Joshua Taylor, 261180779

Initial Project

The original project I had chosen was to work with the *Spotify Recommendation Dataset* from Kaggle, created by Brice Vergnou in order to create a basic music recommendation system using a K-nearest neighbors classifying algorithm. The results will be based on what is "liked" and "disliked" in the dataset. The end goal for this project inside of the MAIS 202 bootcamp timeframe will be to create a simple website where someone can list 5 songs they like and then receive a recommendation for a sixth based on what they gave.

Data Preprocessing

I am working on the Spotify Recommendation dataset created by Brice Vergnou, which is posted on Kaggle. This dataset contains 13 features listed across 197 songs, as well as the "liked" label. The 13 features are danceability, energy, ket, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, temp, duration_ms and time_signature.

The only change that I made to this dataset before beginning to fit my KNN classifier was to scale the features normally. I didn't have to clean the dataset in any other way.

Machine Learning Model

I used SKLearn to implement my library. More specifically, I used the KNeighborsClassifier from sklearn.neighbors, StandardScaler from sklearn.preprocessing, and GridSearchCV from sklearn.model_selection.

I decided that because my dataset is relatively small I shouldn't split my dataset into a training set and a testing set, so I used cross validation. I tried K values in the range of 1 - 50 to make sure that I chose the best possible value for K, which turned out to be 39.

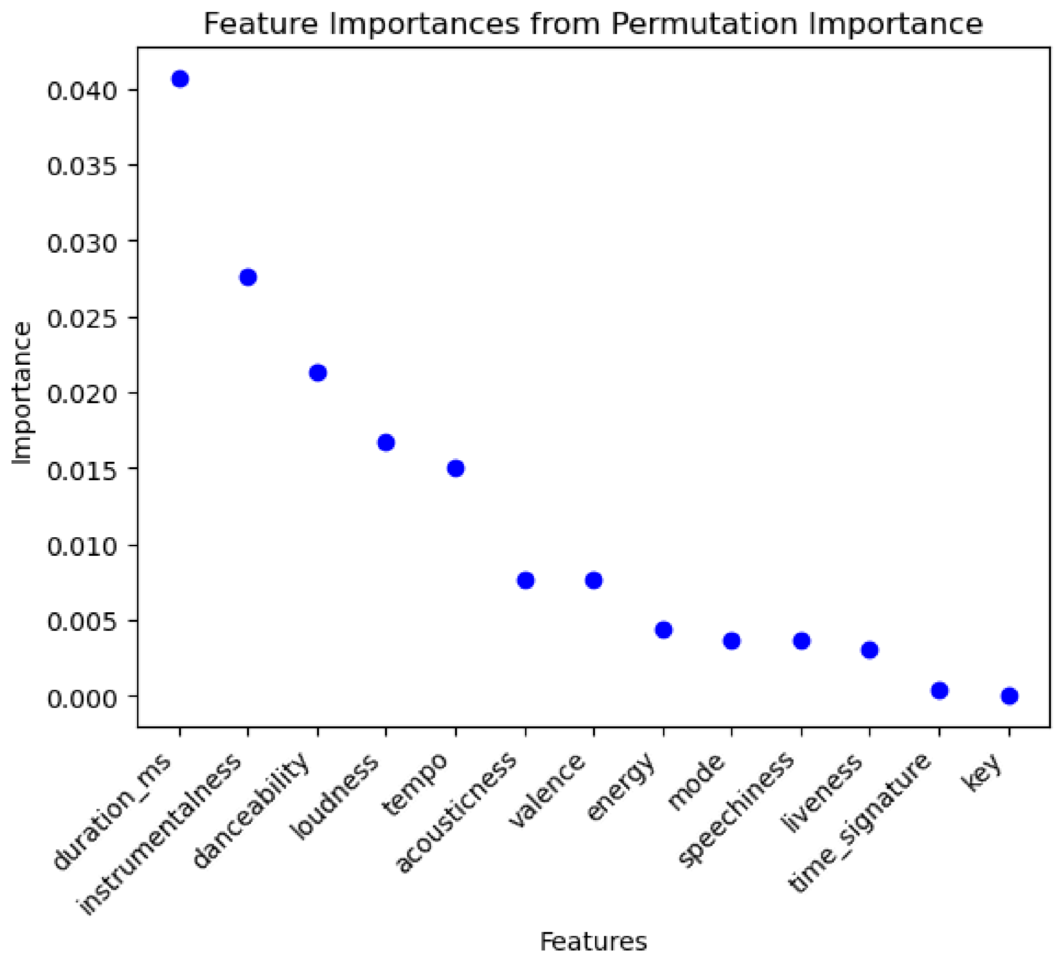
The main challenge that I faced was getting errors from sklean, as I didn't understand all of the syntax. After more research I was better able to use the library.

Preliminary Results

Given that my goal is to build a model that will recommend good songs, I decided that recall was the best metric to use. The stakes in a music recommendation system aren't high, so I think it's better to minimize false negatives (which represent the case where my model fails to recommend a "good song") rather than avoiding false positives (where my model recommends a "bad song"). As stated earlier, I used sklearn's GridSearchCV to perform cross validation of 5 folds for all values between 1 and 50 in efforts to find the optimal value for the K hyperparameter.

Given that I used a dataset that was tailored to someone else's music taste, it is impossible to gather more data to test this model because there is no way to decide if a song should be labeled as "liked" or not. However, the model I built can be analyzed to inform future iterations of this model. Namely, I can analyze which feature in the dataset is most impactful.

To do this, I used sklearn.inspection's permutation_importance. I used it to build a new dataframe that organized each feature according to it's importance to the results. The graph below demonstrates those results.



I was surprised by the results. duration_ms simply represents the length of time a song takes in milliseconds. Although I wouldn't have guessed it, it does make sense– a song that is far too long or far too short may significantly impact someone's taste for it. As for the opposite end of the results, it makes sense that key has little impact because any song can be enjoyed and appreciated in almost any key.

Next Steps

The main flaw in my approach is using such limited data. Although my KNN model works fine, it doesn't have any real-world application yet. That's why I would like to expand on this mode in future iterations by trying to find a bigger dataset or build one using spotify's API. I would like to include songs that are the most popular in the world because I am intrigued by the idea of using my model to predict how popular a song is against Spotify's popularity charts. I would also like to move past the KNN algorithm and move towards something more intricate, such as a neural network.