BTAA – Data Alchemists

Project 2: Spotify Playlist Technical Report

Hritika Mishra
Aasish Tammana
Marcello Borromeo
Supriya Patki
Daniel Pai

***Abstract*—This report presents a comprehensive analysis aimed at predicting the popularity of Spotify playlists, which is quantified by the number of followers. Utilizing a dataset provided to us, which includes various attributes of playlists such as the number of tracks, albums, and artists, as well as the duration and number of edits made to each playlist, we employ several machine learning models to forecast playlist popularity. Our methodology encompasses initial data cleaning, exploratory data analysis to understand the dataset's characteristics and distributions, and rigorous model development. We experiment with multiple regression techniques including Linear Regression, Decision Tree, K-Nearest Neighbors, Support Vector Regression, and Neural Networks. Each model's performance is evaluated based on Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared values, enabling us to identify the most effective predictors and techniques for this application. The insights and findings from this study not only highlight the key factors influencing playlist popularity but also propose actionable recommendations for playlist creators to enhance follower engagement. This report aims to provide a foundational tool for Spotify and its community, offering strategic insights to optimize the curation and marketing of playlists.**

Keywords—Spotify, Playlist Popularity, Machine Learning, Predictive Analytics, Regression Analysis, Linear Regression, Decision Trees, K-Nearest Neighbors (KNN), Support Vector Regression (SVR), Neural Networks, Data Preprocessing, Exploratory Data Analysis, Mean Squared Error (MSE), Mean Absolute Error (MAE), R-squared

I. Introduction

A. Background

The objective of this project is to predict the popularity of Spotify playlists, as measured by the number of followers, using machine learning models. This involves analyzing a comprehensive dataset provided by Spotify, which includes various attributes such as the number of tracks, albums, artists, duration of tracks, and edits made to playlists. By leveraging this data, we aim to uncover patterns and predictors that significantly impact playlist popularity.

Figure 1: Spotify

Our approach includes thorough data cleaning, extensive exploratory data analysis to understand the dataset's structure and key features, and the development and evaluation of multiple predictive models. Models such as Linear Regression, Decision Trees, K-Nearest Neighbors, Support Vector Machines, and Neural Networks are employed and compared to identify the most effective method for predicting playlist followers.

This report details the methodologies used in our analysis, discusses the performance of each model, and provides insights and recommendations based on our findings. The results are intended to guide Spotify and playlist creators in optimizing playlist features to maximize follower engagement and satisfaction.

B. Dataset

The dataset for this project is sourced from Spotify and comprises various attributes of playlists, meticulously curated to include a diverse range of music genres, artists, and user behaviors. Here are the key features included in the dataset:

- Playlist Name: The title of the playlist.
- Collaborative: A Boolean attribute indicating whether the playlist is collaborative, meaning multiple users can add tracks.
- Playlist ID(pid): A unique identifier for each playlist.
- Modified At: The timestamp representing the last modification date of the playlist
- Number of Albums: The total number of unique albums featured in the playlist.
- Number of Tracks: The total count of tracks in the playlist.
- Number of Edits: How many times the playlist has been edited, which can indicate user engagement and satisfaction with the playlist.
- Duration of Tracks: The total duration of all tracks in the playlist, measured in milliseconds.
- Number of Artists: The count of unique artists featured in the playlist.
- Number of Followers: The number of users who follow the playlist, which is the target variable for our predictive models.
- Additional information related to the tracks present in each playlist which is irrelevant to our scope defined.

C. Data Preprocessing

The dataset was initially processed to effectively analyse using various machine learning models:

- Missing Values: The dataset contains some missing entries, particularly in features like 'Number of

Edits', 'Duration of Tracks', and 'Number of Artists'. These are addressed through imputation techniques to ensure robust model training.

- Data Type Conversions: Certain attributes, such as timestamps, are converted into more usable formats
- Normalization/Standardization: Numerical features are scaled to ensure that the model is not biased towards variables with larger magnitudes.

The richness of this dataset allows for a detailed exploration of the factors that influence the popularity of Spotify playlists, facilitating the development of models that can predict future trends and preferences in music streaming behaviour.

II. METHODOLOGY

A. Linear Regression Model

Linear regression is a fundamental statistical and machine learning technique used to predict a quantitative response. In the context of this project, we employ linear regression to forecast the number of followers of Spotify playlists based on various predictive features extracted from the dataset. This model assumes a linear relationship between the dependent variable (number of followers) and the independent variables (playlist features).

From the dataset, relevant features that potentially influence the number of followers are chosen. These include the number of tracks, albums, edits, duration, and artists. We also consider whether the playlist is collaborative as a categorical variable.

For linear regression, it's crucial to select features that are likely to have a linear relationship with the target variable.

The dataset is split into training and testing sets, typically using 80-20 split. This ensures that the model is trained on a portion of the data and validated on unseen data to test its predictive power.

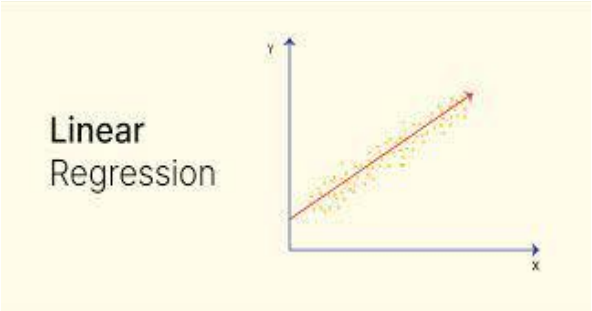

Figure 2: Linear Regression Model

B. Decision Tree Regression Model

Decision Tree Regression utilizes a decision tree (a flowchart-like structure) to predict the output value of a target variable by learning simple decision rules inferred from the training data's features. This model is particularly useful for capturing non-linear relationships that might not be easily modeled by linear methods.

A DecisionTreeRegressor from scikit-learn is configured with a specified random_state to ensure reproducibility of results. The random_state parameter fixes the way the algorithm splits the data at each node, influencing the model's behavior.

The decision tree model is trained on the training dataset (X_train, y_train) using the fit method. This process involves recursively splitting the data into purer subsets, using the feature that results in the highest reduction in variance (or another criterion in the case of other algorithms)

Once trained, the model is used to predict the number of followers on the test dataset (X_test) using the predict method. The predictions (y_pred_tree) represent the model's output based on the decision rules learned during training.

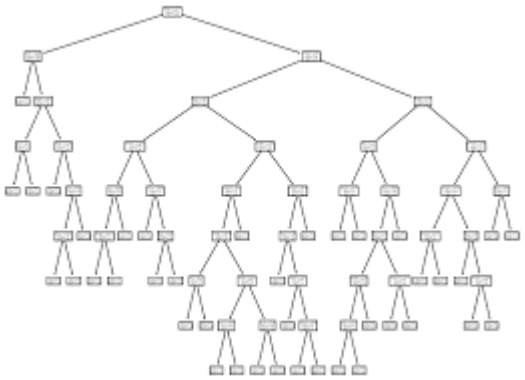

Figure 3: Decision Tree Model

C. K-Nearest Neighbors (KNN) Regression Model

K-Nearest Neighbors (KNN) is a simple, instance-based learning algorithm where predictions for new data points are determined based on similarity to existing data. In the context of regression, KNN predicts the target value by averaging the values of the K nearest neighbors to the query point.

The KNeighborsRegressor from scikit-learn is configured with n_neighbors=5, which specifies that the model should consider the five closest points (neighbors) in the training dataset to make predictions. The choice of K is crucial and can be optimized based on performance metrics.

Unlike model-based algorithms, KNN is a lazy learner that doesn't explicitly learn a model (e.g., there are no coefficients to determine as in linear regression). Instead, fit essentially stores the training dataset which will later be used during the prediction phase.

For prediction using predict, KNN identifies the K-nearest neighbors in the training data for each point in the test dataset (X_test) based on a distance metric (usually Euclidean distance).

The target value for each test point is estimated by averaging the target values of its nearest neighbors.

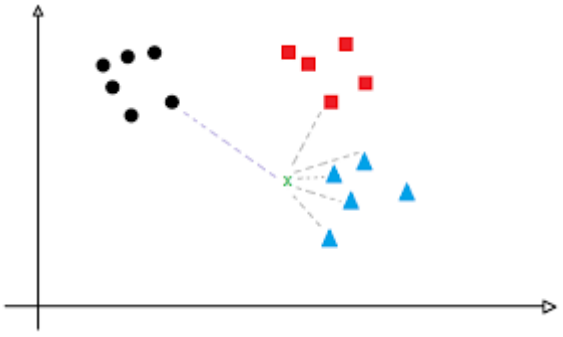

Figure 4: KNN Model

D. *Support Vector Regression (SVR) Model*

Support Vector Regression (SVR) is an extension of the Support Vector Machine (SVM) algorithm to handle regression tasks. It uses the same principles as SVM for classification, aiming to fit the error within a certain threshold. SVR provides a flexible approach to regression by using different kernel functions to handle linear and non-linear relationships.

The SVR from scikit-learn is configured with a Radial Basis Function (RBF) kernel. The RBF kernel is a popular choice for SVR as it can handle non-linear relationships by mapping input features into higher-dimensional space where a linear regression can be fitted.

The SVR model is trained on the training dataset (X_train, y_train) using the fit method. This process involves finding the hyperplane in the higher-dimensional space that best fits the data within the epsilon margin.

Predictions are made on the test dataset (X_test) using the predict method. The SVR model uses the support vectors and kernel function determined during training to predict the output for new data points.

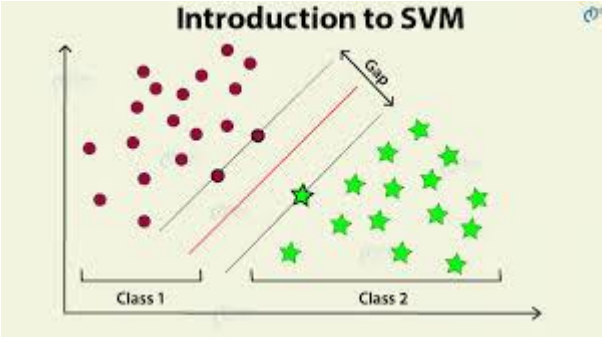

Figure 5: Support Vector Model

E. *Neural Network*

Neural networks provide a robust architecture capable of modeling complex patterns in data through layers of neurons and activation functions. For regression tasks such as predicting the number of followers for Spotify playlists, a neural network can learn non-linear relationships and interactions between features, potentially outperforming simpler linear models.

A Sequential model from TensorFlow's Keras API is used to create a linear stack of layers. This model type is straightforward and well-suited for a stacked general-purpose layer structure where each layer has exactly one input tensor and one output tensor.

The model consists of multiple Dense layers, which are fully connected neural network layers. The term "dense" refers to the fact that each neuron in each layer is connected to all neurons in the next layer.

Layer Setup:

- Input Layer: The first Dense layer is set up with 64 neurons and uses the ReLU (Rectified Linear Unit) activation function. The input_dim parameter is set to the number of features in the dataset (X_train.shape[1]), ensuring that the model can appropriately process the input data.
- Hidden Layers: Two additional Dense layers are included using the ReLU activation function:
    - A second layer with 32 neurons.
    - A third layer with 16 neurons. This architecture helps the network learn more complex patterns in the data.
- Output Layer: The final Dense layer has 1 neuron with a linear activation function, suitable for regression as it can output a range of continuous and linear values that correspond to the predicted number of followers.

The model is compiled using the Adam optimizer, an extension of stochastic gradient descent that has been shown to be effective in practice. The loss function is set to 'mean_squared_error', which is appropriate for regression tasks as it minimizes the squared differences between actual and predicted values, promoting a model that fits the data closely.

The model is trained using the fit method, where X_train and y_train are used as the training data and labels, respectively. Training is performed over 50 epochs with a batch size of 32, and validation data (X_test, y_test) is used to evaluate the model after each epoch. This helps monitor the model's performance and ensure it does not overfit on the training data.

After training, the model uses the predict method to generate predictions for the test set (X_test). This outputs the estimated number of followers for each playlist in the test dataset.

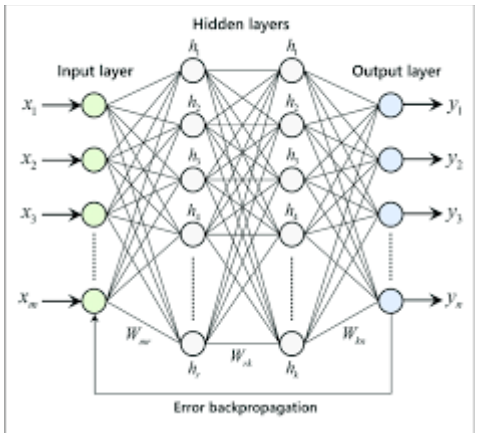

Figure 6: Neural Network Model

III. RESULTS

Considering the dataset quality, after all the preprocessing and data imputation, all models performed relatively poorly.

After training, the model's performance is evaluated on the test set. Key metrics used include:

- Mean Squared Error (MSE): Represents the average of the squares of the errors between what is estimated and the existing data.

- Mean Absolute Error (MAE): Measures the average magnitude of the errors in a set of predictions, without considering their direction.
- R-squared (R^2): Provides an indication of goodness of fit and therefore a measure of how well unseen samples are likely to be predicted by the model, relative to the mean of the target variable.

Here is our observation of the results:

A. Linear Regression Model

The MSE for the model was calculated to be 1084.77, A lower MSE value is preferable as it reflects a closer fit to the data. An MSE of over 1000 suggests that the model predictions are, on average, about the square root of this value (approximately 32.93 followers) away from the actual follower counts, indicating significant prediction errors.

The MAE was determined to be 1.89, An MAE of approximately 1.90 suggests that the model's predictions are, on average, nearly 2 followers away from the actual values.

The R^2 for the model was -0.001. A negative R^2 indicates that the model performs worse than a horizontal mean line of the target variable and suggests that the model does not adequately capture the trends in the data.

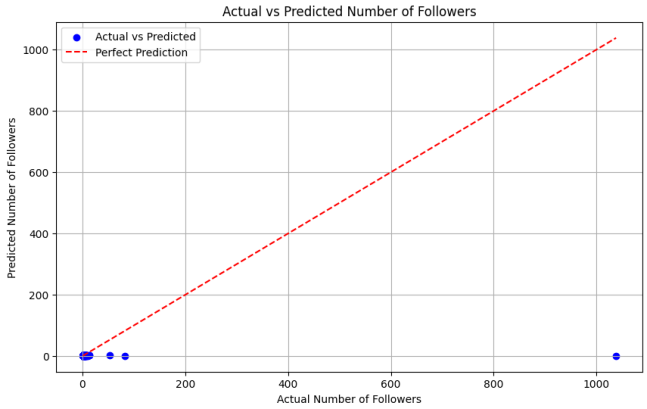

Figure 7: Linear Regressor Output

B. Decision Tree Regression Model

The MSE for the decision tree model was calculated to be 1092.31825. This value is slightly higher than what was observed with the linear regression model, indicating that on average, the squared deviations of predictions from actual follower counts are larger. This suggests that the decision tree model might be experiencing some overfitting, common in decision tree regressions, where the model captures noise in the training data that does not generalize to unseen data.

The MAE was determined to be 2.0795, which is higher than that of the linear regression model. This indicates that the average error in terms of the number of followers is about two per prediction, reflecting somewhat less accuracy in the decision tree model compared to the linear model in terms of absolute values.

The R^2 value for the decision tree was -0.008. Model is likely overfitting to the noise or specific details in the training set.

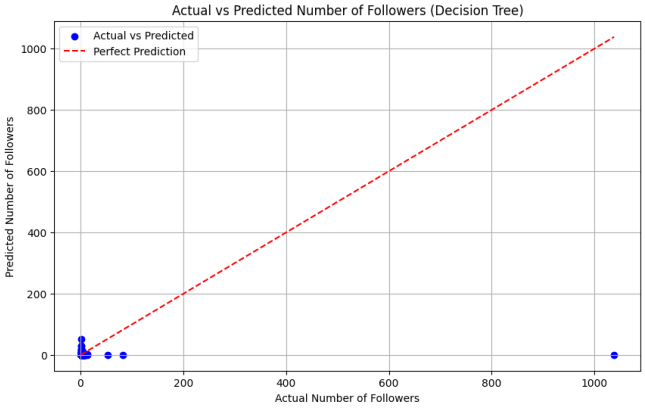

Figure 8: Decision Tree Regressor Output

C. K-Nearest Neighbors (KNN) Regression Model

The MSE for the KNN model was calculated to be 1087.003. A high MSE value like this suggests that the model has a relatively high error rate on the test data.

he MAE was determined to be 1.9, indicating that, on average, the absolute error of the predictions from the actual follower counts is close to 2 followers. Compared to other models, this MAE indicates a relatively moderate level of accuracy, suggesting that the KNN model's predictions are reasonably close to the actual values in terms of magnitude, though not necessarily in precision.

The R^2 value was -0.003. The model fails to adequately capture the variance in playlist followers and hence does not explain the variability in the data effectively.

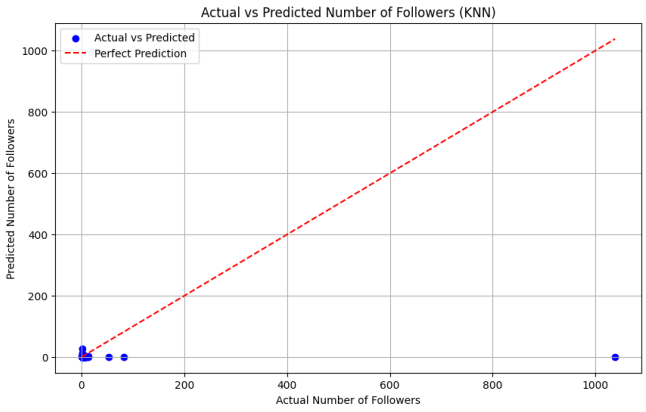

Figure 9: KNN Regressor Output

D. Support Vector Regression (SVR) Model

The MSE for the SVR model was 1085.877. The SVR model's MSE is lower than that of the Decision Tree and KNN but very close to that of Linear Regression, suggesting similar levels of error variance to the latter.

The MAE was recorded at 1.67. This suggests that the average magnitude of the errors in SVR predictions is smaller, indicating that the model's predictions are closer to the actual values, enhancing its practical usability. SVR has the lowest MAE among all models tested.

The R^2 value was -0.002 which is very close to zero and slightly negative. Essentially, this suggests that the SVR model does not perform better than a model that would simply predict the mean number of followers for all inputs.

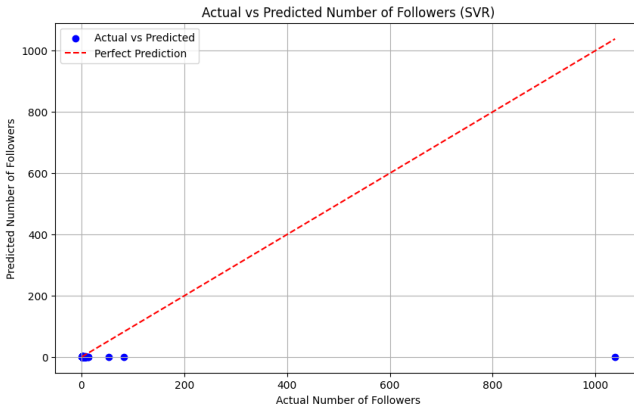

Figure 10: Support Vector Regressor model

E. Neural Network Regression

The MSE for the neural network model was 1086.58. The neural network's MSE is comparable to that of the other models, suggesting that it has a similar level of overall error in terms of squared differences.

The MAE was calculated to be 1.89. The neural network's MAE is very close to that of the Linear Regression and KNN, and slightly higher than SVR, indicating a similar degree of practical prediction accuracy.

The R^2 value was -0.0027

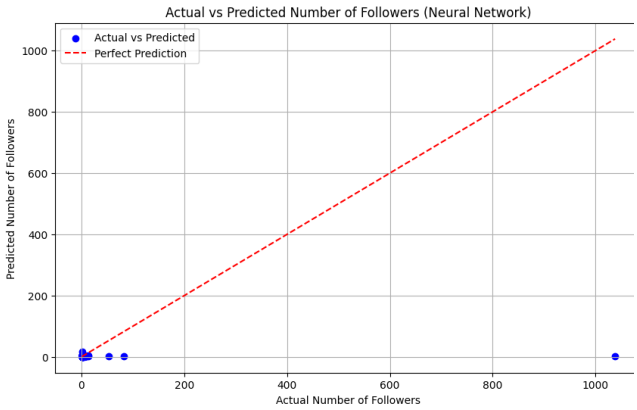

Figure 11: Neural Network Output

IV. CONCLUSION

This project embarked on a comprehensive analysis to predict the popularity of Spotify playlists, using various machine learning models to understand and predict how different features of playlists influence follower counts. The models deployed included Linear Regression, Decision Tree, K-Nearest Neighbors (KNN), Support Vector Regression (SVR), and Neural Networks.

Linear Regression provided a baseline for comparison. While its performance was moderate, it highlighted the limitations of assuming linear relationships in the dataset.

Decision Tree and KNN models did not significantly outperform the baseline, indicating challenges with model complexity and the feature set used.

Support Vector Regression (SVR) showed slight improvements in terms of Mean Absolute Error but still struggled with a negative R-squared, similar to other models.

Neural Networks, while robust and capable of modeling complex relationships, did not dramatically improve the predictions, with results comparable to simpler models

V. FUTURE WORKS

Building upon the initial models developed to predict Spotify playlist popularity, there are several directions for future research and application development that could enhance the utility and sophistication of the analytics tools available for Spotify data. Here are potential projects and enhancements:

- Playlist Similarity and Recommendation System
- Playlist Content Analysis
- User Engagement Prediction
- User Segmentation Based on Playlist Preferences