assess the model's responsiveness to shifts in a single influential variable. We would expect that predictions should grow as `artist_avg_popularity` increases - this will validate the model essentially.

In this setup, random tracks are sampled from the training data for each value of `artist_avg_popularity` ranging from 0 to 100 in 0.1 increments. For each iteration, all features were held as they were, while the `artist_avg_popularity` input was overwritten with the current value in the sweep. Predictions were then made using the trained deep learning model, and the output popularity score was recorded.

This process generated a structured prediction curve for each sampled song, allowing for a visual and statistical analysis of how popularity scores changed with increasing artist momentum. The resulting sweep exhibited a clear, nearly monotonic upward trend, validating the model's strong sensitivity to this metadata feature. At lower values of `artist_avg_popularity`, predictions were more volatile and less stable, reflecting the wider variability in baseline song quality among less popular artists.

The test served as a real-world simulation of artist context influence and confirmed the importance of accounting for metadata when interpreting predictions. While the sweep is not used for training or optimisation, it remains a key diagnostic tool for examining the inner workings of the model and verifying the realism of its output dynamics.

3.3.10 User Interface and Real-Time Prediction Integration

To facilitate practical engagement with the model, an interactive user interface (UI) was developed within the project notebook. The UI serves two primary purposes: (1) to allow users to simulate custom song profiles and obtain immediate popularity predictions, and (2) to illustrate how internal feature configurations translate into model outcomes in real time.

The interface is built using Jupyter widgets and linked directly to the deployed deep learning model. An already trained model can be imported and wrapped to support repeated real-time predicting with extremely low latency. Once initialised, the model accepts structured input arrays of shape $(1, 114)$, matching the one-hot encoded and scaled schema of the training set.

Users interact with the system through a combination of sliders, dropdowns, and formatted inputs. The UI is logically segmented into the following elements:

- **Continuous Feature Sliders:** Normalised inputs for features such as `danceability`, `energy`, and `valence`, msotly ranging from 0 to 1. A few exceptions such as `loudness` or `tempo`'s slider are appropriately set to the original, unnormalised, clipped ranges from Section 1 of the notebook, so for `loudness` the range is `[-60, 0]`. These sliders allow users to experiment with hypothetical values within valid bounds.
- **Artist Metadata Inputs:** `artist_avg_popularity` (range: 0–100) and `artist_song_count` (range: integer inputs from 1 to 3000) are handled separately for display clarity - these features are the most dominant, so they get a separate central placement in the UI.
- **Categorical Dropdowns:** Formatted selectors for `genre`, `key`, `mode`, `time_signature`, and `release year`. These are internally mapped to one-hot encoded vectors based on the training schema.

After configuration, the user triggers the `Predict` button, which constructs the full input vector by assembling scaled continuous values, encoded categorical selections, and injected metadata. This vector is passed to the model, and the output—a predicted popularity score between 0 and 1—is immediately displayed, rescaled back to the 0–100 range for human readability.

The UI system reflects a clear design ethos prioritising transparency and user control. All transformations, mappings, and predictions are fully traceable, and inverse scaling functions are applied for output clarity. This ensures interpretability and avoids the black-box feel often associated with predictive systems.

Importantly, the UI is not used for model training or optimisation; its role is purely demonstrative. It provides potential users—ranging from students to music producers—with a sandbox environment to explore how changes in audio and metadata influence popularity predictions. By allowing direct manipulation of inputs and instantaneous feedback, the interface fosters interpretive learning and encourages critical engagement with the modelling process.

3.4 Ethical, Legal, and Social Considerations

3.4.1 Data Ethics and Licensing

The dataset used was sourced from the Kaggle-hosted Spotify Million Tracks dataset, itself derived from Spotify's API. All data points are publicly accessible and include no personally identifiable information (PII), no user engagement signals (e.g. streams or likes), and no sensitive demographic attributes. The data is structured around track-level and artist-level features, all of which relate to musical content or metadata.

Due to its non-sensitive nature, the project did not require formal ethical approval under university regulations. However, we still adhered to open data licensing constraints, credited the original dataset source, and followed UK GDPR guidelines related to public dataset use for research purposes. All derived datasets and scripts were stored with appropriate documentation to ensure auditability.

3.4.2 Consent and Representation

As no human subjects or user-generated data were used, traditional informed consent procedures were not applicable. Nevertheless, the project recognised the potential reputational implications of modelling artistic content. No individual artists or tracks were explicitly evaluated or ranked in isolation. Where aggregate statistics (e.g. top 30 genres or artists) were presented, they were drawn from public metadata and used only to highlight model trends.

The UI is explicitly framed as an exploratory tool. All results are anonymised at the artist level (no artist or track names included), and feature profiles were randomly generated or taken from internal samples. This avoided personalisation and ensured that outputs could not be traced to specific real-world individuals.

3.4.3 Algorithmic Fairness and Structural Bias

A central concern in music modelling is that data reflects—not removes—platform-level inequalities. Numerous studies (e.g. (Chmiel et al., 2022)) have shown that Spotify's editorial curation and ranking systems favour specific genres (Western pop, hip-hop), languages (primarily English), and artist types (those with label backing). Such structural bias can be embedded in any downstream model unless proactively addressed.

This project adopted several bias mitigation strategies:

- **Genre filtering:** Niche and underrepresented genres were excluded when they could not provide meaningful statistical coverage.
- **Momentum pruning:** The `pruned_df` removed rows where the popularity score was almost entirely explained by `artist_avg_popularity`, aiming to decouple predictive signals from reputation.
- **Temporal integrity:** The `temporal_df` was constructed using rolling statistics that only included prior data, ensuring no future leakage into engineered features.

Despite these efforts, it is acknowledged that the model still learns from platform-level data that reflects existing inequalities. The aim was not to remove bias entirely—an impossible task—but to surface and manage it transparently.

3.4.4 Interpretability and Accountability

The models used in this project, especially Random Forest and Linear Regression, were chosen for their interpretability. Feature importances were extracted and analysed, and correlation heatmaps helped reveal which variables the models relied on most. This transparency is especially important when modelling cultural products like music, where implicit assumptions can lead to controversial inferences.

3.4.5 Societal Impact and Creative Ethics

One of the most important dimensions of this work is its potential impact on creative practice. Models that predict or optimise musical features for popularity could theoretically be used by producers to "game" recommendation systems or suppress artistic risk. This concern echoes broader debates in AI ethics regarding automation in the arts, from generative image tools to AI lyric generation.

To counteract this risk, the project explicitly avoided any prescriptive framing.

Furthermore, by keeping the modelling pipeline open-source, the report encourages critique and adaptation. It invites further work on bias auditing, decolonised music data, and genre-specific model training, rather than suggesting a single universal approach.

3.4.6 Compliance and Integrity

The project complies with institutional policies outlined in the academic handbook:

- *All work is original, individually submitted, and adheres to the assessment code of conduct.*
- *Any GenAI tools were used strictly for ideation, editing, or code structure—never to generate written report content.*

A Statement of Authorship has been submitted, and all programming assets are transparently documented in the accompanying Jupyter notebook. No commercial or production deployment has taken place, and the project remains within the scope of academic inquiry.

Summary

Ethical diligence in this project has been proactive rather than reactive. Bias was not merely acknowledged—it was actively filtered, structured, and visualised. Interpretability was not an afterthought—it was a central design principle. And while the system inevitably reflects the platform and data it learns from, it does so in a way that encourages human reflection rather than automation.

In Chapter 5, the broader implications of this approach are discussed in the context of fairness, responsibility, and future research in musical AI systems.

Chapter 4

Results

4.1 Introduction

This chapter presents the findings obtained using the methodological pipeline established in Chapter 3. It follows the modelling chronology from correlation analysis through model evaluation and feature sensitivity, ending with user interface performance. Each section is structured to reflect both statistical rigour and interpretability, ensuring relevance to the project's aim of understanding and predicting song popularity from internal features.

4.2 Correlation Analysis and Feature Relationships

Understanding how internal features relate to popularity is a critical first step before any modelling. This section explores several correlation perspectives, derived from `pruned_df` and `temporal_df`, which were specifically constructed to reduce artist bias and control for temporal leakage.

4.2.1 Continuous Feature Correlations

Initial correlation analysis was performed on `df`, the main dataset containing all tracks. This includes songs strongly influenced by artist momentum, but it provides a full comprehensive view of how generalised internal features relate to popularity.

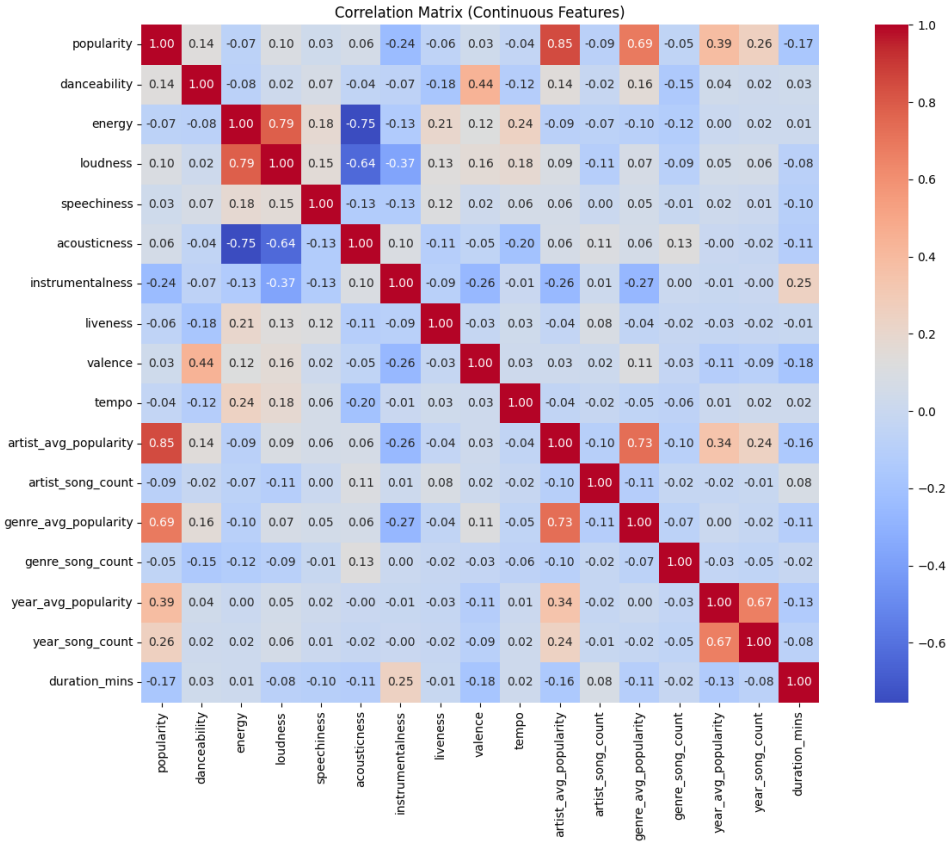

Figure 4.1: Correlation heatmap of continuous variables from `df`.

The correlation matrix shows:

- `artist_avg_popularity` has the highest correlation with the target ($r \sim 0.85$), reinforcing the known influence of artist reputation.
- `genre_avg_popularity` ($r \sim 0.69$) and `year_avg_popularity` ($r \sim 0.61$) also show strong correlations, confirming the predictive value of contextual metadata.
- Among the audio features, `loudness` ($r \sim 0.1$) and `danceability` ($r \sim 0.14$) were most positively associated with popularity, echoing studies that link upbeat characteristics with commercial success (Chmiel et al., 2022).

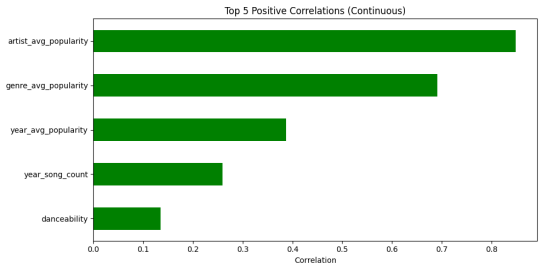

Figure 4.2: Top 5 continuous features with the strongest positive correlation to popularity.

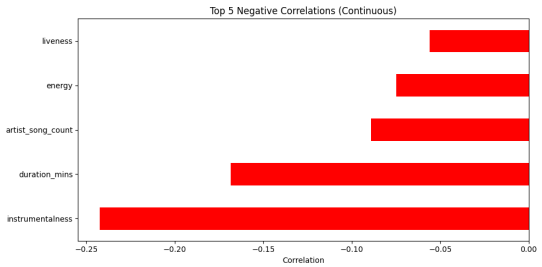

Figure 4.3: Top 5 continuous features with the strongest negative correlation to popularity.

Features like `Speechiness` and `acousticness` showed weak correlations ($r < 0.1$), which is consistent with their presence in niche or non-mainstream genres, while some such as `instrumentalness` display negative scores of ~-0.24.

4.2.2 Categorical Feature Relationships

A one-hot encoded correlation analysis was conducted for categorical variables such as genre, key, and year.

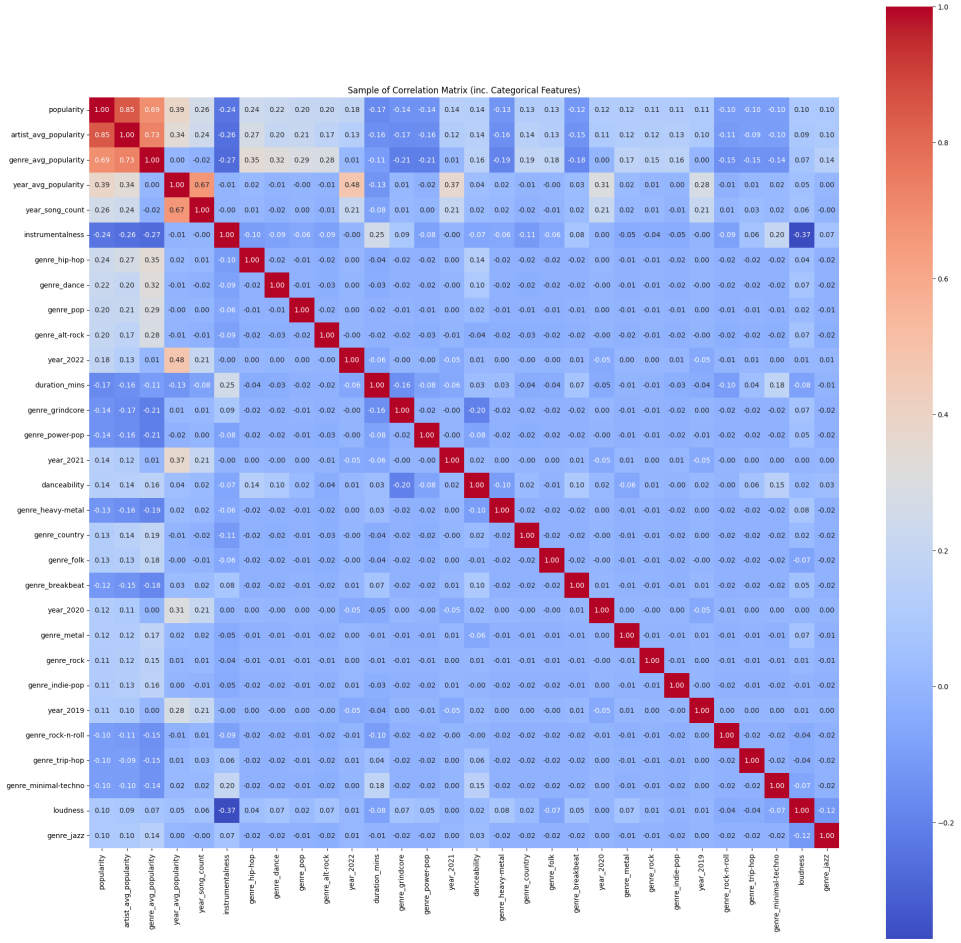

Figure 4.4: One-hot encoded correlation matrix of categorical features such as genre, key, and year against popularity.

The results highlight that:

- Mainstream genres like `dance`, `pop` and `alt-rock` positively correlate with popularity.
- Genres like `classical`, `comedy`, or `gospel` were associated with lower popularity scores.
- Temporal categories (especially 2016–2019) showed elevated average popularity, likely due to Spotify's shifting curation and listener habits.

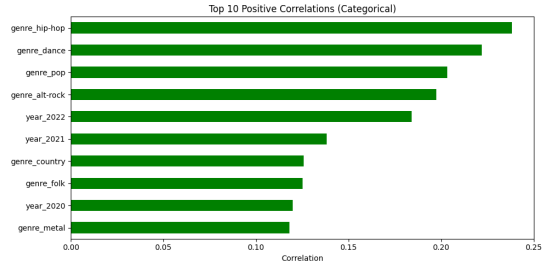

Figure 4.5: Top 5 categorical features with the strongest positive correlation to popularity.

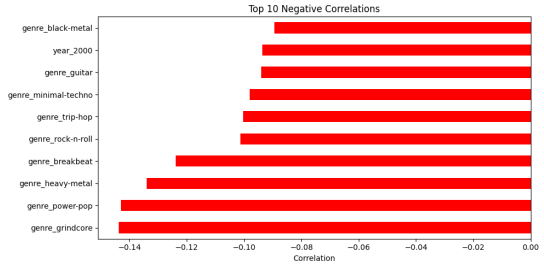

Figure 4.6: Top 5 categorical features with the strongest negative correlation to popularity.

These findings supported the importance of explicit one-hot encoding and informed feature selection for downstream modelling.

4.2.3 Temporal Correlations and Leak-Awareness

Using `temporal_df`, all engineered features were recalculated with `.expanding().mean().shift(1)` to ensure that no future information leaks into the model.

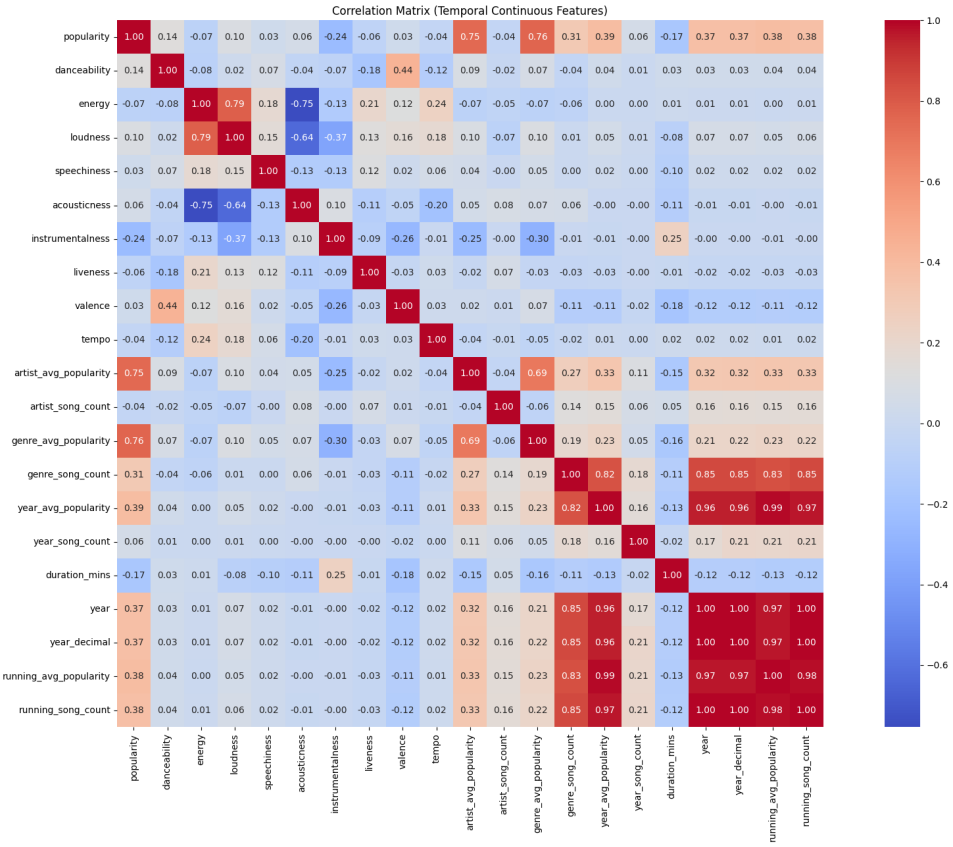

Figure 4.7: One-hot encoded correlation matrix of temporally-aware features such as genre, key, and year against popularity.

This adjustment led to notable shifts:

- `artist_avg_popularity` dropped significantly in correlation from r ~ 0.85 to 0.75, indicating prior leakage in temporally naive implementations.
- `genre_avg_popularity` rose to become the most stable predictor under temporal constraints (r ~0.76), followed closely by `year_avg_popularity` (see Fig. 4.8 & Fig. 4.9 below:

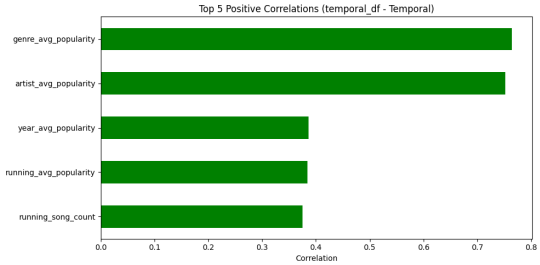

Figure 4.8: Top 5 temporally-aware features with the strongest positive correlation to popularity.

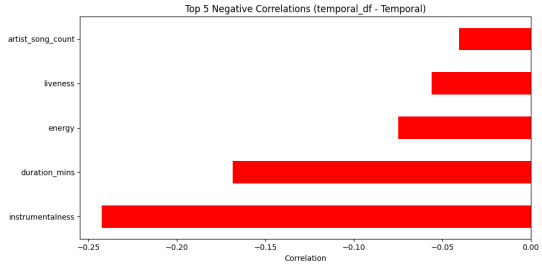

Figure 4.9: Top 5 temporally-aware features with the strongest negative correlation to popularity.

This confirms that genre and cultural timing—rather than artist legacy—carry the most transferable predictive power when time-awareness is enforced.

Summary: This analysis confirms that structured metadata, especially year and genre-level aggregates, are critical drivers of popularity. The influence of artist prestige, while initially dominant, diminishes extensively under causal-safe analysis. These insights guide the modelling priorities and validate the design of engineered features for fairness and predictive reliability.

4.2.4 Exploratory Multivariate Feature Relationship Analysis

The following figures present a set of exploratory scatter plots examining the joint relationship between two audio features (Energy and Loudness) and their combined association with popularity. These plots help reveal how musical intensity and volume dynamics correlate with success, and how these relationships differ depending on artist prominence or data pruning.

The first row compares the unfiltered dataset against the pruned version, which removes outliers and tracks with extreme artist dominance, showing a clear cut shape. The subsequent figures apply increasingly selective filters: isolating high-popularity songs (Figure 4.12) indicating a refined loudness-energy centroid of extremely popular high-intrinsic-audio-quality songs and tracks with high or low `artist_avg_popularity` (Figures 4.13–4.15). These visualisations illustrate how popularity trends can shift depending on artist influence and dataset conditioning - and reveals that there is clear feature relationship patterns amongst varying-performant songs.

A full set of additional multivariate relationships — including Danceability, Valence, Tempo, Speechiness, Acousticness, Liveness, and Instrumentalness — are provided in the second Appendix, offering a broader picture of bivariate feature dynamics across the dataset.

Note that this is just an example – displaying the exploratory power of `pruned_df`.

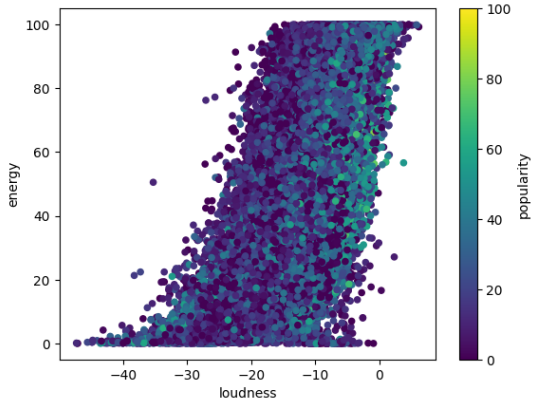

Figure 4.10: Scatter Plot of Energy, Loudness and Popularity.

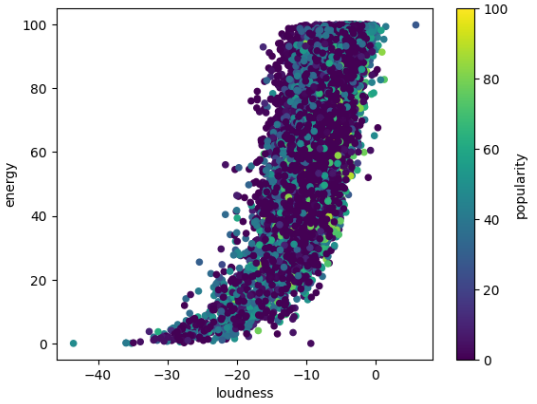

Figure 4.11: Scatter Plot of Energy, Loudness and Popularity (Pruned).

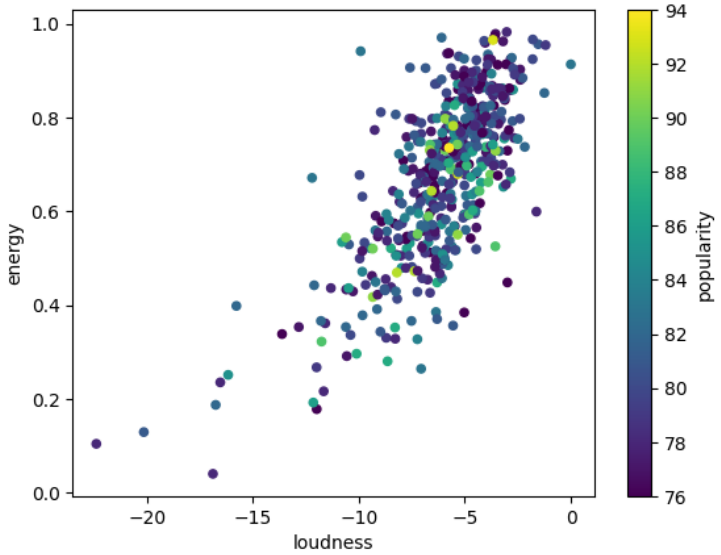

Figure 4.12: Scatter Plot of Energy, Loudness and Popularity Filtered where $popularity > \tilde{}70$ (Pruned).

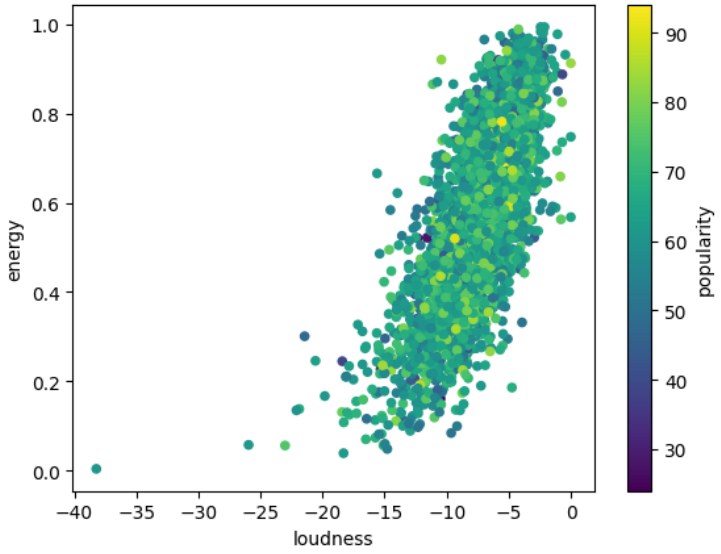

Figure 4.13: Scatter Plot of Energy, Loudness and Popularity Filtered where $artist\_avg\_popularity > \tilde{}60$ (Not Pruned).

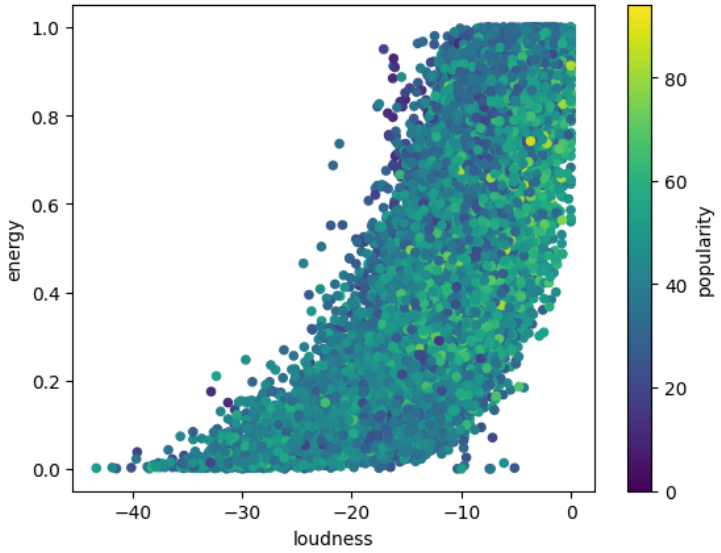

Figure 4.14: Scatter Plot of Energy, Loudness and Popularity Filtered where $artist\_avg\_popularity > \tilde{}30$ (Not Pruned).

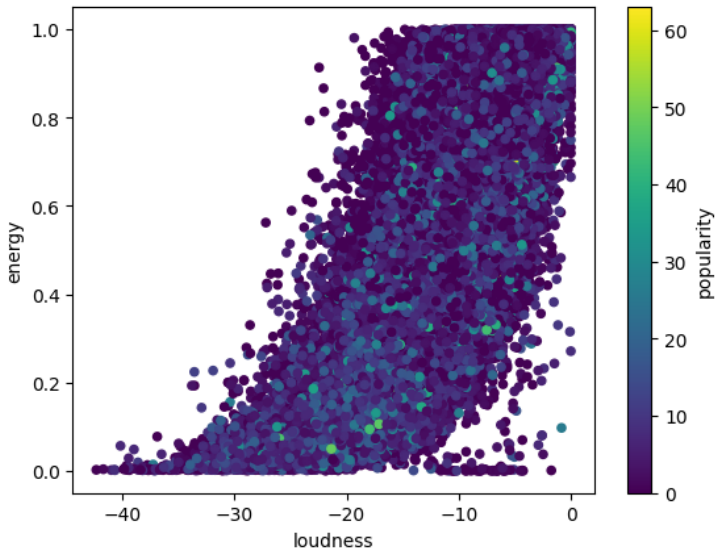

Figure 4.15: Scatter Plot of Energy, Loudness and Popularity Filtered where $artist\_avg\_popularity < \tilde{}10$ (Not Pruned).

4.3 Baseline Model: Linear Regression

To establish a transparent performance benchmark, a linear regression model was trained using the fully preprocessed and normalised `train_df`. This model serves as a baseline due to its simplicity and interpretability, allowing a first assessment of how much popularity variance can be explained linearly by internal features.

4.3.1 Performance Overview

The model achieved an R^2 score of 76.2%, suggesting that more than three-quarters of the variability in popularity can be explained by a simple linear combination of inputs. However, this performance was later exceeded by ensemble and deep learning models.

A custom accuracy metric was also applied, comparing predicted and actual popularity values proportionally. The model achieved an average custom accuracy of 63.02%, indicating moderate real-world reliability but notable imprecision at individual prediction level.

	Coefficient(β)
danceability	0.015
energy	-0.005
loudness	0.066
speechiness	-0.040
acousticness	0.009
...	...
year_2018	0.024
year_2019	0.029
year_2020	0.036
year_2021	0.041
year_2022	0.068

114 rows × 1 columns

Figure 4.16: A table of Coefficients indicating quantified feature dominance

Coefficient Interpretation

One advantage of linear regression is coefficient transparency. Each feature's weight offers direct insight into its influence:

- Strong positive coefficients were observed for `artist_avg_popularity`, `genre_avg_popularity`, and `danceability`, mirroring earlier correlation patterns.
- Negative weights were associated with niche genres and non-mainstream characteristics (e.g. `gospel`, `comedy`).

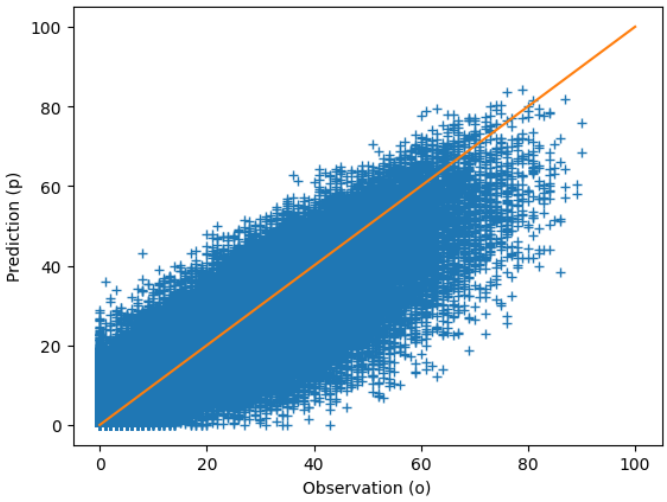

Figure 4.17: A plot of observed values against predicted values

This interpretability reinforces the pipeline's analytical goals and confirms that the signal captured by engineered metadata is substantial even without complex modelling.

Prediction Behaviour and Limitations

Despite its strengths in transparency, linear regression showed limited expressive power:

- A scatter plot of predicted vs. actual values [Figure 4.17 showing the deviation between predicted and actual values] shows substantial deviation from the diagonal, particularly at low and high extremes.
- The model systematically underpredicts high popularity songs and overpredicts low ones, flattening the response space.
- It assumes additive independence between features, which fails to model key interactions (e.g. `energy & valence`).

4.3.2 Tree-Based Models: XGBoost and Random Forest

To address the limitations of linear regression and better capture the nonlinear structure of the data, two tree-based ensemble methods were implemented: XGBoost and Random Forest. Both were trained on the full `train_df` dataset using the same input features, ensuring comparability. These models are particularly well-suited for structured datasets and offer interpretability through feature importance rankings.

4.4 XGBoost

XGBoost (Extreme Gradient Boosting) is a sequential ensemble model that builds decision trees by minimising residual error. It is known for its robustness, speed, and ability to model complex interactions.

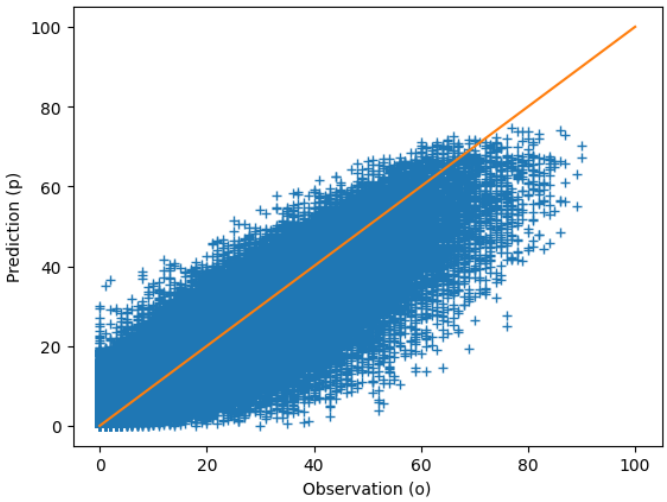

Figure 4.18: A plot of observed values against predicted values

Performance Overview –

The XGBoost model achieved an R^2 score of 79.76% and a custom accuracy of 64.86%, significantly improving on the linear baseline. This demonstrates its capacity to capture threshold effects and nonlinear combinations of internal features.

Feature Importances –

The top five features in XGBoost, measured by total gain across splits, were:

- `artist_avg_popularity`, `genre_avg_popularity` and `year_avg_popularity`
- `danceability` and `valence`

These closely align with prior correlation results and reaffirm the structural value of engineered metadata.

	Feature Importance
danceability	0.002
energy	0.001
loudness	0.002
speechiness	0.002
acousticness	0.001
...	...
year_2018	0.001
year_2019	0.001
year_2020	0.001
year_2021	0.003
year_2022	0.014

114 rows × 1 columns

Figure 4.19: A table of Coefficients indicating quantified feature dominance

Model Behaviour

XGBoost performed robustly in mid-range popularity predictions but showed slight underestimation for highly popular songs. Residual plots revealed tighter distribution than linear regression, though still with some skew. Hyperparameter tuning was kept conservative to avoid overfitting and maximise interpretability, and within the timeframe of the project.

4.5 Random Forest

Random Forest is a parallelised ensemble method that aggregates predictions across multiple independently trained trees. Each tree is fit on a random subset of the data and features, which reduces variance and improves generalisation.

Performance Overview –

Random Forest was the top-performing model, achieving an R^2 score of 81.91% and a custom accuracy of 66.21%. These metrics indicate strong predictive power and consistency across the input space.

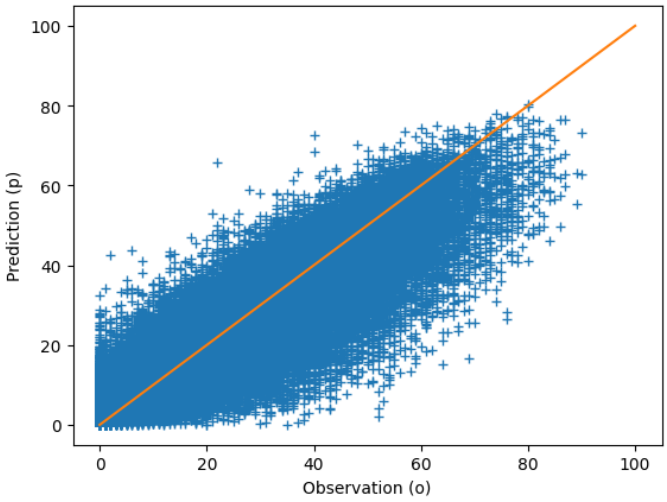

Figure 4.20: A plot of observed values against predicted values

Feature Importances

Feature rankings were nearly identical to XGBoost, led by `artist_avg_popularity`, `genre_avg_popularity`, and `year_avg_popularity`. The Random Forest rankings were also more stable due to the bagging structure.

	Feature Importance
danceability	0.015
energy	0.012
loudness	0.015
speechiness	0.015
acousticness	0.014
...	...
year_2018	0.000
year_2019	0.000
year_2020	0.000
year_2021	0.000
year_2022	0.001

114 rows × 1 columns

Figure 4.21: A table of feature importances indicating quantified feature dominance

Summary: The analysis showed that Random Forest predictions were better aligned with actual popularity across the full spectrum. Unlike XGBoost and Linear Regression, the Random Forest showed minimal skew, especially at the tails. Both tree-based models outperformed the linear baseline, with Random Forest delivering the strongest overall performance. Their success validates the design of the feature set, particularly the engineered metadata.

4.6 Deep Learning Results

To explore the benefits of a flexible, non-linear predictive framework, a deep learning model was implemented using TensorFlow's Functional API. This architecture was designed to accommodate the structural differences between one-hot encoded categorical features and continuous numerical features, enabling distinct pathways for each type before merging for final prediction.

Model Architecture and Training

The model begins with a single input layer of 114 features, corresponding to all preprocessed variables. Using `Lambda` layers, the input was split into two parts: a 16-dimensional one-hot encoded vector and a 97-dimensional continuous feature vector. These were processed separately through dedicated branches before being merged.

The one-hot branch consisted of a single dense layer of 128 units, followed by batch normalisation, a ReLU activation, and dropout (rate $= 0.1$). The continuous branch passed through three dense layers of 256, 128, and 64 units respectively, each followed by batch normalisation, ReLU activation, and dropout layers (rate $= 0.2$). The outputs of both branches were concatenated and passed through an additional dense layer of 64 units with ReLU activation and dropout, before reaching a final output neuron with linear activation to predict the normalised popularity score.

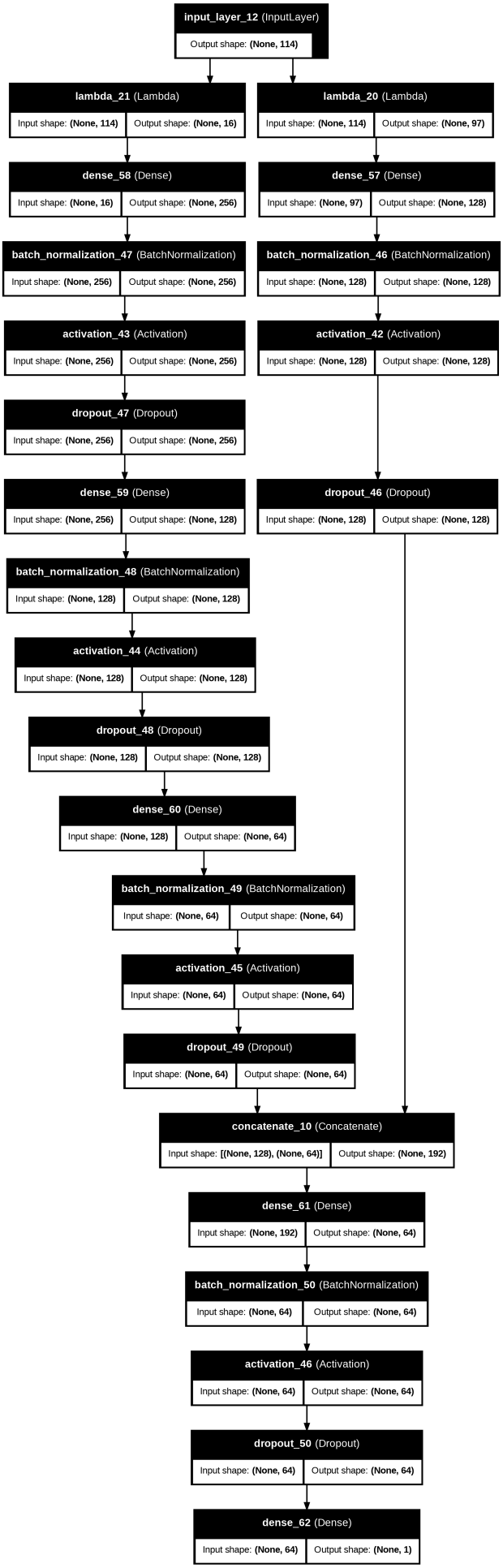

Figure 4.22: A diagram of the DNN Architecture *(continuous to the left)*

Training was conducted using the RMSprop optimiser and the mean squared error (MSE) loss function. The model was trained for 30 epochs using a batch size of 96 and a 20% validation split. Training loss stabilised at approximately ~0.0053 and test loss at ~0.0059, suggesting strong convergence and generalisation.

This architecture, by explicitly modelling categorical and continuous features through tailored sub-networks, provided a deeper interpretive structure while maintaining a high level of predictive accuracy. It also better aligned with the mixed-type feature engineering present in earlier preprocessing stages.

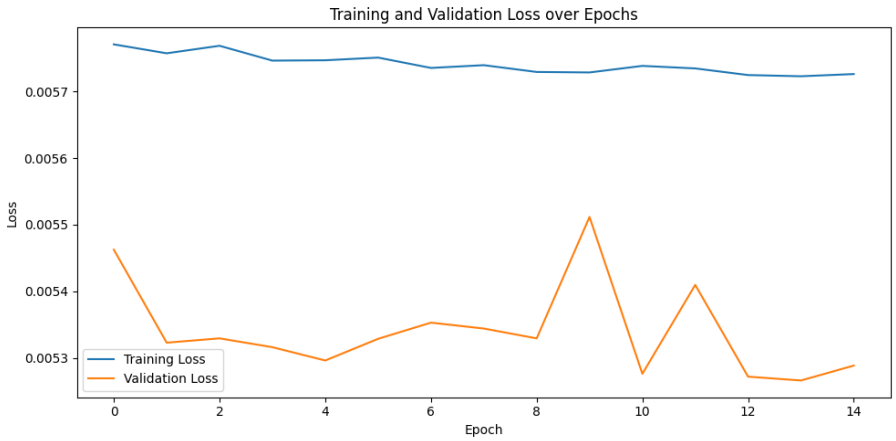

Figure 4.23: A plot of Training and Validation MSE loss during Model Training (15 epochs)

Performance Overview

The final model achieved an R^2 score of ~79.86% and a custom accuracy of ~64.86%, placing it between the XGBoost and Linear Regression benchmarks. Although not outperforming Random Forest, it demonstrated an ability to capture patterns overlooked by simpler models.

Prediction Behaviour

The DNN produced a more centred distribution of predicted values. Residuals revealed overestimation in the lower-mid-popularity range and slight noise in low-density regions. This is likely due to the flexible, non-linear architecture being sensitive to rare feature combinations - albeit the model has generalised nicely and isn't over-fit. This proves the potential for DNNs for popularity prediction is not only viable, but favourable.

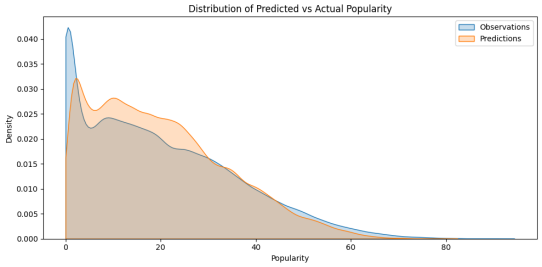

Figure 4.24: A KDE plot showing the distributional fit of predicted over observed data

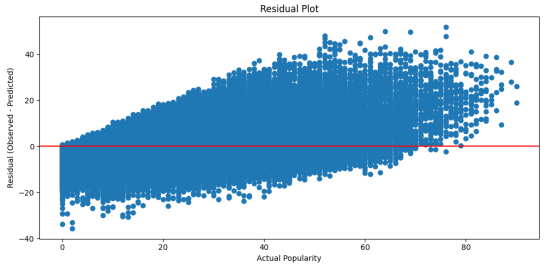

Figure 4.25: A residual plot of actual and observed values

Compared to linear and tree models, the DNN offered smoother but less calibrated responses, especially where training data was sparse or noisy – despite the use of *RMSProp*.

Output Distribution and Interpretation

The output distribution of the DNN was plotted against actual popularity values.

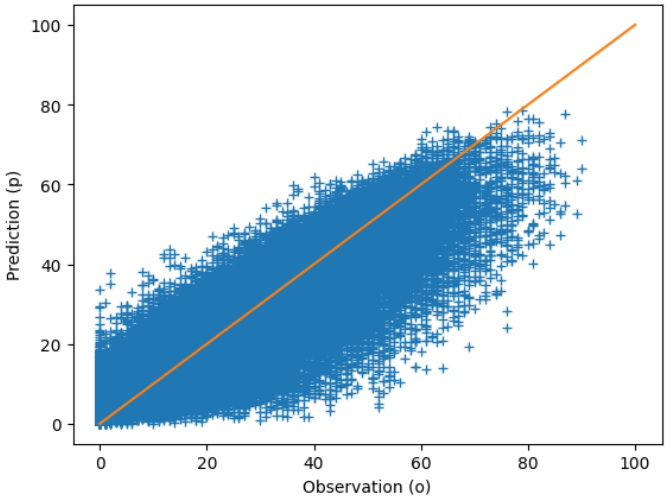

Figure 4.26: A plot of observed values against predicted values

Unlike Random Forest, which displayed sharp thresholds and splits, the DNN distribution was more continuous and realistic in shape. This indicates that while the DNN is not the most precise, it produces outputs that better resemble real-world distributions.

Considerations and Future Potential

Despite slightly lower accuracy than Random Forest, the DNN's smooth output and expressive capacity offer strong future potential. It could serve as the foundation for:

- A real-time input optimiser for creative tools
- Future integrations with lyrics, audio spectrograms, or additional modalities
- Deployment in scenarios where continuous or probabilistic predictions are preferred

- Used in conjunction with user interaction data

At present, it stands a strong model, and – based on analyses – is the most realistic predictor for predictions in this scenario, hence it's selection for use in the User Interface system built in Section 5 of the Notebook.

Summary:

The Deep Neural Network demonstrated the feasibility of non-linear modelling in popularity prediction, capturing nuanced patterns with reasonable accuracy suitable for use in the UI. Its smoother predictions and flexible architecture show promise for future multimodal and real-time systems above other models, despite lower accuracy scores than Random Forest.

4.7 Comparative Model Evaluation

With four models implemented and tested-Linear Regression, XGBoost, Random Forest, and Deep Neural Network – this section summarises their relative strengths and weaknesses. The purpose is to contextualise their performance holistically and inform the final model choice for deployment.

4.7.1 Performance Summary

Each model was evaluated using the same `train_df` dataset and assessed via R^2 score and a custom proportional accuracy metric. Results are summarised in the table below:

Table 4.1: Summary of model performance metrics

Model	R^2 Score (%)	Custom Accuracy (%)
Random Forest	81.91	66.21
Deep Neural Network	79.81	64.86
XGBoost	79.76	64.86
Linear Regression	76.2	63.02

While Random Forest outperformed the other models on both metrics, confirming its suitability as the most robust predictor would be a mistake - we identified prior that, despite being the most accurate quantitively, it's visual distribution and behaviour was less ideal than the depth, generalisation and tuning of the DNN. It's important to understand that accuracy score (especially in complex regression) is not the penultimate indicator of performance – this can be understood that an accuracy score of 100% would signify a heavily over-fit model – not a top performer.

4.7.2 Trade-offs and Interpretability

Each model offers unique trade-offs:

- **Random Forest** balances strong accuracy with transparent feature importances and consistent behaviour across feature ranges. Its ensemble nature offers robustness without the complexity of deep networks.
- **XGBoost** provided comparable performance with more tuning flexibility and faster execution but showed sensitivity to extreme values and a narrower margin of improvement.

- **Deep Neural Network** offered flexibility and smoother predictions, showing the most promise, especially for future extensions with richer feature inputs and temporal awareness.
- **Linear Regression** performed weakest overall but proved valuable as a baseline for initial benchmarking and understanding direct feature effects.

Summary: Random Forest – as expected by literature – persisted as the most effective model in terms of accuracy, delivering the highest performance while preserving interpretability and operational consistency – the only key drawback being its slow performance and over-representation of extreme popularity values and a lack of nuance, generalisation and critical depth that the DNN provides, thence determining the DNN as our choice for the UI predictor system.

4.7.3 Feature Sensitivity: Controlled Sweeps

To evaluate the validity, responsiveness and interpretability of the DNN model, a controlled feature sweep was conducted using `artist_avg_popularity`. This method isolates the effect of a single feature by systematically varying its value while keeping all others constant.

Method

The sweep was executed by sampling random base rows from `train_df`, then overriding the `artist_avg_popularity` value with increments from 0 to 100 (scaled to [0, 1]). This allowed comparison of predicted popularity across the full range of artist reputations.

Each modified input is passed to the Deep Neural Network model for predicting the popularity of each input.

Results and Observations

The Deep Neural Network model showed a clear positive trend: higher `artist_avg_popularity` values consistently resulted in higher predicted song popularity. However, the predictions at lower input values exhibited substantial variance, while those at higher popularity levels became increasingly stable and consistent.

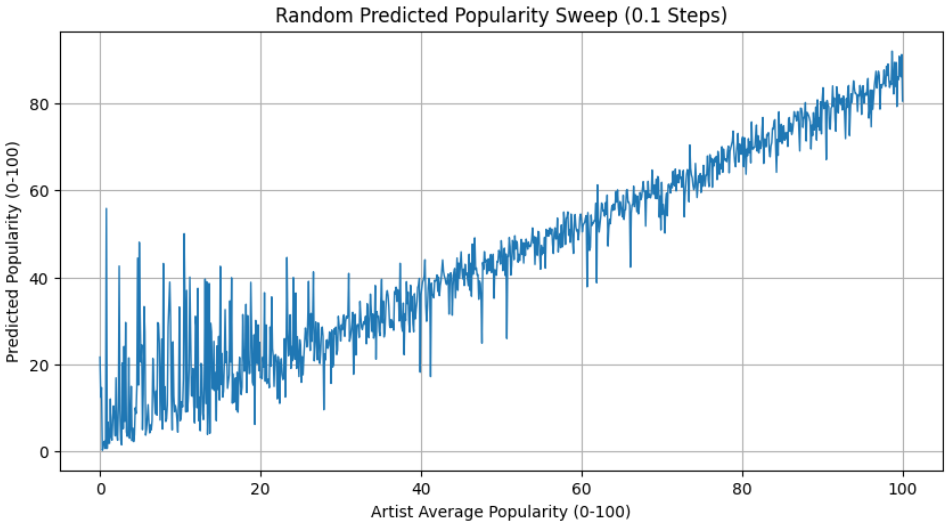

Figure 4.27: Line plot – predicted popularity vs `artist_avg_popularity` Sweep (DNN)

The graph confirms the model's strong sensitivity to artist reputation. As `artist_avg_popularity` increases, predicted popularity scores rise steadily—eventually exceeding ~85 in the upper range. While the results are more volatile in the lower spectrum, the consistent upward trajectory across the entire range reinforces the model's practicality for real-world prediction tasks.

Interpretive Value

Controlled sweeps provide insight into how models generalise individual input changes. In this case, DNN's smooth curve reveals it handles edge cases cautiously, likely due to regularisation.

These response patterns are especially useful for transparency in practical use – particularly in scenarios where model predictions may influence creative or strategic decisions.

4.7.4 User Interface Results

An essential component of this project was the development of an interactive, user-friendly interface for exploring model predictions. This user interface (UI), implemented within a Jupyter Notebook environment using `ipywidgets`, allows real-time testing of feature configurations and serves as a pedagogical tool for understanding musical feature impacts on predicted popularity.

Interface Design and Functionality

The interface presents users with:

- Dropdown menus for categorical features (e.g. genre, mode, key, release year), mapped to human-readable labels.
- Sliders for normalised continuous features (e.g. danceability, energy, valence, tempo), with values between 0 and 1.
- Direct numerical inputs for integer-based fields (e.g. `artist_song_count`).

All inputs are dynamically mapped to the format expected by the trained model (`train_df` structure). The interface handles encoding, scaling, and prediction generation internally.

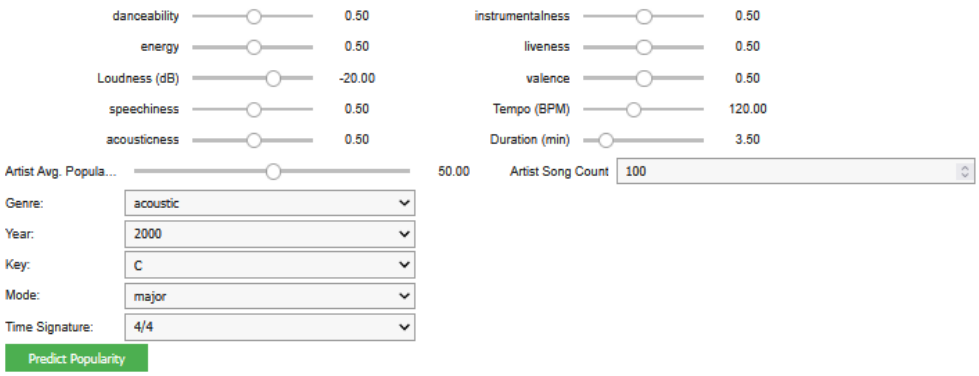

Figure 4.28: Screenshot of the user interface layout.

Live Prediction and Example Outputs

Upon user interaction, the interface produces an immediate popularity prediction based on the Random Forest model. Example outputs include (`artist_avg_popualrity` set to 50:

- A song with high danceability, valence, and genre = `Dance Pop` yields predicted popularity ~65–80.
- A song with low energy and genre = `Jazz` shows lower predicted popularity ~ 30–40.

These outcomes reflect model logic discussed in earlier sections and demonstrate internal consistency with feature importances and correlation patterns.

Table 4.2: Example input–output pairs from UI tests

Feature Configuration	Genre	Predicted Popularity
High danceability/valence	Pop	77.3
Low energy/acousticness	Jazz	36.2

Interpretability and Educational Use

The UI provides a direct, accessible bridge between model internals and user experimentation. It allows:

- Musicians to explore what traits may increase the likelihood of popularity.
- Researchers to test hypotheses about audio–metadata relationships.
- Educators to demonstrate ML model behaviour through interactive simulation.

This interface embodies the project's interpretability and accessibility goals, extending the utility of the model beyond static reporting.

Limitations and Development Scope

While the UI performs well as a demonstrator, several limitations are acknowledged:

- No input validation is currently enforced; users can enter musically implausible combinations.
- Visual feedback on prediction confidence, uncertainty, or sensitivity is not yet implemented.

Planned improvements include integration of the existing penalty function to warn against unrealistic feature profiles, as well as optional visual plots for model response curves.

Summary: The UI system successfully demonstrates the real-time application of the project's predictive pipeline. It offers intuitive input mechanisms, live feedback, and interpretive transparency, establishing a strong foundation for future interactive and educational extensions.

4.7.5 Summary of Findings

This chapter has provided a structured analysis of the results derived from modelling Spotify song popularity using internal features only. The insights obtained across correlation analysis, model evaluation, and feature interpretation offer a cohesive narrative of both predictive success and methodological robustness.

Predictive Feature Insights

Key engineered features `artist_avg_popularity`, `genre_avg_popularity`, and `year_avg_popularity` consistently ranked highest in both correlation and feature importance analyses. These metadata signals proved more predictive than audio features alone and retained their influence under temporal correction.

Notably, the use of `temporal_df` revealed that `genre_avg_popularity` is a more generalisable and temporally fair predictor than artist-based signals, reinforcing the importance of data causality in model interpretation.

Practical Extensions

The project demonstrated that predictive modelling can be made interpretable and actionable through user interfaces. The final UI allows interactive testing of input combinations, offering an exploratory platform for understanding feature–outcome relationships.

The results also lay groundwork for future functionality, including real-time feature optimisation, penalty integration for unrealistic combinations, and multimodal data extensions.

Summary:

Predicting song popularity using internal attributes is not only feasible but can achieve high accuracy when structured correctly. The DNN offers a balance of generalisability and nuance suitable for both research and deployment. These results establish a reliable predictive baseline and provide the analytical foundation for the broader discussion of implications in Chapter 5.

Chapter 5

Discussion and Analysis

5.1 Interpretation of Results and Evaluation of the Modelling Approach

The findings from Chapter 4 offer a consistent and revealing picture: engineered metadata features dominate prediction accuracy across all models. Artist, genre, and year-level statistics emerged as the most influential, with Random Forest and XGBoost ranking them highest in feature importances. Even within the Deep Neural Network (DNN), performance sweeps revealed strong dependencies on artist-level metadata, particularly `artist_avg_popularity`. Importantly, leak-aware correlation analysis using `temporal_df` demonstrated that once future information was excluded, `genre_avg_popularity` became the most reliable and causally valid predictor, confirming the structural importance of musical context.

Each modelling approach revealed different strengths and limitations. Linear Regression offered transparency and a benchmark for interpretability, but failed to capture non-linear relationships. XGBoost delivered strong performance but showed inconsistencies at prediction extremes. Random Forest proved accurate and stable, particularly in the mid-range of popularity, but often exaggerated upper and lower tail predictions. The DNN, while slightly less accurate than Random Forest in numerical metrics, produced smoother output distributions and demonstrated better generalisation to complex feature interactions. Most notably, it was selected for the user interface system due to its expressive capacity, realistic output range, and suitability for real-time deployment.

The interpretability–complexity trade-off was approached pragmatically. While Random Forest offered explicit feature importance, the DNN's architecture – including separate branches for categorical and continuous inputs – allowed interpretable structure even without formal SHAP or LIME integration. Controlled sweeps over dominant features such as `artist_avg_popularity` served as a valuable interpretive proxy, demonstrating the model's learned sensitivity to relevant variables.

In sum, the models successfully internalised real-world dynamics of popularity, capturing not only structural metadata signals but also the nuanced role of artist reputation and temporal context. The DNN's deployment in the UI confirmed its capacity to generalise and provide practical outputs for interpretive and creative exploration.

5.2 Significance of the Findings

The project validates the premise that Spotify popularity is not entirely unpredictable – it can be meaningfully modelled using internal features alone. With R^2 scores approaching 82% and consistent behaviour across models, this work demonstrates that a significant proportion of a song's popularity can be explained by its structure, context, and metadata – without relying on engagement signals.

Among these features, engineered metadata – including `artist_avg_popularity`, `genre_avg_popularity`, and `year_avg_popularity` – proved consistently dominant. Their predictive power exceeded that of any raw audio feature, reinforcing the notion that popularity is as much about context and position as it is about musical content. The use of `temporal_df` ensured that these insights were not compromised by temporal leakage, strengthening their causal interpretability and confirming that genre-level context is one of the most stable predictors.

From a methodological perspective, the project achieved a rare balance of performance, clarity, and responsible design. Key choices – including pruning artist-aligned entries, constructing temporal-safe features, and building a user interface grounded in model logic – elevated both the credibility and usability of the system. The decision to use a DNN in the UI, based on its balance of generalisability and responsiveness, further advanced the project's goal of bridging prediction with interpretability.

These findings contribute to ongoing work in MIR and creative AI by demonstrating that predictive models can offer transparency, not just automation. Rather than simply forecasting success, they offer a lens into how musical structure aligns with cultural reception – a valuable tool for researchers, curators, and creators alike.

5.3 Limitations

Despite the strength of the modelling framework, several limitations constrain generalisability and future deployment.

First, the dataset reflects Spotify's catalogue bias – favouring English-language, Western, and algorithm-promoted tracks. The exclusion of user engagement signals (e.g., streams, skips, playlist placements) was deliberate and justified, but it necessarily limits the system's ability to account for exposure-driven popularity dynamics.

Second, while `temporal_df` was constructed to support leak-aware correlation analysis, final models were trained on `train_df`, which includes future-aware aggregate metadata. As such, the predictive power of features like `artist_avg_popularity` may reflect retrospective privilege, introducing risks of circularity if used for forecasting. Although feature sweeps and causal-safe exploration help contextualise this issue, future models should fully integrate temporally valid training data.

Third, the DNN, though effective and well-behaved, remains a partially opaque model. No formal interpretability framework (e.g., SHAP, LIME) was implemented, limiting traceability of predictions beyond architecture and sweep tests. While this was mitigated through careful UI design and controlled inputs, further interpretability work would be necessary for production environments.

Fourth, the user interface, while functional and pedagogically rich, lacks enforcement of input realism. Implausible combinations can be constructed, and no penalty system or

optimiser module is yet integrated. This limits its use as a prescriptive or diagnostic tool, though it remains a strong exploratory platform.

Lastly, the scope of features was limited to structured metadata and audio-derived vectors. Important modalities such as lyrics, social context, and artist branding were excluded. The models were also static and non-sequential, preventing any analysis of evolving artist trajectories over time.

These limitations were necessary to ensure methodological control and focus, but they set clear directions for future work.

5.4 Summary

This chapter has interpreted and evaluated the key findings of the project, situating them within the broader goals of predictive modelling, interpretability, and methodological fairness. The results showed that engineered metadata – particularly `artist_avg_popularity`, `genre_avg_popularity`, and `year_avg_popularity` – are the strongest predictors of popularity, and that structured modelling approaches can capture these relationships effectively.

While Random Forest delivered the highest accuracy, the Deep Neural Network offered smoother, more realistic outputs and was better suited to generalisation and interface integration. The selection of the DNN for deployment in the user interface reflects this trade-off between expressive modelling and practical application.

Causality-aware analysis using `temporal_df` helped isolate the most robust and ethically interpretable features, confirming that success is not random but shaped by structural, temporal, and contextual signals. Though limitations remain – including temporal circularity in training, interpretability gaps in the DNN, and UI realism – the findings validate the project's hypothesis: musical popularity can be predicted from internal features, and those predictions can be made transparent and interactive.

Chapter 6

Conclusions and Future Work

This final chapter reflects on the project as a whole, drawing together the empirical findings and methodological design choices to answer the central research question: to what extent can internal audio and metadata features be used to predict Spotify song popularity? It then outlines well-motivated avenues for future development, grounded in the limitations and opportunities identified in Chapter 5.

6.1 Conclusions

The aim of this project was to determine whether the popularity of a song on Spotify could be accurately predicted using only internal audio features and structured metadata, deliberately excluding user engagement metrics such as stream counts or playlist placements. This constraint shaped every stage of the pipeline — from data preprocessing to model selection and deployment — with a particular emphasis on interpretability, transparency, and fairness.

Using a dataset of over one million songs spanning the streaming era (2000–2022), the project developed three specialised data pipelines (`train_df`, `pruned_df`, and `temporal_df`) to support different analytical and modelling objectives. These pipelines enabled rigorous correlation analysis, leak-aware feature engineering, and the training of four predictive models: linear regression, XGBoost, Random Forest, and a deep neural network.

Key Findings

- **Popularity is structurally predictable.** The strongest model (Random Forest) achieved an R^2 of 81.91%, confirming that internal features — particularly contextual metadata — contain sufficient signal to explain most of the variance in popularity.
- **Engineered metadata outperforms raw audio features.** Features such as `artist_avg_popularity`, `genre_avg_popularity`, and `year_avg_popularity` dominated all importance rankings. Their predictive power remained under causality-aware constraints using `temporal_df`, confirming their robustness and interpretability.
- **UI model selection prioritised generalisability.** Although Random Forest was the most accurate, a deep neural network (DNN) was selected for the user interface. Its architecture — with separate branches for continuous and one-hot inputs — produced smoother and more realistic outputs for real-time exploration.
- **Causal design choices improved validity.** The use of pruning, year-safe aggregations, and correlation filtering mitigated leakage and improved the interpretability of observed trends. In particular, `temporal_df` allowed safe exploration of feature relationships without forward-looking bias.

- **UI implementation promotes accessibility.** The final notebook-based interface allowed users to manually adjust feature sliders and dropdowns to observe how popularity predictions change. While still exploratory, the system enables transparency and bridges complex models with intuitive interaction.

Overall Positioning

Unlike many commercial systems that rely on black-box architectures or user engagement signals, this project adopted a structured, ethically-aware framework for interpreting popularity. Prediction was used not to automate judgement, but to uncover which internal traits systematically relate to public success. The findings affirm that popularity is not purely random or exposure-driven — it is shaped by genre context, artist history, and temporal positioning, all of which are modelled reliably through internal features alone.

6.2 Future Work

While the current system achieves strong results under realistic constraints, several avenues exist for future development. These are outlined below as modular extensions, each addressing a limitation or opening a new opportunity to improve performance, interpretability, or deployment realism.

1. **Causal-Safe Model Training**

   Although `temporal_df` was used for feature analysis and interpretation, all final models were trained on `train_df`, which contains future-aware group averages. To improve causal robustness:

   - Shift group aggregates (e.g., artist, genre, year) using '.shift(1)' before training.
   - Use rolling-window or time-series validation frameworks.
   - Integrate tools like DoWhy or EconML for formal causal probing.

2. **Optimiser Integration and Real-Time Suggestions**

   The current UI supports manual exploration. Optimisation can extend this by:

   - Enabling feature recommendations for target popularity.
   - Using `scipy.optimize` with domain-aware penalty functions to constrain suggestions.
   - Offering guided experimentation to surface hidden potential in song or catalogue features.

3. **Multi-Modal Feature Expansion**

   Additional modalities could improve model breadth:

   - NLP-based lyric sentiment or topicality.
   - CNN-based spectrogram embeddings for timbral richness.
   - Visual metadata from artwork or video.

   These could be integrated into multimodal neural networks or used as stand-alone enrichment layers.

4. **Improved Interpretability of Black-Box Models**

   Though sweep testing added interpretive value to the DNN, further work could include:

- SHAP or LIME visualisations.
- Self-attention layers to expose internal logic.
- Surrogate linear models for UI explainability.

5. **Dataset and Evaluation Scaling**

   To ensure external validity:

   - Add new Spotify data beyond 2022.
   - Apply to third-party benchmarks such as the Million Song Dataset.
   - Conduct qualitative studies with musicians, producers, or A&R professionals.

6. **Temporal and Sequential Modelling**

   To better reflect the evolution of music:

   - Explore RNNs or Transformers for time-aware learning.
   - Model artist trajectories and long-term development.
   - Enable dynamic popularity forecasting across releases.

In summary, this project offers a transparent, causally-aware, and exploratory approach to predicting song popularity. Its hybrid modelling strategy, UI integration, and empirical design provide a solid foundation for future academic, industrial, or creative work in music prediction and interpretability.

Chapter 7

Reflection

This project has been one of the most challenging and rewarding learning experiences of my degree. It provided an opportunity not only to apply technical skills, but to engage with research strategy, data ethics, and critical thinking in a deeply autonomous context. From problem identification to final delivery, the process required me to make iterative design decisions under uncertainty, constantly balancing trade-offs between accuracy, interpretability, and methodological rigour.

One of the most valuable lessons I learned was how to structure a complex machine learning pipeline from scratch. While I had prior experience with modelling, this project forced me to think carefully about causal leakage, overfitting, temporal validity, and the interpretability of features. Constructing the `temporal_df` pipeline, in particular, taught me how important it is to align analytical design with real-world data constraints – not just to boost accuracy, but to ensure the results are meaningful and ethically sound.

I also developed a deeper appreciation for user-facing design through building the interactive prediction interface. Translating a complex model into an intuitive UI challenged me to think from the user's perspective and emphasised the value of transparency. I learned how to use modular design patterns and visual feedback loops to make machine learning more accessible, without sacrificing analytical depth.

The most difficult part of the project was managing its scale. Working with over one million records and 100+ features required me to optimise code efficiency, manage memory constraints, and rethink my original plan several times. Not all aims were achieved – for example, I had initially hoped to deploy the UI as a live web application, but technical constraints and time limitations made this infeasible. If I were to restart the project, I would prototype the user interface earlier, as this would have informed several earlier modelling decisions.

This project has also reshaped how I think about data-driven systems. It highlighted the increasing dominance of metadata in algorithmic outcomes – a finding that, while analytically fascinating, also raised important questions about creativity, fairness, and the future of music curation. These reflections will shape how I approach both research and engineering in the future.

Overall, the project has not only advanced my technical proficiency in Python, TensorFlow, and data analysis, but also helped me mature as a researcher. It reinforced the importance of clarity, curiosity, and humility when exploring complex, high-dimensional problems – lessons I will carry with me into future work in AI and creative technology.

References

Chmiel, M., Prałat, P., Klamut, P. and Szymański, B. K. (2022), 'Analyzing and predicting success of professional musicians', *Scientific Reports* **12**(1), 21914.
URL: *https://www.nature.com/articles/s41598-022-25430-9*

Devgenius (2023), 'Mini ml project — predicting spotify songs' popularity'.
URL: *https://blog.devgenius.io/mini-ml-project-predicting-spotify-songs-popularity-part-1-ec1c906b8ff8*

Fang, Y., Wang, Y. and Jin, X. (2024), 'Predicting hit songs using audio and visual features', *MDPI Proceedings* **89**(1), 43.
URL: *https://www.mdpi.com/2673-4591/89/1/43*

Joseph, O. (2023), 'Using machine learning to predict song popularity based on their audio features', *Medium* .
URL: *https://medium.com/@oladokunjoseph2/using-machine-learning-to-predict-song-popularity-based-on-their-audio-features-7c64429af853*

Kumar, S., Chang, Y. and Liu, D. (2021), 'Modelling time-decay and timestamp uncertainty in behavioural event sequences', *Scientific Reports* **11**, Article 94229.
URL: *https://www.nature.com/articles/s41598-021-94229-1*

Miu, A. C. and Baltes, F. R. (2022), 'Music we move to: Spotify audio features and reasons for listening', *PubMed* .
URL: *https://pubmed.ncbi.nlm.nih.gov/36174020/*

Music as a Tool for Ethics (2022).
URL: *https://www.researchgate.net/publication/364130560*$_{M}usic_{a}s_{a}T ool_{f}or_{E}thics$

NHSJS (2025), 'The impact of artificial intelligence on music production: Creative potential, ethical dilemmas, and the future of the industry'.
URL: *https://nhsjs.com/2025/the-impact-of-artificial-intelligence-on-music-production-creative-potential-ethical-dilemmas-and-the-future-of-the-industry*

Putri, M. D., Santoso, H. B. and Purwandari, B. (2023), 'Predicting song popularity based on spotify's audio features: Insights from the indonesian streaming users', *Journal of Information and Telecommunication* **7**(3).
URL: *https://www.tandfonline.com/doi/full/10.1080/23270012.2023.2239824*

Rukavina, T., Zhao, C. and Li, F. (2024), 'Predicting music track popularity by convolutional neural networks on spotify features and spectrogram of audio waveform', *arXiv* .
URL: *https://arxiv.org/pdf/2505.07280*

Schedl, M., Gómez, E. and Serra, X. (2024), 'Surveying more than two decades of music information retrieval research on playlists', *ACM Computing Surveys* .
URL: *https://dl.acm.org/doi/10.1145/3688398*

Social Music Discovery: An Ethical Recommendation System Based on Friends' Preferred Songs (2024), *Multimedia Tools and Applications* .
URL: *https://link.springer.com/article/10.1007/s11042-024-19505-0*

The Impact of Playlist Characteristics on Coherence in User-Curated Music Playlists (2025), *EPJ Data Science* .
URL: *https://epjdatascience.springeropen.com/articles/10.1140/epjds/s13688-025-00531-3*

TU Dublin (2021), 'Feature engineering vs feature selection vs hyperparameter optimization in the spotify song popularity dataset'.
URL: *https://arrow.tudublin.ie/cgi/viewcontent.cgi?article=1021context=ittscicon*

UvT (2022), 'Predicting the popularity of spotify songs using a neural network'.
URL: *https://arno.uvt.nl/show.cgi?fid=171864*

Veritas AI (2024), 'Tuning into trends: Machine learning models for song popularity prediction on spotify'.
URL: *https://nhsjs.com/2024/veritas-ai-tuning-into-trends-machine-learning-models-for-song-popularity-prediction-on-spotify*

What Makes a Song Popular: Analysis Using Various Machine Learning Algorithms (2023), *Studies in Science of Science* .
URL: *https://sciencejournal.re/index.php/studies-in-science-of-science/article/view/754*

Appendix – Project Repository

All extensive code, models, and documentation associated with this project are available in the following GitLab repository:

https://csgitlab.reading.ac.uk/sw021313/cs3ip-notebook-and-accompanying-documents.git

This includes the final version of the notebook, supporting scripts, and all documentation referenced throughout this report.

Appendix – Further Feature Relationships

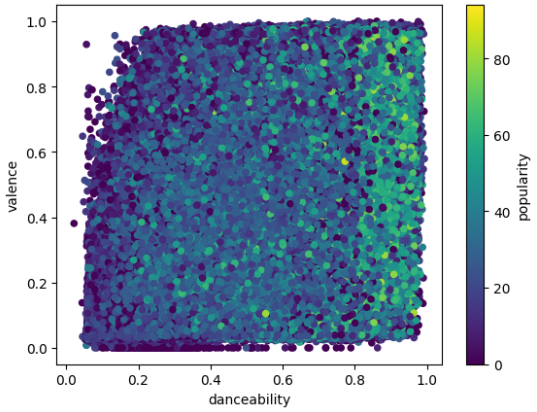

Figure 1: Valence, Danceability and Popularity.

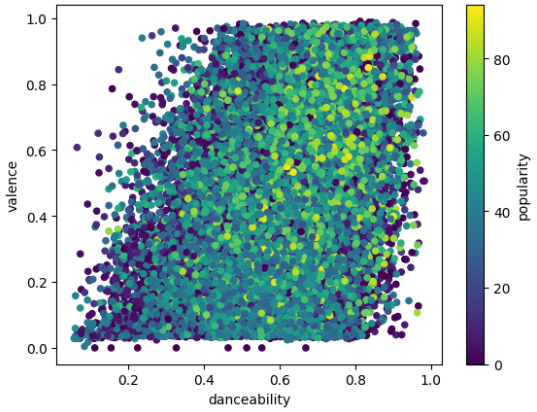

Figure 2: Valence, Danceability and Popularity (Pruned).

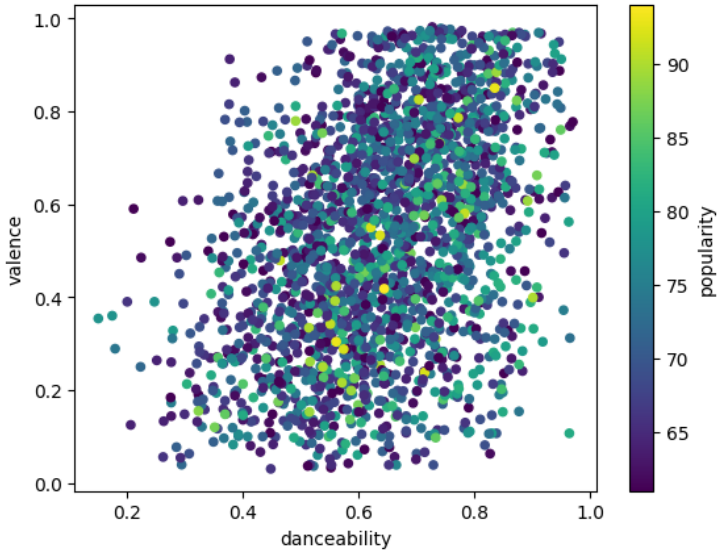

Figure 3: Valence, Danceability and Popularity Filtered where $popularity > \tilde{}70$ (Pruned).

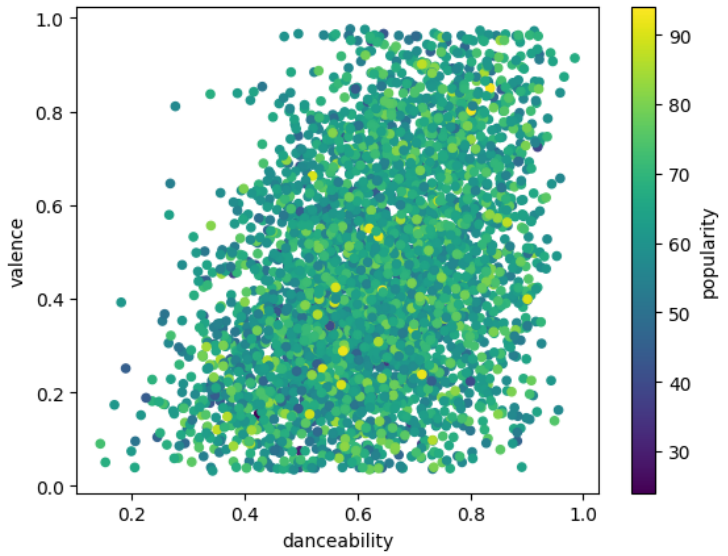

Figure 4: Valence, Danceability and Popularity Filtered where $artist\_avg\_popularity > \tilde{}60$ (Not Pruned).

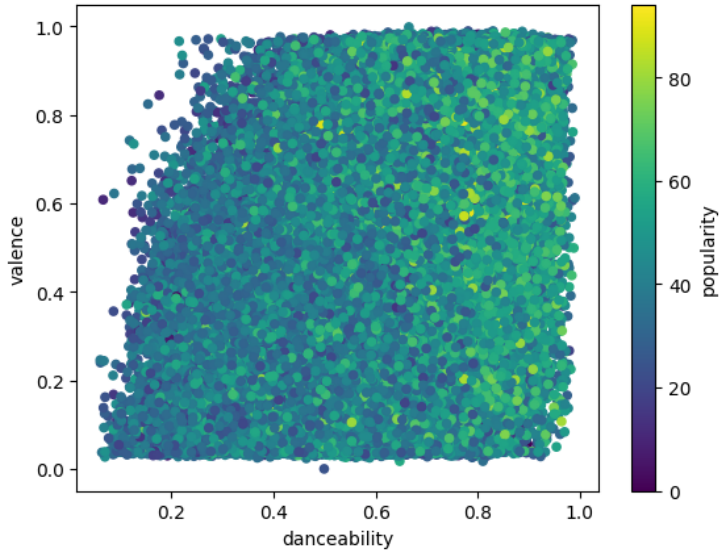

Figure 5: Valence, Danceability and Popularity Filtered where $artist\_avg\_popularity > \tilde{}30$ (Not Pruned).

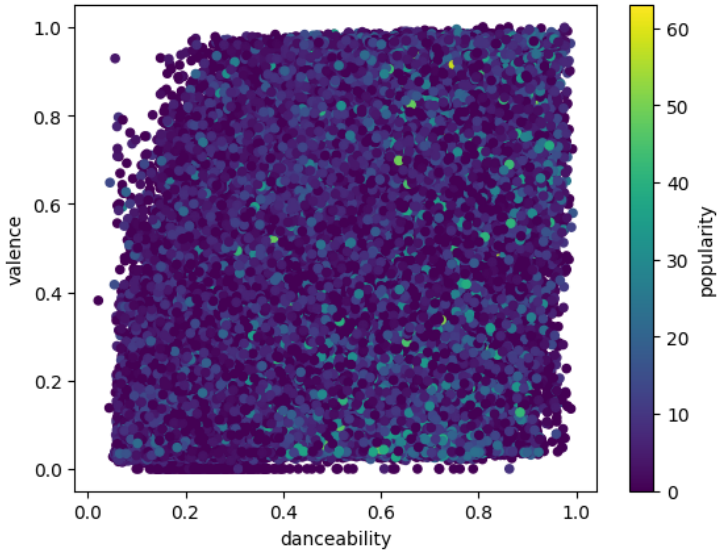

Figure 6: Valence, Danceability and Popularity Filtered where $artist\_avg\_popularity < \tilde{}10$ (Not Pruned).

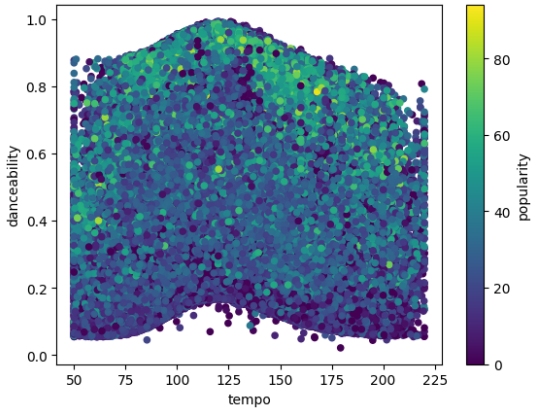

Figure 7: Danceability, Tempo and Popularity.

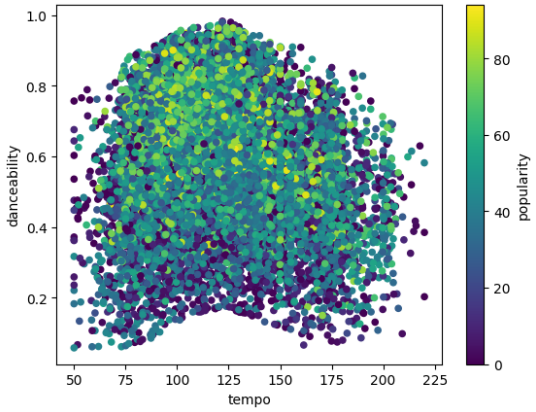

Figure 8: Danceability, Tempo and Popularity (Pruned).

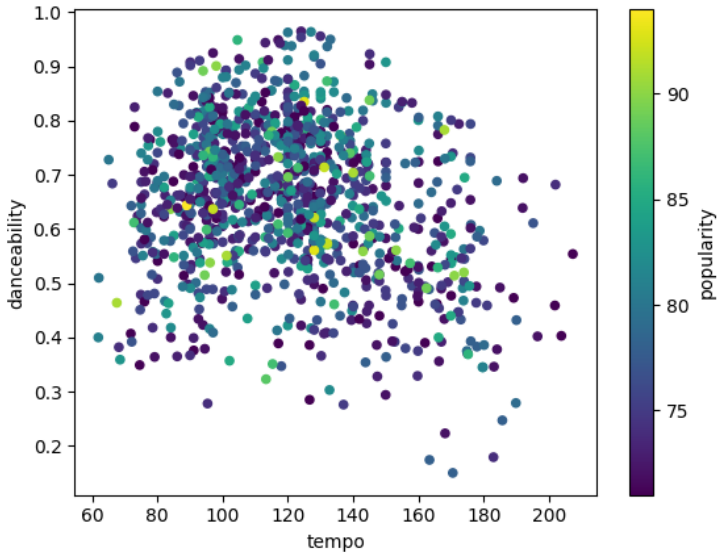

Figure 9: Danceability, Tempo and Popularity Filtered where $popularity > \tilde{}70$ (Pruned).

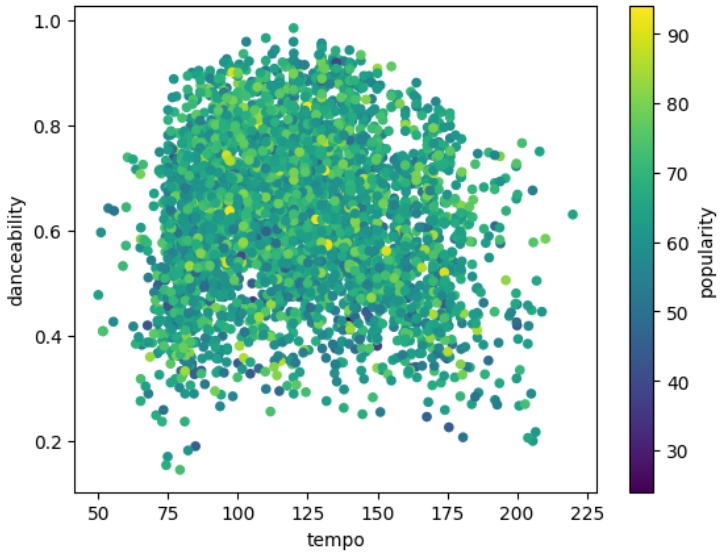

Figure 10: Danceability, Tempo and Popularity Filtered where $artist\_avg\_popularity > \tilde{}60$ (Not Pruned).

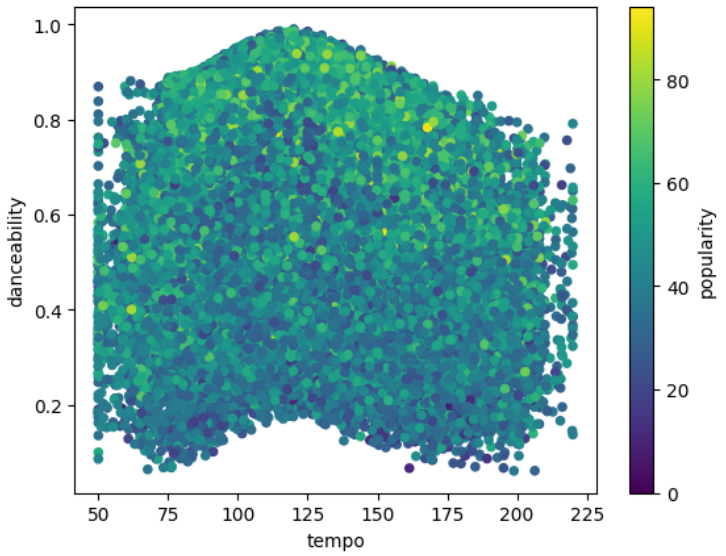

Figure 11: Danceability, Tempo and Popularity Filtered where $artist\_avg\_popularity > \tilde{}30$ (Not Pruned).

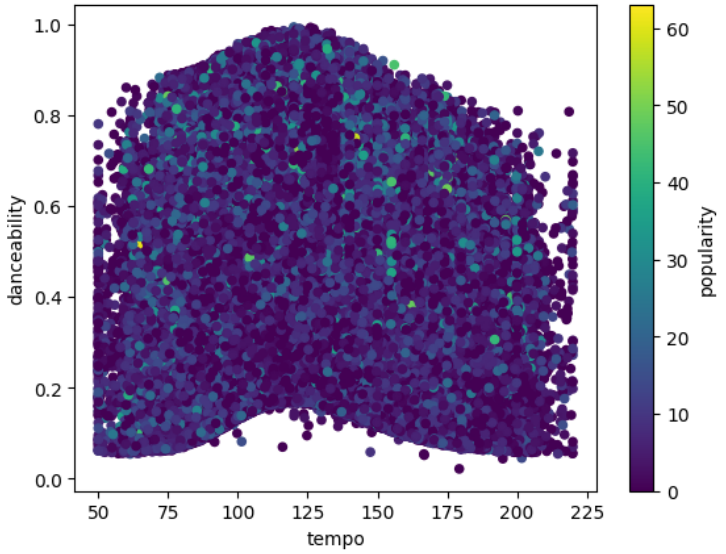

Figure 12: Danceability, Tempo and Popularity Filtered where $artist\_avg\_popularity < \tilde{}10$ (Not Pruned).

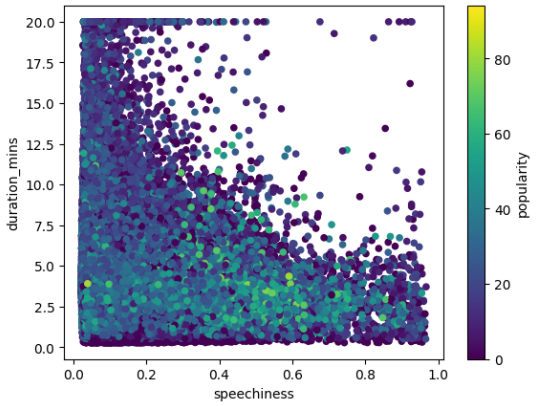

Figure 13: Danceability, Speechiness and Popularity.

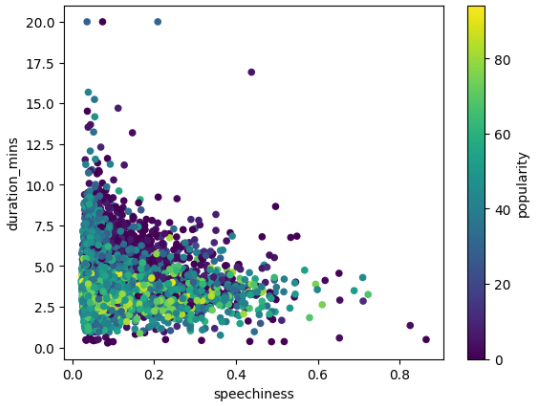

Figure 14: Danceability, Speechiness and Popularity (Pruned).

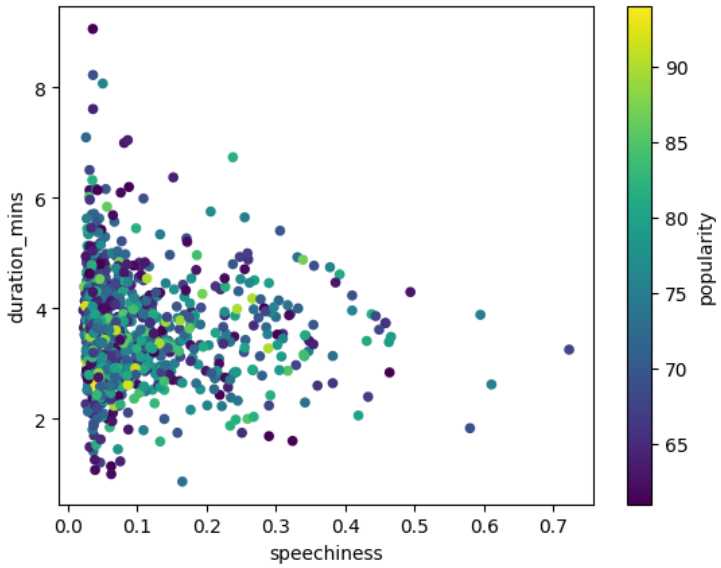

Figure 15: Danceability, Speechiness and Popularity Filtered where $popularity >$ ~70 (Pruned).

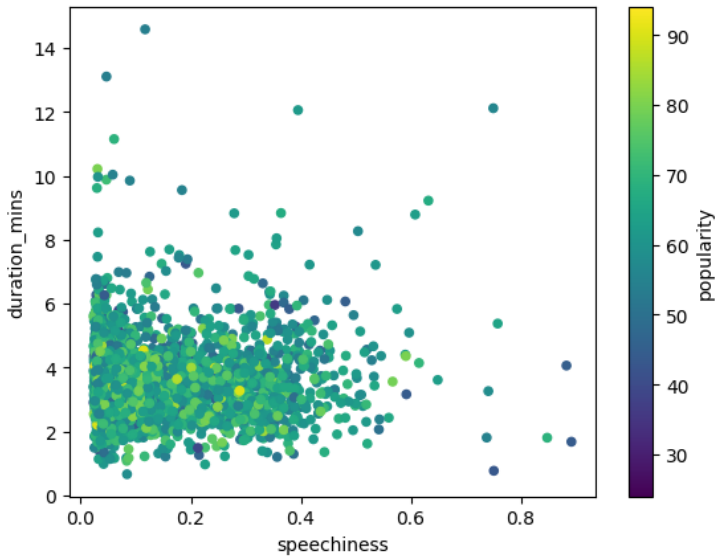

Figure 16: Danceability, Speechiness and Popularity Filtered where $artist\_avg\_popularity > \tilde{}60$ (Not Pruned).

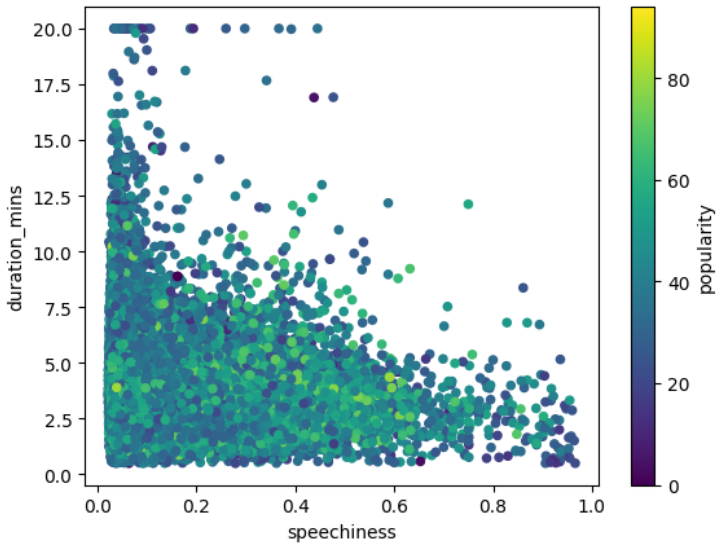

Figure 17: Danceability, Speechiness and Popularity Filtered where $artist\_avg\_popularity > \tilde{}30$ (Not Pruned).

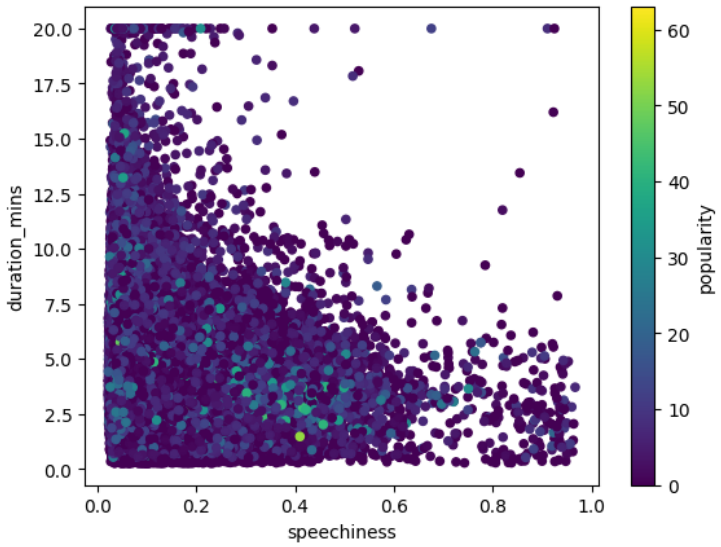

Figure 18: Danceability, Speechiness and Popularity Filtered where $artist\_avg\_popularity < \tilde{}10$ (Not Pruned).

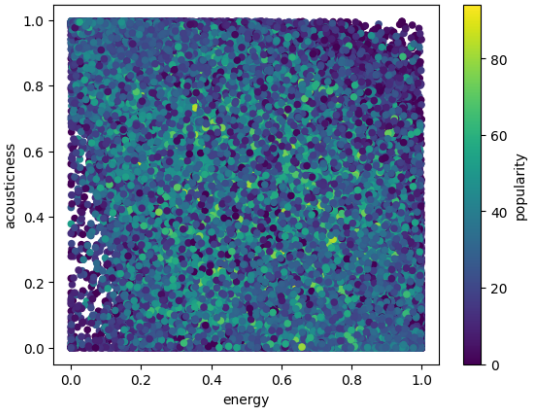

Figure 19: Acousticness, Energy and Popularity.

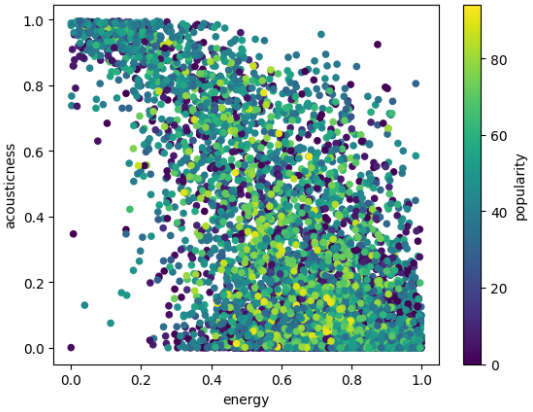

Figure 20: Acousticness, Energy and Popularity (Pruned).

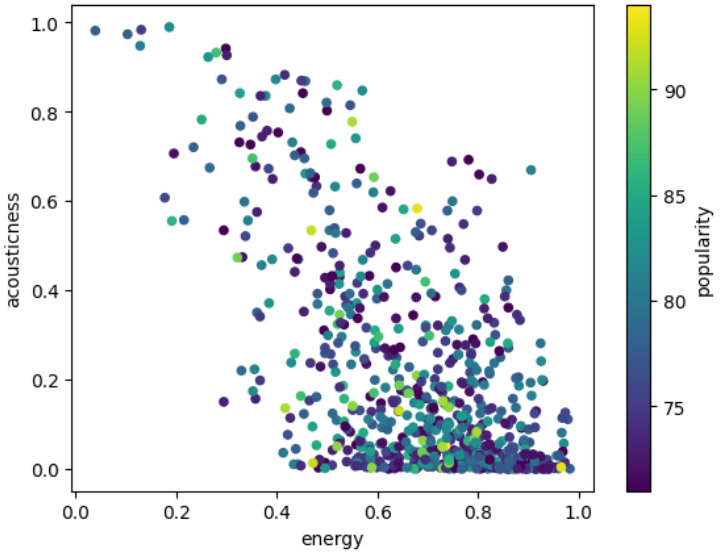

Figure 21: Acousticness, Energy and Popularity Filtered where $popularity > \tilde{}70$ (Pruned).

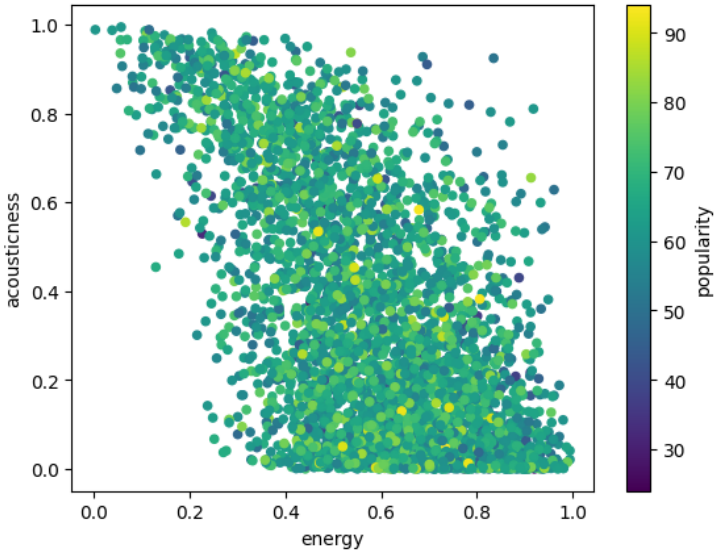

Figure 22: Acousticness, Energy and Popularity Filtered where $artist\_avg\_popularity > \tilde{}60$ (Not Pruned).

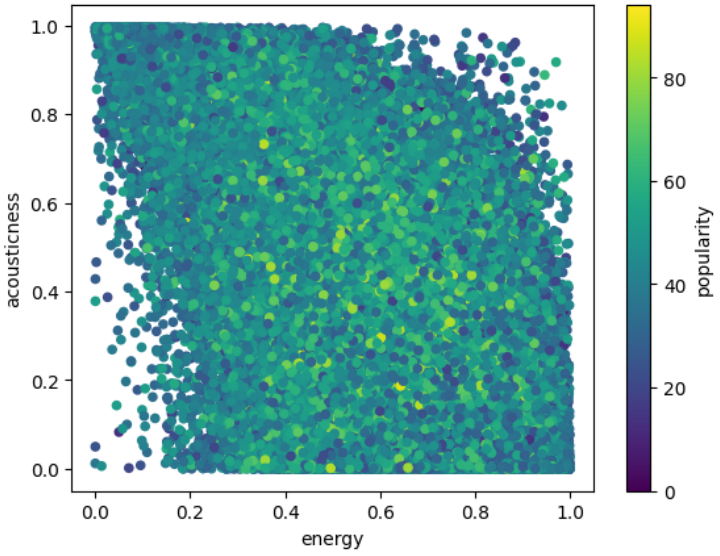

Figure 23: Acousticness, Energy and Popularity Filtered where $artist\_avg\_popularity > \tilde{}30$ (Not Pruned).

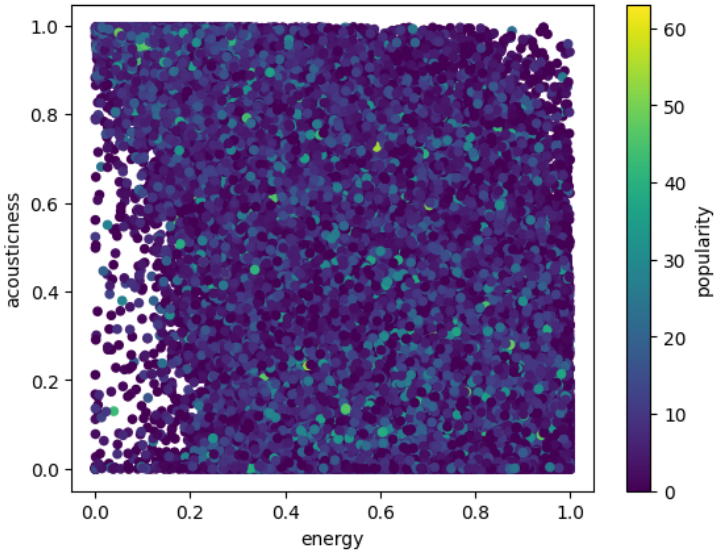

Figure 24: Acousticness, Energy and Popularity Filtered where $artist\_avg\_popularity < \tilde{}10$ (Not Pruned).

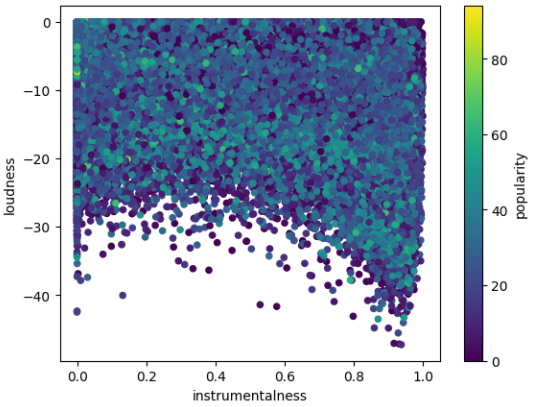

Figure 25: Liveness, Instrumentalness and Popularity.

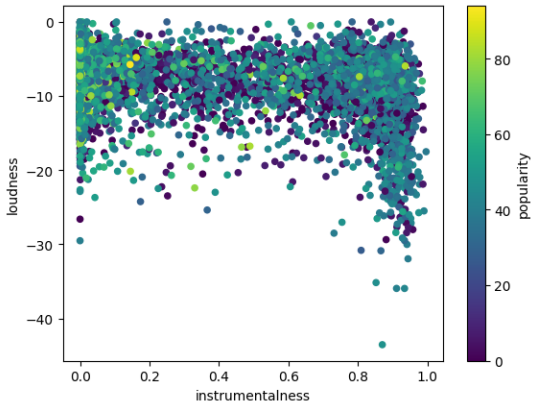

Figure 26: Liveness, Instrumentalness and Popularity (Pruned).

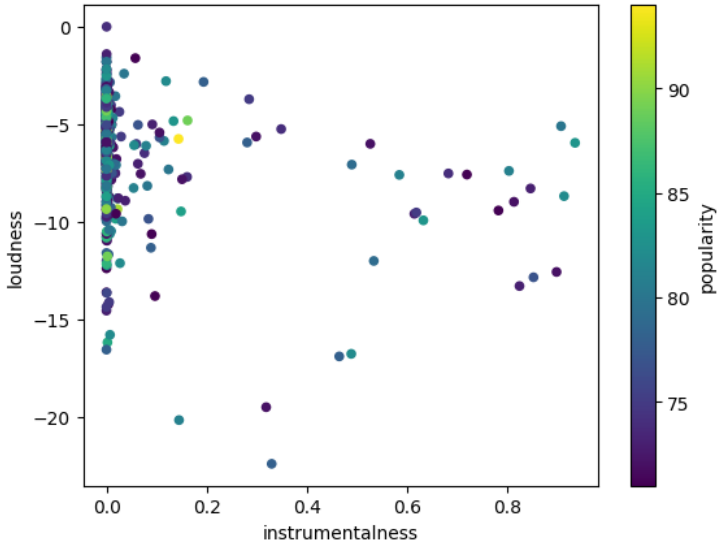

Figure 27: Liveness, Instrumentalness and Popularity Filtered where $popularity >$ ~70 (Pruned).

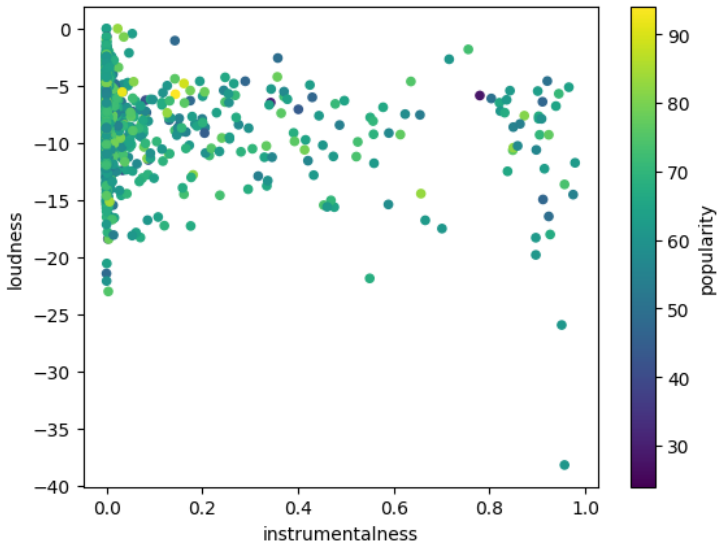

Figure 28: Liveness, Instrumentalness and Popularity Filtered where $artist\_avg\_popularity > \tilde{}60$ (Not Pruned).

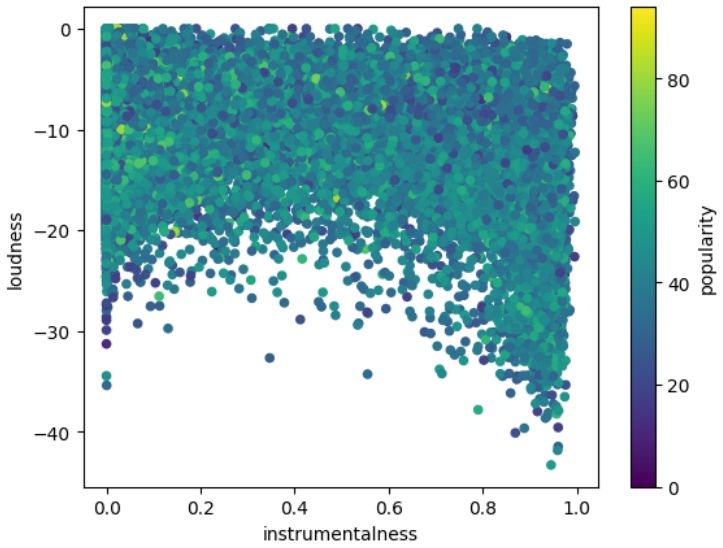

Figure 29: Liveness, Instrumentalness and Popularity Filtered where $artist\_avg\_popularity > \tilde{}30$ (Not Pruned).

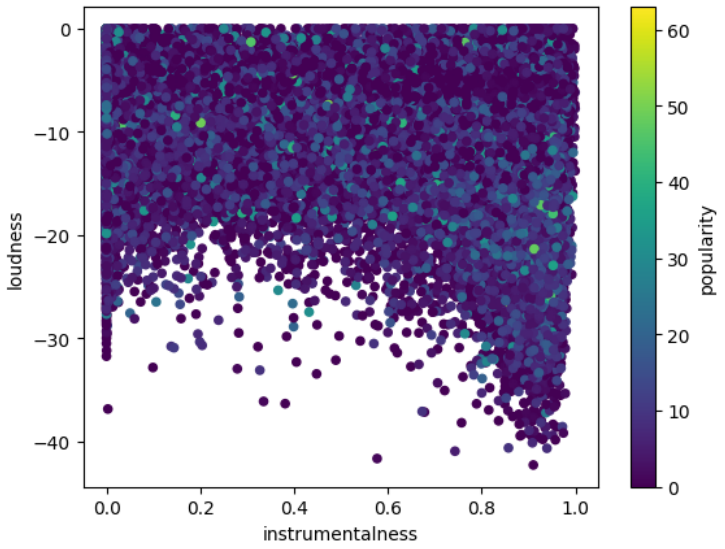

Figure 30: Liveness, Instrumentalness and Popularity Filtered where $artist\_avg\_popularity < \tilde{}10$ (Not Pruned).