Jose Cortes

December 2, 2024

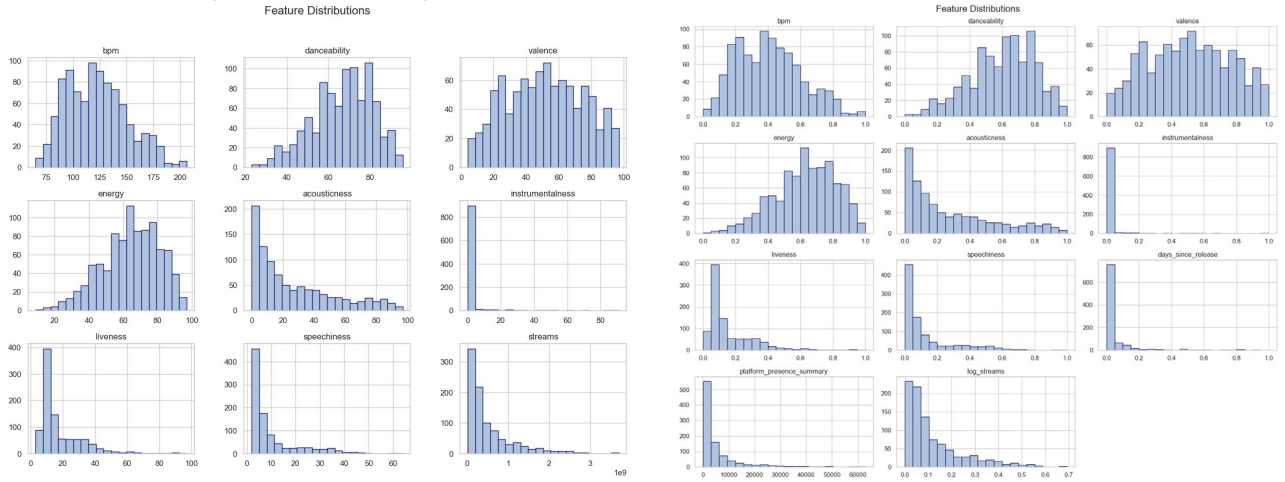
Predicting Spotify Song Popularity Using Deep Learning

Introduction

Have you ever wondered what makes a song popular on Spotify? In this project, I'm tackling that exact question by predicting the number of streams a track gets based on its features, things like tempo, energy, and danceability. Using a dataset of top Spotify songs from Kaggle, I've built a regression model to uncover how these characteristics influence a song's popularity.

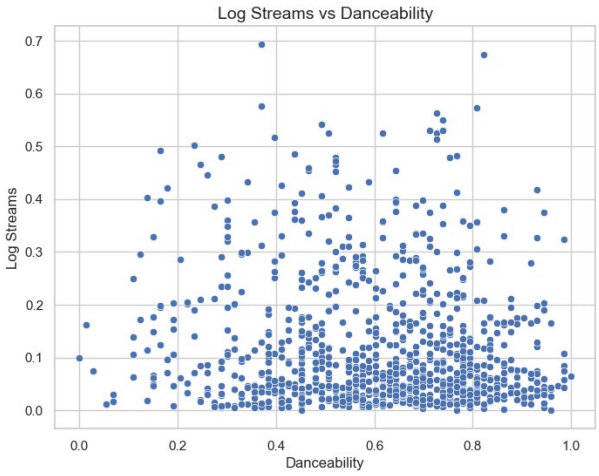
This project is exciting because it combines deep learning with music, something we all connect to. It's also a great way to apply the neural network concepts and techniques I've learned, like fine-tuning hyperparameters and evaluating models.

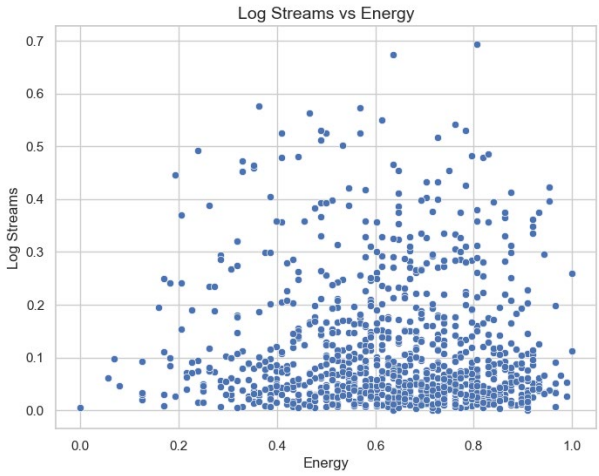
Beyond being a fun challenge, this project demonstrates how data can offer actionable insights for artists and producers, helping them understand what drives a song's success. By the end, I aim to build a reliable model, identify the most influential features, and see how well these tools can predict music popularity



Dataset Exploration

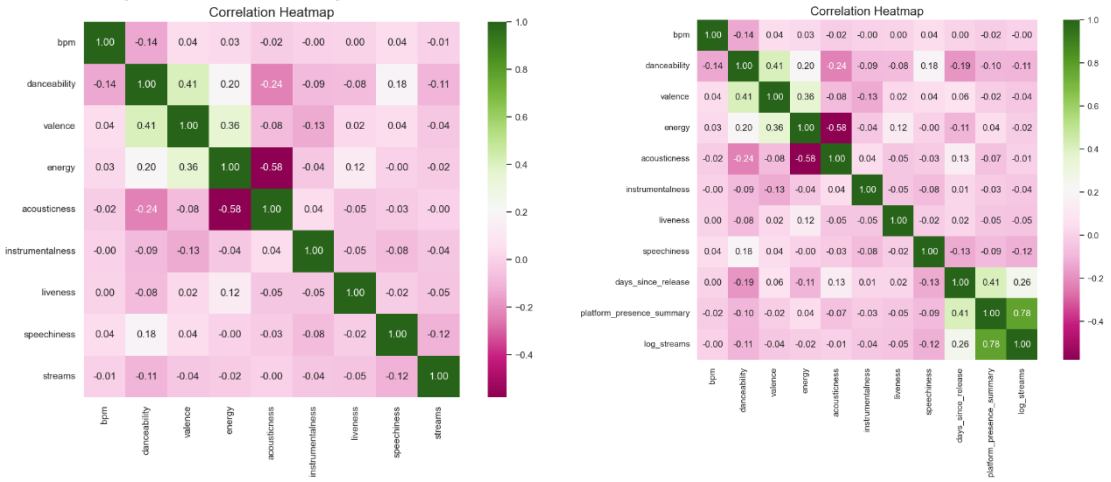
The dataset I used comes from Kaggle and features Spotify's top songs from 2023. It's a treasure trove of information about tracks, including attributes like danceability, energy, and streaming numbers.





This project's goal is to uncover trends in song characteristics and predict what makes a track popular, based on its streams. Key features include the track and artist names, audio attributes (e.g., danceability, energy), and even the song's presence on platforms like Apple Music or Shazam. The dataset combines numerical data (like streams and audio scores) with categories (like artist names) and dates (release dates).

During exploration, I found some interesting quirks: missing values in streaming stats, duplicate entries for a few tracks, and some outliers in audio features. Cleaning involved removing duplicates, filling in gaps, and normalizing features for consistency. For categorical columns, like platform presence, I used encoding to make them model-ready.

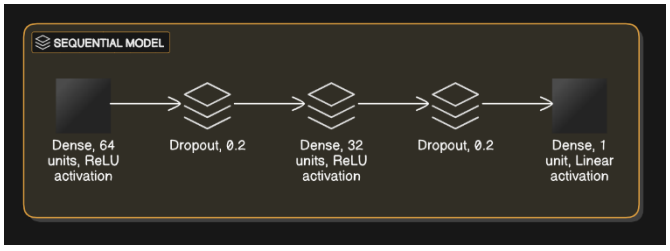


Visualizing the data revealed exciting patterns. For example, songs with higher energy and danceability often had more streams. Correlation heatmaps also highlighted relationships between features, giving clues about what might make a song a hit. These insights shaped how I prepared the data for modeling and guided my approach throughout the project.

Model Creation

For this project, I chose a Multi-Layer Perceptron (MLP) model. It's a type of neural network that's well-suited for regression tasks, like predicting the log-transformed stream counts. Since the dataset was tabular, with a mix of numerical and categorical features, an MLP felt like the perfect fit. It's flexible enough to capture complex relationships that simpler models might miss.

Initially, I considered linear regression during the data exploration phase. While it's a good baseline for understanding the data, I felt it might oversimplify the patterns in this case. The MLP, on the other hand, offers a richer representation of the data's underlying structure.

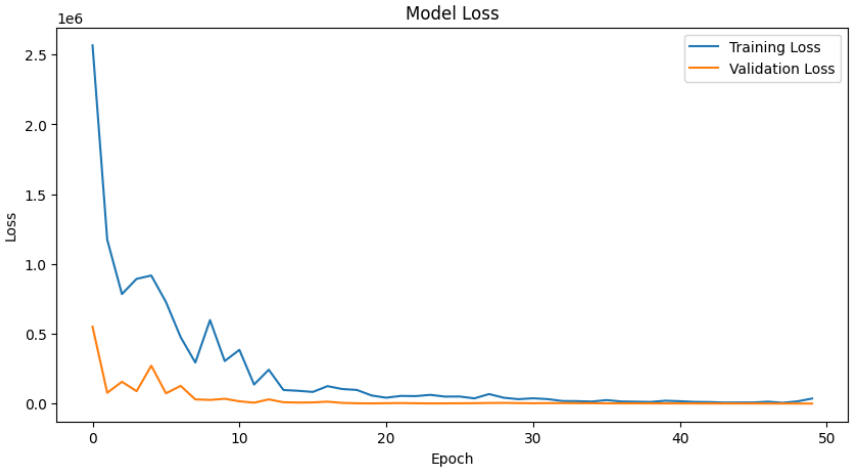


To build the model, I used TensorFlow and Keras. These libraries made it easy to define the architecture, which included:

- Two hidden layers with 64 and 32 neurons, respectively.

- ReLU activations for the hidden layers to handle non-linearity.
- A dropout layer (20%) to reduce overfitting.
- Adam optimizer to train the model efficiently, minimizing mean squared error.

Finding the right balance between complexity and performance was the hardest part. The initial version of the model overfitted the training data, so I introduced dropout layers and spent time tuning the hyperparameters, like the learning rate and number of neurons.



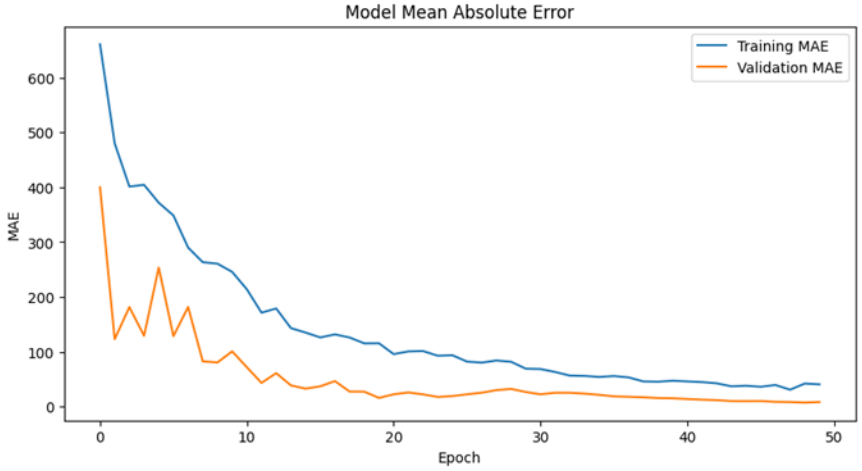
Overall, building the MLP model was a rewarding challenge. It showed me how small tweaks, like normalizing inputs or adjusting the dropout rate, can significantly improve performance. This model isn't just a tool for prediction; it's a testament to how deep learning can uncover patterns that aren't immediately obvious.

Hyperparameter Exploration

In building my model, I experimented with several key settings to improve performance:

- Learning Rate: I tested 0.001 and 0.01 to find a balance between training speed and stability.
- Dropout Rate: Tried 0.2 and 0.3 to control overfitting and improve generalization.
- Number of Neurons: Experimented with 64 and 128 to capture data complexity.
- Batch Size: Tested 16 and 32 for their impact on generalization and training efficiency.
- Number of Epochs: Compared 50 and 100 to gauge diminishing returns in training.

These choices weren't random; they were inspired by best practices, my prior experience, and early results. For example, a too-high learning rate caused instability, and initial overfitting hinted that dropout adjustments were needed.



I took a systematic approach, starting broad to understand trends, then narrowing the focus based on what worked. Using GridSearchCV helped efficiently test parameter combinations and observe interactions.

Some findings surprised me: a dropout rate of 0.3 effectively reduced overfitting more than expected, while increasing neurons to 128 captured subtle data nuances. On the flip side, tweaking batch size (16 vs. 32) had minimal impact, showing the model was robust to this parameter.

The learning rate and neuron count had the most significant impact, directly influencing convergence speed and accuracy. These experiments were critical for fine-tuning the model, and the final settings reflect a careful balance between complexity and performance.

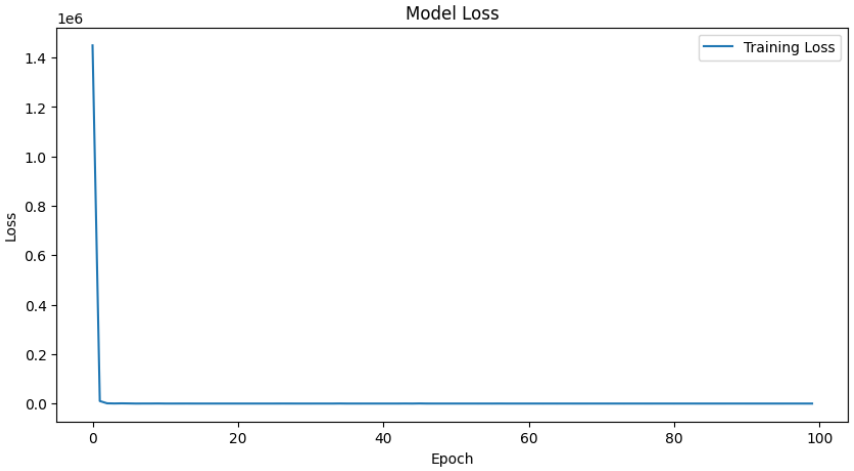
Results Documentation

The model's performance was evaluated using Mean Squared Error (MSE) and Mean Absolute Error (MAE), which are standard metrics for regression tasks. These metrics showed how close the predictions were to the actual values, with MAE providing an average of absolute errors and MSE emphasizing larger errors.

The MAE indicated that the predictions were close to the actual values on average, performing better than expected on the test set. The MSE, however, was slightly higher, highlighting the presence of a few larger errors. This pointed to challenges with outliers or edge cases in the dataset.

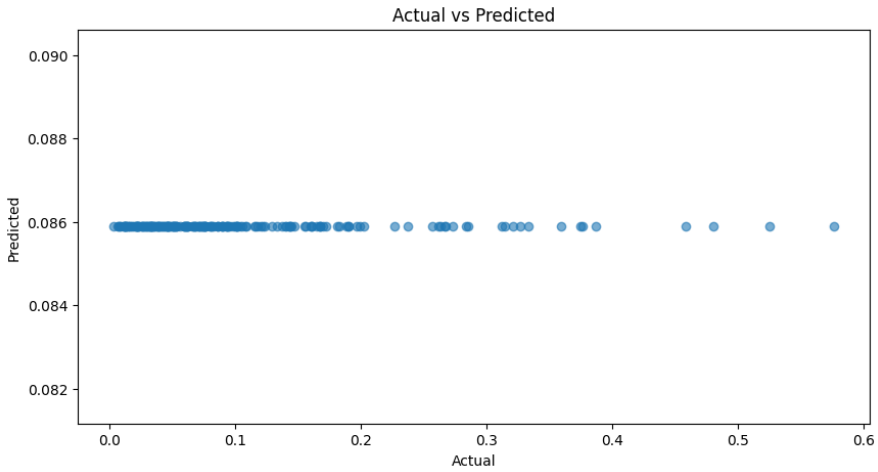
While these metrics provided a good overview, they did not capture everything about the model's performance. For example, areas like outliers or specific data segments were better

understood through visualizations. A scatter plot of actual vs. predicted values highlighted that most predictions were close, but some significant outliers inflated the MSE.



The model performed well in predicting features with strong correlations to the target variable, such as danceability and energy. However, it struggled with extreme cases, particularly in the streams feature, which had a skewed distribution.

Key visualizations that helped include the loss curve, which showed stable training and validation processes, and the scatter plot, which revealed consistency in predictions apart from a few anomalies. These visual tools provided insights that metrics alone could not.



In practical applications, this model could be used to predict trends in song popularity based on audio features. Improvements like handling outliers more effectively, experimenting with advanced architectures, and enhancing feature engineering would further improve the model's reliability and generalization.

The overall performance was satisfactory for a first iteration, with clear opportunities for refinement and growth.

Conclusion

This project was an exciting journey into the intersection of music and machine learning. I set out to predict what makes a song popular, and along the way, I uncovered meaningful insights about how features like energy and danceability correlate with streams. The MLP model performed well overall, despite challenges like handling outliers.

One of the biggest lessons I learned was the importance of balancing simplicity and complexity. Small tweaks in hyperparameters, like adding dropout or fine-tuning the learning rate, made a significant difference in the model's performance. I also gained a deeper appreciation for visualizing data; it revealed trends and gaps that numbers alone couldn't show.

Moving forward, there's so much room for growth. Exploring ensemble models, addressing outliers more effectively, or even incorporating additional datasets could make the model even better. This project is just the beginning of what's possible when we combine data with creativity!

References

Elgiriyewithana, Nidula. Top Spotify Songs 2023. Kaggle, 2023, https://www.kaggle.com/datasets/nelgiriyewithana/top-spotify-songs-2023/data. Accessed 2 Dec. 2024.

Colonna, Lucio. Spotify 2023: Focus on Artists. Kaggle, 2023, https://www.kaggle.com/code/lcolon/spotify-2023-focus-on-artists. Accessed 2 Dec. 2024.