Name: Elliott Rose
ID: 300540768

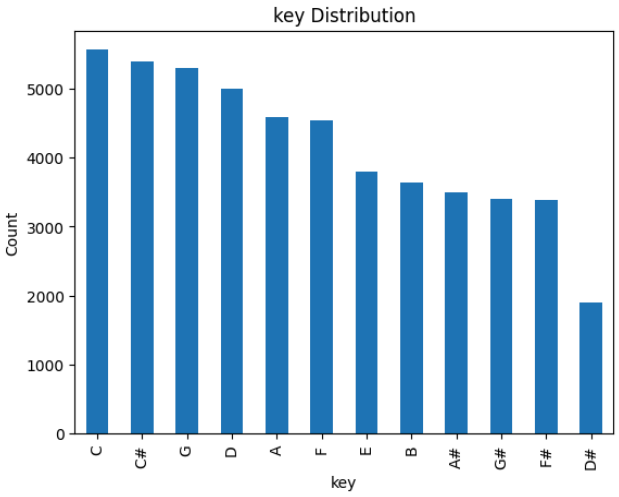
CORE:

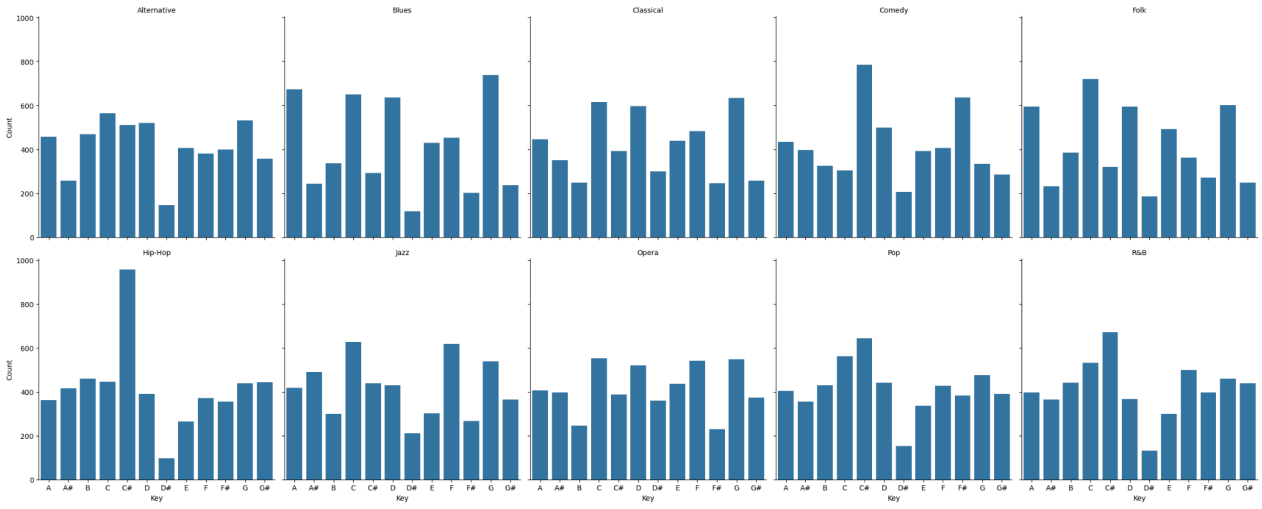
To start off with I used standard data exploration techniques to explore and find noteworthy correlations in all the separate data sets. Initially I briefly read over the CSV files to gain a surface level understanding of what I was dealing with. Once looked over I then combined them into one singular dataframe, conveniently they all have a "genre" category in each of the individual csv files. Using the .head(), .describe(), and .value_counts() functions I was able to get a deeper understanding of each genre's contents.

The head() function shows the first 5 rows of the data which allows me to see what features a song is composed of. This enabled me to test out the relationship between different features to find patterns and correlations. The describe() function then allowed me to see the general statistics for all the columns in the combined dataset. value_counts() summarized the count of each unique value of a feature.

Once I had used these data exploration techniques I was able to find the following correlations:

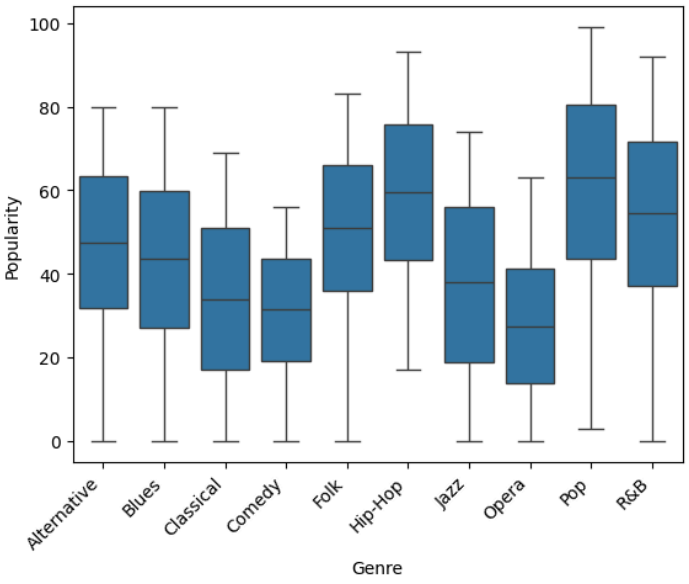
1) My first noteworthy find while exploring the data was the frequency of specific keys. Shown on the right is a bar graph of the key frequency within all genres. The D# key was substantially less common than the rest of the keys. The other keys all followed a linear descending pattern with the C key being the most common. It is also interesting to note that the C# key, while being quite common, has a massive spike in the hip-hop genre. C# also has a notable spike in the pop genre, while not as severe as hip-hop's spike, it is still noticeable. It's evident that the distribution of keys varies by genre. This could prove useful in predicting the genre. For example, if an instance had a key of C# It is very likely to be hip-hop.



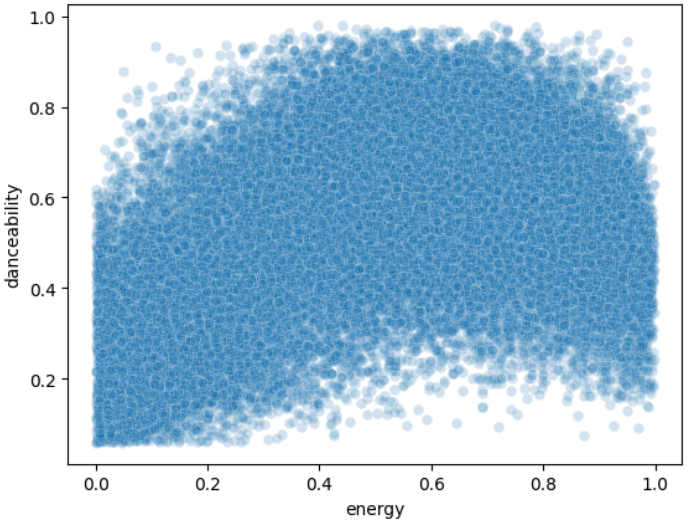


2) Secondly, there appears to be a correlation between genres and popularity of an instance. The data suggest that the genre of an instance influences its popularity, to an extent. This indicates that the popularity of an instance could be useful in predicting its genre. For instance in the data shown to the right, if the instance has a high popularity then its most likely going to be one of three genres:
    a) Pop (most likely)
    b) Hip-hop
    c) R&B

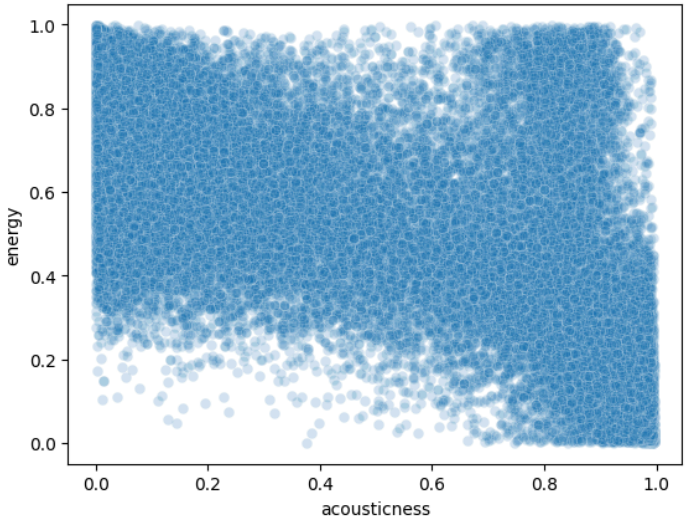
    In the opposite way using this, if we were to find that the instance was of very low popularity. Then there would be a very likely chance it could be of the opera genre.



3) While comparing the energy and danceability I noticed they seem to be directly intertwined. Either end of the energy spectrum seems to result in a lower danceability rating, leaving a perfect middle ground range. Between 0.5 and 0.6 energy danceability reaches its peak, however as shown on the graph to the right it quickly drops off. The pattern follows an inverse parabolic shape, where instances closer to 0 or 1 have on average a lower danceability range.



4) Finally I explored the correlation between acoustic-ness and energy. shown in the chart to the right, the pattern seems to show an inverse linear relationship. The higher the acoustic-ness the lower the energy and vice versa. In saying that however, there is a cluster of outliers in the top right that defy this pattern. These instances seem to retain their energy despite having a high acoustic-ness. Upon closer inspection, I found that this cluster is composed mostly of the comedy genre. This correlation could help predict the genre of an instance, as determining the significance of each feature individually may prove to be challenging.



COMPLETION:

Initially when making my system I decided to split the training data into an 80/20 ratio (training/testing) which I think was sufficient. My reasoning for doing this despite being already given testing data is that the testing data had no genres within the dataset. Thus by simulating my own testing set (genres included), I allow myself to test my model locally. This is important as Kaggle only allows for 5 entries a day, so this way I'm unrestricted in my testing abilities.

The initial plan was to use KNN to classify instances based on their nearest neighbors. I initially believed this would be effective because I had observed correlations between features in previous tasks, suggesting that instances with similar feature values would be correctly classified by KNN (which relies on similarity). After further research into various models however, I decided to use the Random Forest Classifier (RFC). While KNN would be suitable for this task (especially for low-dimensional datasets), I preferred a more robust and scalable model. RFC offers better scalability, computational efficiency, and uses ensemble learning to capture key relationships in the data.My initial RFC model used the basic configuration with default hyperparameters, except for setting the random state to 42. This resulted in an accuracy of 60.1% on the test portion of the training data. I started with the default setup to establish a baseline accuracy for RFC.Next, I reflected on insights from the previous task to explore how they could improve genre classification using RFC.

My first finding, that genre is related to the key of an instance, led me to adjust the n_estimators and n_features hyperparameters in the RFC. n_estimators control the number of decision trees, while n_features sets how many features are considered. This was important because adjusting the number of trees helped capture the key-based pattern, and tweaking the number of features allowed me to emphasize the "key" feature in predicting the genre.

The second key finding showed a link between genre and popularity, with some genres being more popular on average. This led me to focus on the max_depth hyperparameter, which controls the depth of each tree in the forest. By increasing max_depth, the RFC can better capture how popularity influences genre.

Finding 3 reinforced the need to adjust the max_depth hyperparameter, as the relationship wasn't linear. The inverse parabolic shape between energy and danceability suggests a more complex pattern. By increasing max_depth, the model can better capture these complexities and any deeper connections between genre and popularity that may be more intricate than expected.

In relation to finding 4, which showed the link between energy and acousticness, I adjusted the min_sample_split hyperparameter. This change prevented the model from making overly detailed splits that capture noise, allowing it to focus on meaningful splits that highlight differences, especially those that deviate from the pattern in finding 4.

I also adjusted the max_samples hyperparameter, which controls how much of the training data is used to train each subtree in the RFC ensemble.

Adjusting the hyperparameters did improve the model's accuracy by about 5%, however once submitted to Keggle it was brought down marginally. I tested different values to find the final ones for my submission and this is what they ended up as: n_estimators = 225, max_depth = 25, max_samples = 0.9, max_features = 3, and min_samples_split = 35. This resulted in an improvement from 60.1% to 65.14% (local), and 64.92% (Keggle).

CHALLENGE:

The Random Forest Classifier (RFC) is more challenging to interpret compared to simpler models like KNN because it involves multiple decision trees and complex feature interactions. While RFC improves accuracy and scalability, it’s harder to understand why certain predictions are made, which could lead to trust issues or difficulties in deployment. Users may be unsure of how the model reaches its decisions, which could affect their confidence in it.

KNN is much easier to interpret since it relies on the proximity of instances based on feature similarity. However, KNN might not capture complex relationships in the data as effectively as RFC, especially in higher dimensions or with non-linear patterns. While KNN is simpler, it may underperform in capturing intricate relationships like those related to genre and popularity.

Though RFC can capture intricate relationships in the data (like the ones between genre, key, and popularity), it still falls short of providing clear explanations for predictions. In simpler tasks, RFC might be overkill compared to more interpretable models like KNN.

Compared to even more complex ensemble methods, RFC strikes a balance. It's more interpretable than methods like boosting, though still more opaque than simpler models like KNN. However, more complex ensemble methods (e.g. Gradient Boosting) might provide even better performance at the cost of increased interpretability challenges.