Carlos Gonzalez

Predicting the popularity of a song and clustering different genres together

**Introduction**

Throughout time many people have aspired to become popular musicians to no avail, this project would be able to help classify what makes a popular song. On the other hand, differentiating the genres of songs using their features allows for the development of the Spotify app because it allows the app to choose what songs belong to what genre with minimal interference from a person. This is all-important because it allows for the advancement of music apps in general such as Apple Music, Pandora, and Spotify. Each song in the data set has different features such as acousticness, artists, and what key the song is played in. This dataset there includes various files such as data.csv which contains the data given above and will be used the most. The first part of my project revolved around creating a model that will predict whether a song will become popular based on the features given in the dataset. This is something that is done a lot throughout various aspects of not only the music industry but also the business side of things. For example, on the commercial side of things there exists companies that want to only advertise songs if they have the capability of becoming a hit. On the financial side of things, companies will prefer a song that will increase their overall commissions. Based on the musical side of things, companies such as Spotify want to help push a song that will be a hit and be exclusive to Spotify’s streaming service. A prediction model that I am talking about will more suffice in helping them make choices that will make the business the most money. The second portion that I decided to do was more of a classification problem and creates main genres and clusters the sub-genres into these new main genres. The purpose of this is that it would allow for the new genres to be similar to the very specific genres given. This type of clustering proves a lot of purposes and allows for the data to be manipulated in various ways for example it could help create a recommendation system based on the genre of music the user prefers even if their preferred genre is obscure. In a summary, I took care of everything that needed to be done such as data cleaning, data preprocessing, exploratory data analysis, training the machine learning model, and testing the machine learning model. All of this will be explained more in-depth in the next section.

**Approach**

My first step was to look through the data using exploratory data analysis. I was looking at the features that had a strong correlation to the popularity of a song. I decided to drop some columns from the dataset because they were not helpful to our models. The ID column was simply a unique identification string given to a song. This means every row in ID is unique and in no way helps in any model needed. Another column that was dropped was the name column, this column gave us the name of the song and was not needed. The other column that was dropped was the release date column, this column was eliminated because it was like the year column, so rather than having a column of strings I decided that having a numerical column to represent when a song came out was preferable. I checked for duplicates and null values. I removed duplicates and luckily found no null values in the dataset. I then created a heatmap representing the Pearson correlation matrix to see the correlation between the numerical features to get a brief look at what features were heavily correlated with popularity. I then checked the top 20 artists from the dataset, there was no reason for this section but rather it was for fun and I was simply curious. My next step was to look at features with the highest absolute value Pearson Correlation Coefficient by doing this I figured out that the features most associated with popularity are year, acousticness, loudness, energy, instrumentalness, and speechiness. With those features, I created their distance plot to see how the data was laid out and if something needed to be normalized or fixed. The second step I took was differentiating genres. For this part, I left the data as default and normalized except for the year feature using the MinMaxScaler. I then used the elbow method for clustering algorithms to figure out what would be the best k for the k means clustering algorithm that will be performed on the dataset. To compare what value of k is better, I used the Sum of Squared Error and found that the best k is 25. One thing to note is that I only used a sample of 1000 when checking for the best value of k because I lacked the computational power to check up to the k value of 200 for the whole dataset. I then used 25 as my k value to see how many songs would fit in each genre and which genre has the most songs in them. The next step I took was a little longer and this revolved around my goal of making a decent prediction model. At this point, I had already cleaned the data and all I did, in the beginning, was split the data into its training set and testing set. My respective y\_train and y\_test were simply the column that held the popularity of a song. I then decided that I needed to preprocess my data. My first step was to transform the name of the artist into some numerical identifier. This is different from the ID column that I dropped because the ID column was unique for every song and I want an identifier based on the name of the artist. I then had to transform the data in tempo and instrulmentalness. Next, I had to use the OneHotEncoder to encode the categorical features because they were going to be used in a regression model, which requires numerical features. I then applied the MinMaxScaler to the artists, duration\_ms, loudness, and tempo columns. I then finally normalized the popularity value by dividing it by 100. My next step was to test the three chosen models. My first model was the linear regression model and I created two models. One model only held the features with those who had a correlation coefficient above .2 and one that used all the features. I then calculated the root mean square error of the testing model and the training model. My next step was to create a K nearest neighbor model, I had the idea of checking the best k value with a range from 1 to 200 but I lacked the computing power to do this in a reasonable amount of time, so I simply just checked from 1 to 100. I then calculated the best RMSE given from the K-Nearest Neighbor Model. The next model I created was a decision tree model. I trained two different types of models one was a single-run decision tree and the other was a loop decision tree which had a range of 2 to 200 max-leaf nodes. The single-run tree had a max of 41 leaf nodes. I then checked which max number of leaf nodes produced the best RMSE.

**Results**

Exploratory Data Analysis

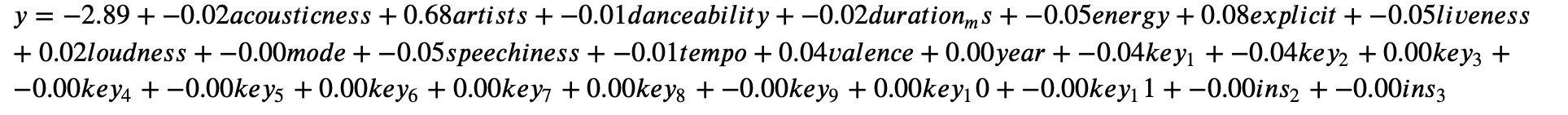
Graphical user interface

Description automatically generated

This is the numerical heatmap that I used that allowed me to look at what features correlated to popularity the most.

**Part 1 Prediction Model**

Linear Regression Model

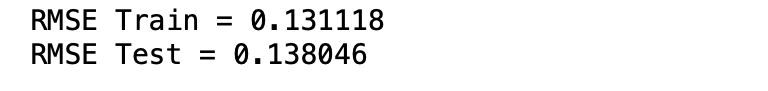


Chart, scatter chart

Description automatically generated

RMSE for Linear Regression using features with a correlation coefficient at or above 0.2A picture containing text

Description automatically generated

RMSE for Linear Regression that uses all Features. 

As you can tell one of the biggest factors when it comes to the popularity of a song is the artists. Popular artists will more than likely produce a popular song. The first figure allows us to see how exactly each feature interacts with each other in determining if a song will be popular or not. As one can see the RMSE for this linear regression is way better than the linear regression model that only uses features with a correlation coefficient above 0.2.

K-Nearest Neighbor Model

Chart, line chart

Description automatically generated

This graph shows the value of k as the independent variable and RMSE as the dependent variable. As shown the testing set has a stable RMSE for each value of K, but the same cannot be said for the training set. As the number of k gets bigger the RMSE also gets larger. A smaller RMSE is wanted but in this case, there is an issue that when the value of k is small then the training set’s RMSE is small, but the testing set’s RMSE is bigger which is not desirable.

Decision Tree

Chart

Description automatically generated

Chart, scatter chart

Description automatically generated

Table

Description automatically generated

Above I have the figures and tables that belong to the loop decision tree, or in other words, looks for the best number of max leaf nodes. As shown, the higher number of max-leaf nodes goes the bigger the difference of RMSE exists between the training and testing set which is not desirable. We are looking for somewhere in which the difference is not that large. This desirable number of max-leaf nodes was found at 178 and had a training RMSE of 0.103 and a testing RMSE of 0.112.

All of this led me to the conclusion that the best model to predict when a song would be popular is the decision tree model.

Differentiating Genres

Chart

Description automatically generated

Chart

Description automatically generated

Using the elbow method I was able to deduce that the best k value, in this case, would be 25. Upon using 25 as K I was able to cluster songs into 25 genres and see what genre had the most songs and one can determine something from it. For example, considering this day and age it can be inferred that Genre 3 most likely encompasses the top 40 genres or in other words pop music because that is what most people listen to or will hear at some point in their lives.

**Conclusion**

This project was quite long and required a lot of steps. At first, I performed Exploratory Data Analysis to look at the features inside of the dataset and to see what type of preliminary work would be needed. Upon looking at the features I noticed that there existed certain columns that would not help create models. Those features would simply just be a hindrance and would require a lot more preprocessing and serve very little purpose to the model. Then I noticed I had to do some data preprocessing to not only the numerical features but also the categorical features to be able to use everything in the regression models. I then created various models that would help predict the popularity of a song such as a linear regression model that took all the features in and another that only uses features that have a correlation coefficient of 0.2 or higher. The next model that I used was the K-Nearest Neighbor Model. During this step, I used the K-Nearest Neighbor Model with various values for K and checked each Root Mean Square Error to evaluate what the best one would be. After this was done, I created two different decision tree models, one single run tree and a loop tree that would check the tree using different max leaf nodes. Overall, for this part, it was pretty noticeable that the decision tree ended up with the best results at max\_leaf\_nodes = 178 and test\_size = 0.2. The next portion was using the K-means clustering algorithm to help differentiate genres. I used various values for K and came up with the value of 25 as being the best and led to the best clustering. I then created a graph to show which genres had the most songs in them. Overall, I feel as though this project gave me the chance to use a large dataset and teach me more about different models and their uses. For example, I was not aware that KNN could be used as a regression also, I simply thought it could only be used for classifying problems.

**Acknowledgements**

Kaggle Dataset:

<https://www.kaggle.com/yamaerenay/spotify-dataset-19212020-160k-tracks/tasks>  
Links I used:

<https://www.kaggle.com/yamaerenay/spotify-dataset-19212020-160k-tracks/tasks?taskId=2173>

<https://scikit-learn.org/stable/modules/tree.html>

<https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/>

<https://en.wikipedia.org/wiki/Decision_tree>

<https://www.geeksforgeeks.org/decision-tree/>

<https://www.geeksforgeeks.org/k-means-clustering-introduction/>

<https://www.geeksforgeeks.org/k-nearest-neighbours/>

<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html>