Song Stream Prediction

A study of predictive models for music streaming trends and analysis

By Dhavalashri Prasad

Introduction

The music industry is constantly evolving, and predicting the popularity of a song has become increasingly important for artists, record labels, and music streaming platforms. As a singer myself, I have always been fascinated by the inner workings of the music industry and the factors that contribute to the popularity of a song. This fascination served as my inspiration to work on this specific project, where I aimed to predict the number of streams a song is going to get based on its characteristics and analyze the trends of songs.

I got this dataset from Spotify API. I used two datasets - the **features** dataset and the **streams** dataset. The features dataset contains information about various characteristics of a song such as danceability, mode, key, energy, duration, and more. The streams dataset provides information about the number of streams in billions. The data is split in such a way that 80% of the data is used for training the model and 20% are used for testing it.

To better understand the data, exploratory data analysis (EDA) was done, where I analyzed relationships between the characteristics of the song such as the relationships between loudness and energy, acousticness and energy, energy and popularity, duration and popularity, valence and popularity, and more.

To predict the popularity of a song, I used various machine learning algorithms such as linear regression, Support Vector Regressor (SVR), Random Forest Regressor, XGBRegressor, Neural Network Regressor, and K-means clustering. I compared the results of all these models. Evaluation of the performance of the models was done using metrics such as R^2 score, Mean Squared Error (MSE), and accuracy.

Overall, this project aimed to provide insights into the characteristics of popular songs and develop a machine learning model to predict song popularity. The findings of this project could be useful for record labels, artists, and music streaming platforms to predict the success of their songs and improve their marketing strategies.

Linear Regression

Ridge regression model using the scikit-learn library in Python is used to do linear regression. Regularization is performed to prevent overfitting and improve the generalization of the model. The regularization parameter alpha is a hyperparameter that controls the strength of the regularization in Ridge regression. By tuning the value of alpha, we can control the trade-off between the fit of the model to the training data and the complexity of the model. A higher value of alpha leads to more regularization, which can help to prevent overfitting, but may also result in a less accurate model. Conversely, a lower value of alpha leads to less regularization, which can result in a more accurate model but may be more prone to overfitting. Therefore, it is important to tune the value of alpha to find the optimal balance between model accuracy and complexity. For a correct value of alpha parameter, the MSE is obtained very low with high training and test accuracies.

```
Mean Squared Error: 0.00013080393655552267
Training Set Accuracy: 0.9991203248624334
Test Set Accuracy: 0.999023295355255
```

Random Forest Regression

Random forest regression model using the scikit-learn library in Python is used. Random forest regression model with 100 trees and random_state parameter is set to 42 is used to ensure that the random number generator used in the model is initialized to a fixed value, which ensures reproducibility of the results.

Random forest regression is an ensemble learning method that constructs multiple decision trees and aggregates their outputs to make predictions. The number of trees in the forest is controlled by the n_estimators parameter. The random forest algorithm has several advantages over other regression algorithms, such as the ability to handle both continuous and categorical features, and resistance to overfitting. However, tuning the hyperparameters of a random forest can be a complex task, and it is important to carefully select the appropriate values of the hyperparameters to achieve the best possible performance.

```
Mean Squared Error: 0.006546288629999873
Training Set Accuracy: 1.0
Test Set Accuracy: 0.9666459684117473
```

The output shows that the model has a relatively high MSE but a perfect accuracy score on the training set and a high accuracy score on the test set, indicating that the model may have overfit the training data but generalizes well to new data.

Support Vector Regressor (SVR)

Support vector regressor model using grid search for hyperparameter tuning is performed.

‘C' is the regularization parameter that controls the trade-off between the margin width and the number of misclassifications. A large value of 'C' indicates a low regularization, meaning the model will have a smaller margin and will be more likely to classify all training examples correctly. A small value of 'C' indicates a high regularization, meaning the model will have a larger margin but may allow more training examples to be misclassified.

'gamma' is the kernel coefficient for the radial basis function (RBF) kernel. It determines the shape of the decision boundary and how much influence a single training example has. A small value of 'gamma' indicates a large similarity radius, which can result in a smoother decision boundary. A large value of 'gamma' indicates a smaller similarity radius, which can result in a more complex decision boundary that fits the training data more closely.

By performing a grid search over a range of values for 'C' and 'gamma', the algorithm can find the combination of hyperparameters that results in the best performance on the given dataset.

```
Mean Squared Error: 0.0044647831853250001
Training Set Accuracy: 0.9598649060706137
Test Set Accuracy: 0.9666617489525248
Best hyperparameters: {'C': 15, 'gamma': 15}
```

XGBRegressor

Hyperparameter tuning for an XGBRegressor model using GridSearchCV is performed. GridSearchCV is used to perform a search over the hyperparameter grid, with 5-fold cross-validation (cv=5) and verbose output (verbose=1). The **learning rate** hyperparameter controls the step size at which the gradient boosting algorithm iteratively fits new trees to the residuals of previous trees. A smaller learning rate can improve the stability of the model but may require more trees to achieve the same level of performance. The **max-depth** hyperparameter controls the maximum depth of each decision tree in a random forest or gradient boosting model. The **colsample_bytree** hyperparameter controls the fraction of features used to fit each tree in a gradient boosting model. A smaller colsample_bytree can reduce the risk of overfitting but may require more trees to achieve the same level of performance. The **reg_alpha** hyperparameter controls the L1 regularization applied to the weights of the features in a gradient boosting model.

```
Fitting 5 folds for each of 1215 candidates, totalling 6075 fits
Best hyperparameters:  {'colsample_bytree': 1.0, 'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 200, 'reg_alpha': 0.0001, 'subsample': 0.8}
R^2 score on test set:  0.9966845351822086
```

The grid search is performed using 5-fold cross validation and tries 1215 different combinations of hyperparameters. The R^2 score on the test set is 0.9966845351822086, which indicates that the model is able to explain 99.7% of the variance in the test set. This is a very high R^2 score, suggesting that the model is performing well on the data.

Neural Network Regressor

Hyperparameter tuning for a neural network model built using Keras is performed. The input data is first normalized using the **StandardScaler** from scikit-learn. This is done to ensure that all features have the same scale, which can help improve the performance of the model. Then, a function is defined to create the model object with the desired hyperparameters, which includes two hidden layers with 64 units each and an output layer with one unit. The optimizer used is **Adam**, and the loss function is mean squared error (MSE).

The hyperparameters that are being tuned include **batch size, number of epochs, and learning rate**. The batch size is the number of samples used in each training batch. The number of epochs is the number of times the model is trained on the entire dataset. The learning rate determines the step size taken during optimization. These hyperparameters can have a significant impact on the performance of the neural network model. For example, a smaller batch size can lead to faster convergence and better generalization, but may increase the training time. A larger number of epochs can improve the performance of the model, but may also increase the risk of overfitting. The learning rate determines the step size taken during optimization and can impact the speed of convergence and the quality of the solution.

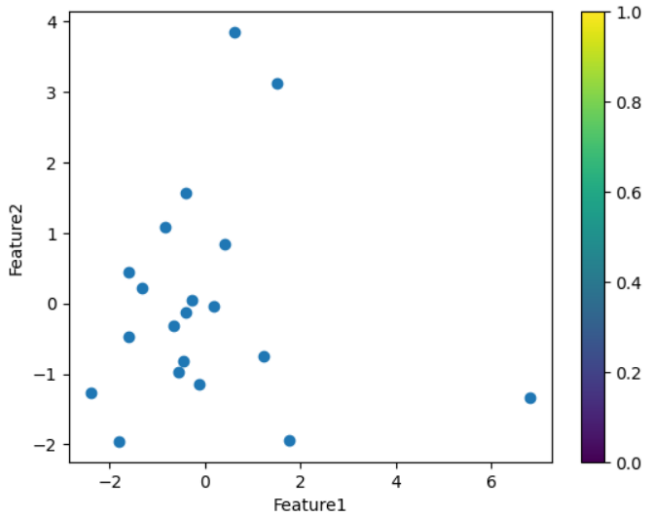
```
Best Parameters: {'batch_size': 8, 'epochs': 150, 'learning_rate': 0.01}
R^2 Score: 0.9972969806931828
```

The best parameters found were a batch size of 8, 150 epochs, and a learning rate of 0.01. The R^2 score on the test set was 0.997, which indicates that the model fits the data well and can make accurate predictions.

The StandardScaler was used to normalize the input data by scaling the features to have a mean of 0 and a standard deviation of 1. This preprocessing step is important to ensure that the features are on a similar scale, which can improve the performance of the model.

K-means clustering

Principal Component Analysis (PCA) is used to transform high dimensional data into a lower dimensional feature space while retaining the important information. It does this by identifying the directions of maximum variance in the data and projecting the data onto a lower-dimensional subspace while retaining as much of the original information as possible. This helps to reduce the computational complexity of models while still preserving the significant features of the original data. PCA is often used as a preprocessing step in machine learning pipelines to improve the performance of models. So, PCA is performed on the data to reduce the dimensionality of the feature space.



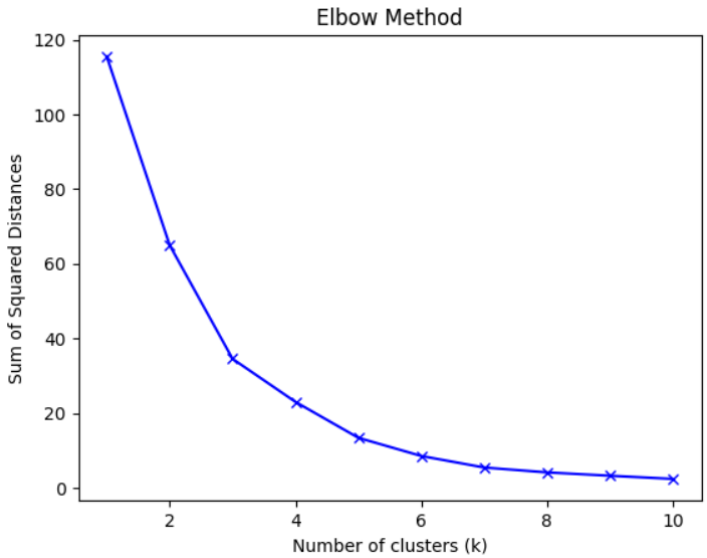
K-means clustering is performed on the dataset using the two principal components obtained through PCA. The objective is to determine the optimal number of clusters to use for the data.

Elbow point

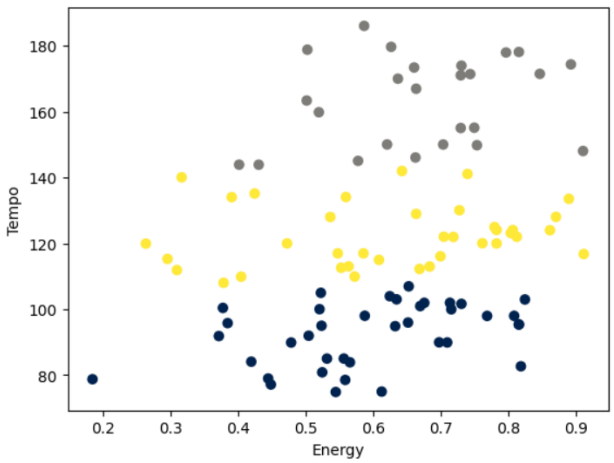
In k-means clustering, the elbow method is a heuristic technique used to find the optimal number of clusters in a dataset. The elbow point is the point of inflection on the curve that represents the sum of squared distances between each point in a cluster and the centroid of that cluster, as a function of the number of clusters k.

The idea behind the elbow method is to select the k value at which the decrease in the sum of squared distances between each point and the centroid slows down significantly, creating an "elbow" shape in the curve. This k value is considered the optimal number of clusters for the dataset.

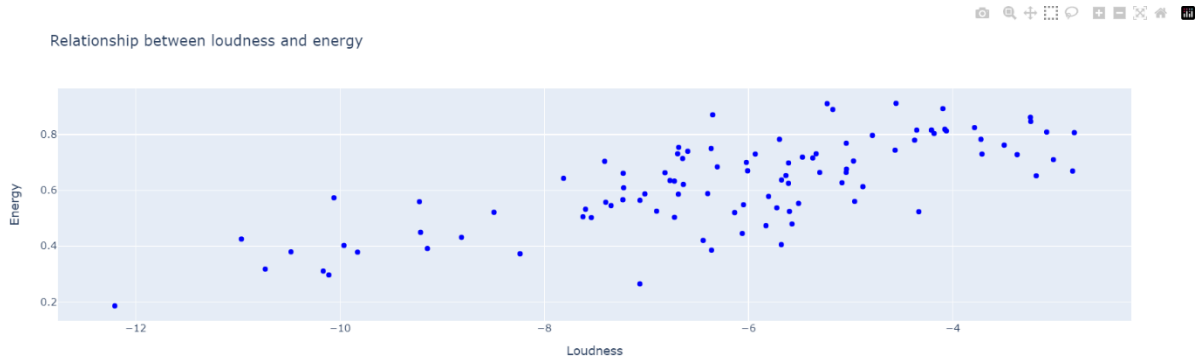
To compute the elbow point, we first run the k-means algorithm for a range of k values, typically from 1 to 10, and record the sum of squared distances for each k value. Then, we plot the k values against the corresponding sum of squared distances and look for the point where the curve starts to flatten out.



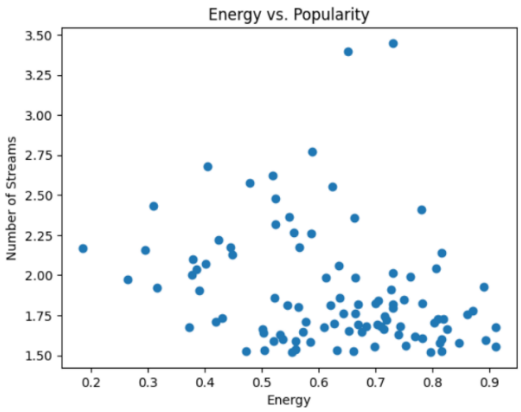
Number of clusters are determined using this elbow method. We can see that the graph starts to level off when the corresponding value on the X-axis is 3. So, clusters based on three features: energy, tempo and streams, are formed.



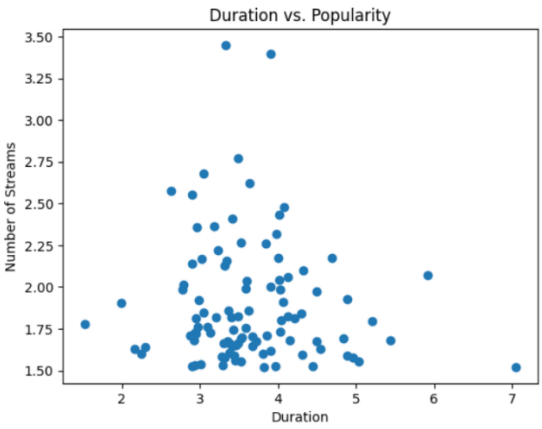
Exploratory Data Analysis



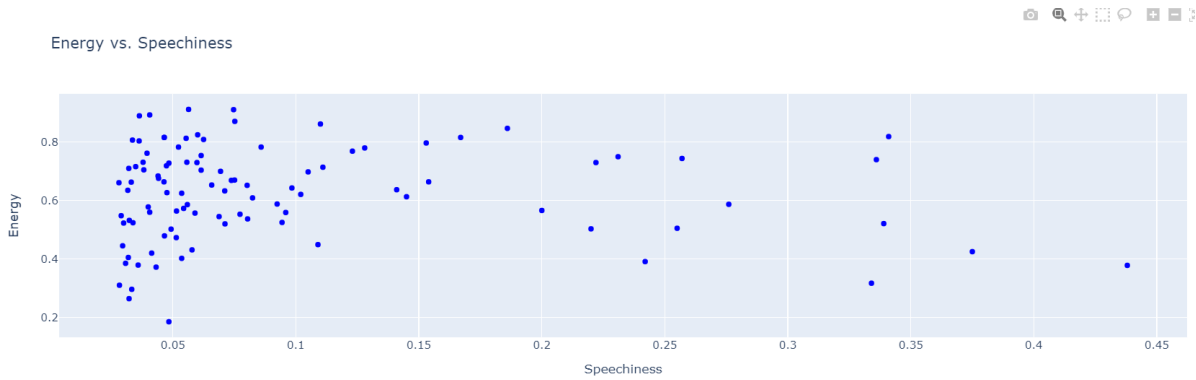
The reason why energy may increase with loudness is that louder sounds tend to be associated with higher levels of excitement or intensity. In music, this can manifest in the form of increased temp o, more complex harmonies, and more dynamic variations in volume and texture. These characteristics can contribute to the perception of a song as having higher energy, even if the actual energy level of the audio signal remains constant.



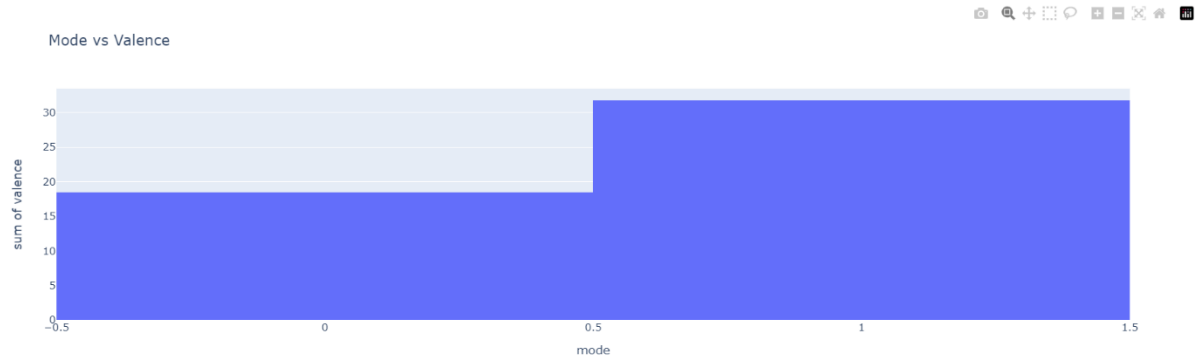
When a song has high energy, it can create a sense of excitement and momentum that can be infectious for listeners. This can make the song more memorable and enjoyable, and can encourage people to share the song with others.



Songs of duration in range of 3-5 minutes are streamed more.



Sometimes there are songs which have dialogues in-between them. There's minimal music during this part. This characteristic of a song is not very desirable by the audience. Hence when the speechiness is high in a song, the energy of the song is low.

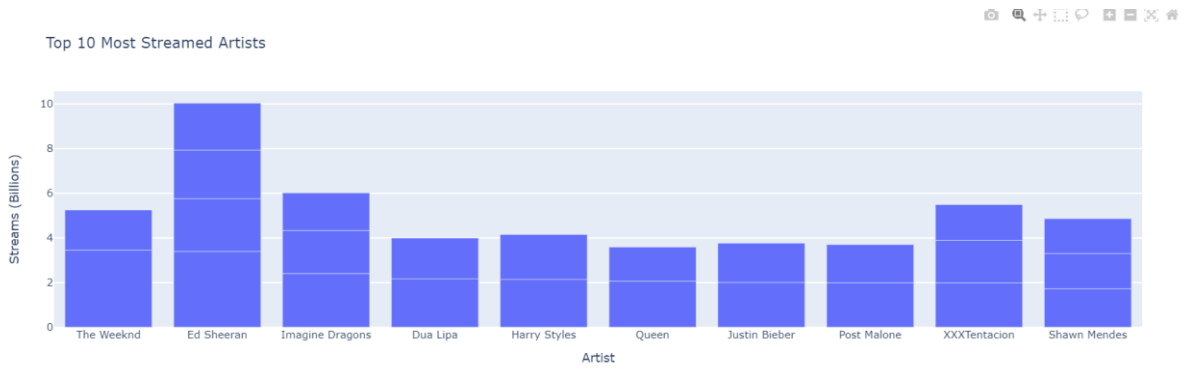


In music, there are 12 different scales that are used in composition. Half of these scales are classified as major scales, while the other half are classified as minor scales.

Major scales are often associated with a positive or happy vibe, while minor scales are often associated with a more negative or sad vibe. This is because major scales are typically composed using notes that create a sense of resolution or completeness, while minor scales often include notes that create a sense of tension or unease.

To analyze the mood or valence of a song based on its scale, we can assign a value of 1 to songs composed with a major scale and a value of 0 to songs composed with a minor scale. By comparing the valence scores of songs associated with major and minor scales, we can see that songs composed with major scales generally have higher valence scores than those composed with minor scales. This suggests that listeners may perceive songs composed with major scales as more positive or uplifting

than those composed with minor scales.



This graph is showing the top ten artists streamed on Spotify and the number of streams their songs have, in billions.

Conclusion

Based on the R^2 scores on the test set, the neural network regressor model performed the best with a score of 0.9973, followed closely by the XGB Regressor model with a score of 0.9967. The SVR model also performed well with a test set accuracy of 0.9667, but the linear regression and random forest regressor models had lower test set accuracies of 0.9990 and 0.9666, respectively.

The neural network regressor model could have performed the best due to its ability to learn complex non-linear relationships between the features and the target variable. Neural networks have the ability to learn from the data and automatically extract features that are relevant for making accurate predictions.

Additionally, hyperparameter tuning can play a crucial role in the performance of a model, and it's possible that the hyperparameters for the neural network were tuned to better fit the data and improve its predictive ability.

Overall, the neural network regressor model outperformed the other models in terms of R^2 score, indicating that it was able to explain the variation in the target variable the best. The XGB Regressor model was also able to explain a significant amount of the variation in the target variable.

What was difficult for me

Hyperparameter tuning is a crucial step in machine learning as it involves finding the optimal values of hyperparameters that maximize the performance of a model. Hyperparameters are parameters that are not learned by the model during training but are set before training begins. These parameters determine the architecture of the model, the optimization algorithm, and the regularization techniques used to prevent overfitting.

Finding the optimal hyperparameters is a difficult task and often requires a lot of trial and error. It involves selecting a range of values for each hyperparameter and training the model with every combination of values using a technique like GridSearchCV or RandomizedSearchCV.

In my project, hyperparameter tuning was likely the most challenging aspect of implementing machine learning models. It required me to have a deep understanding of the models and their hyperparameters, as well as the dataset I was working with. I had to carefully select the range of values for each hyperparameter and experiment with different combinations of values to find the optimal set of hyperparameters that maximized the performance of the model.