Main Documentation

I. The dataset used in this project is the Spotify Tracks Dataset, containing information on 114,000 songs collected from Spotify. It includes metadata such as track_name, artists, album_name, and track_genre, along with detailed audio features like popularity, danceability, energy, tempo, valence, loudness, and instrumentalness. Other attributes include explicit (whether the track contains explicit content), key (musical key), mode (major or minor), duration_ms (track length), and time_signature.

II. We wanted to explore what auditory features influenced the popularity of a song on Spotify. We wanted to see if we could predict if a song was "popular" or "unpopular".

III. To address the problem of figuring out what key variables affect a song's classification of "popular" or "unpopular", random forest was the best methodology when looking at performance results and accuracy in its predictions. First off, it achieved a very high accuracy of 97.98%, the highest among the models trained. In addition, the AUC score for the Random Forest model was 0.788, indicating that it was the best model to differentiate between "popular" and "unpopular" songs. In addition, the model is very robust to overfitting as it utilizes bootstrap aggregation and trains trees on slightly different datasets. This reduces the variance of the individual trees so the final model generalizes better to unseen data. Finally, the random forest model gives us key insights into feature importance. This is done by calculating how much each feature reduces impurity, or GINI index, across splits in the decision trees. We see that tempo and loudness influenced popularity most, at 12.3% each. Instrumentalness was least important at 8.2%.

IV. Logistic Regression achieved an accuracy of 95.75% but struggled with class imbalance, often misclassifying popular songs. KNN performed slightly better, with an accuracy of 95.5% at k=9, but lacked interpretability and scalability for larger datasets. Random Forest outperformed both, achieving an accuracy of 97.98% and providing valuable insights into feature importance, with tempo, loudness, and acousticness being key predictors of song popularity. PCA effectively reduced dimensionality by identifying components that captured most of the data's variance, while clustering provided insights into grouping patterns, though it was not central to classification. Neural Networks matched Logistic Regression in accuracy but required significant computational resources and struggled with class imbalance, similar to other models. For cross-fold validation, we opted to use 5 folds to separate training into 4 of those sets and validation into one, for each iteration. Evaluation metrics such as accuracy, confusion matrices, precision, recall, and ROC-AUC were used to assess performance. With these metrics, we can see that the Random Forest model was the most accurate and easiest to generalize with the most confidence. Although it still has the limitation of having a low true positive rate of about 57.79%, it was the best model to fit our data set.

V. To use our code, please make sure to run the first two cells and install all the proper packages. To load the dataset, first find the dataset by going to https://huggingface.co/datasets/maharshipandya/spotify-tracks-dataset and downloading the data set. Then upload the corresponding file to your Google Colab, Jupyter Notebook, etc. and load this data using the third cell of code.

Appendix

I. To conduct EDA, we first explored the basic distribution of numeric features and the distribution of total popularity within the dataset using a histogram. After that, we computed the correlation matrix of all the numeric data to see how closely things were related with one another. Its correlation between popularity and other features seemed to be rather minimal. The top continuous numeric variables that seem to be most correlated with popularity were loudness (0.05), danceability (0.035), instrumentalness (-0.095), speechiness (-0.045). We then used a scatter plot to plot all four of these variables against popularity. It was hard to tell a relationship between these variables due to the overwhelming number of data points. We then split our dataset up into two smaller datasets: a “popular” song dataset and an “unpopular “song dataset. Songs with a greater than 70 popularity score were thrown into the “popular” dataset and songs with less than or equal to a popularity score of 70 were thrown into an “unpopular” dataset. The correlation coefficient for this new sub dataset did not change significantly. Next, we separated our data into a train, test, and validation set. We used 80% of the original data for training, 20% of the training data for validation, and 20% of the original data for testing.

II. We decided that we were simply going to drop the rows with missing data due to the fact that our dataset had 114000 rows of data but very few rows of missing data. There was not much more data pre-processing to complete as the Spotify data set given was already relatively clean for our methods of analysis.

III. We applied regression analysis to explore the relationship between popularity and all the other features. Using a forward selection algorithm, we found that instrumentalness, valence, speechiness, and danceability were the most significant features. To further investigate these features, we used linear regression to compare each variable to popularity. The R^2 scores for each model was consistently low, which likely meant that popularity was influenced by multiple features, not just one. We then computed a multivariable linear regression using all four features, trying to explore if there were any significant relationships there. The R^2 score remained very low though, meaning that popularity is most likely influenced by nonlinear factors. The residual plots for all models were similarly all over the place. Consequently, we were not able to use this analysis for feature importance. Similarly, low R^2 values means that our model was not overfitted, so regularization was not needed.

IV. For logistic regression, we prepped the code by creating a popularity threshold of 70 so that if a song's popularity score was above 70, it was considered as "popular". To do this, we analyzed all the auditory numeric features. After normalizing our training and testing datasets, we created a logistic regression model using the liblinear solver. This gives the binary value of 0 or 1, suggesting a song as "popular" or not. With this, we created a confusion matrix of the prediction results and calculated the logistic regression accuracy, true positive rates, and true negative rates. On our first try, our true negative rate was 100% while the true positive rate was 0% with an accuracy of 95.75%. This seemed to be due to the fact that our train test splitted data was heavily skewed towards "unpopular" songs. This was something that we had anticipated while conducting EDA as only around 4% of the whole dataset of songs were "popular" (had a rate of above 70). To dive deeper in this, we conducted cross validation to see how weighing "popular" songs and "unpopular" songs differently would affect these scores. As we placed a heavier weight on "popular" songs, the accuracy seemed to have gone down. However, this seemed to have been accompanied by an increased true positive rate and decreased true negative rate. The ROC and AUC curve looked fine. This means that logistic regression might not be the best model used to predict relationships between popularity and other auditory numeric variables. In terms of feature importance, the logistic model placed a big emphasis on instrumentalness, loudness, and energy. Regularization was not applied due to the fact that there was no overfitting.

V. K-Nearest Neighbors and Random Forest were both used for classification on our dataset. KNN was evaluated using different values of k , 3, 5, 7, and 9, and achieved a cross-validation accuracy of 95.558% when k = 9. From the confusion matrix we found that the true positive rate was around 4.75% and the true negative was 00.53%. This indicated that the model was good at predicting true negatives but struggled with identifying the minority class of "popular" songs. This is a reflection of how little "popular" songs there are in the dataset. We fit our training set to a random forest model with the original train test split dataset. We were able to achieve a higher accuracy of 97.98% with the Random Forest. From the confusion matrix, we found that the true negative rate was 99.77% and the true positive rate was improved to 57.79%. However, the metric for the true positive rate is still not ideal and was again a reflection for the imbalance of "unpopular "and "popular" songs in the dataset. The ROC curve and AUC rate were fine. The top 5 variables that the random forest placed an emphasis on were tempo, loudness, acousticness, speechiness, and valence. Overall, the Random Forest ensemble is the better performing model. This approach performed better in detecting the minority class. It also made it robust to overfitting and its ability to capture non-linear relationships and rank feature importance enhanced its interpretability.

VI. Originally, we wanted to use SVD to compute PCA, however, due to the size of our data, we used the python PCA model. After fitting the data to a PCA model, we found the loading variables of all 9 principal components and constructed a scree plot. We decided to focus on only 2 principal components using the elbow method for scree plots, which covered well over 90% of variance in the dataset. PC1 had a 99% emphasis on tempo and PC2 had a 99% emphasis on loudness. Using the first two PCs, we computed two clusters using k means clustering. There were two distinct clusters using this method, a red cluster (cluster 0) and a blue cluster (cluster 1). Cluster 0 corresponded to songs with higher tempo, loudness, and energy while cluster 1 corresponded to songs with higher acoustic. After computing a chi square test, it seems that cluster 0 is probably more associated with "popular" songs while cluster 1 is associated more with "unpopular" songs.

VII. Our project attempted to use a neural network to classify songs as "popular" or "unpopular". We classified songs as "popular" if they had a popularity score of < 70. The goal was to predict popularity using various numerical features. For the neural network architecture, we designed a simple feedforward neural network with one hidden layer containing 128 neurons and ReLU activation. The output layer consisted of two neurons using sigmoid activation function. We used the Adam optimizer and cross-entropy loss function to train the model. We also examined different learning rates including 0.0005, 0.001, 0.005, and 0.01 and epochs including 1, 5, 10, 25, and 50 to tune the model. With the different learning rates and epochs, we iteratively trained the model and performance on the test set was monitored to determine the best hyperparameters. Results showed that the neural network successfully learned the task and reached a maximum test accuracy of 95.75%. Training loss consistently decreased as epochs increased, which demonstrated effective learning. For example, at a learning rate of 0.005, the training loss decreased over 100 epochs from 0.706 to 0.356. Higher learning rates of 0.005 and 0.01 allowed for faster convergence compared to a 0.0005 learning rate which required more epochs to reach the same accuracy. However, despite the high test accuracy, the confusion matrix revealed class imbalance issues. The model performed well in predicting "unpopular" songs, which was the majority class, but failed to identify the "popular" songs. The ROC curve for the neural network showed an AUC of 0.53, which is only slightly higher than randomly guessing a song to be "popular" or "unpopular". These results may have occurred due to a bias toward the majority class and did not generalize well for the minority class. To address this issue in the future, we could use class weighting in the loss function to give more importance to "popular" songs during training.

VIII. An aspect of hyperparameter tuning that we applied in preparing for our project was determining the threshold for classifying a song as either "popular" or "unpopular". Our goal is to predict if a song would be popular, so we set a popularity threshold of > 70 for defining a song as "popular". This is important because it affects the distribution of the data into two classes: "popular" and "unpopular". To determine the threshold, we constructed a histogram to examine the distribution of the "popularity" scores in the dataset. Based on the histogram, we observed that the scores are highly skewed, with the majority of the songs having very low popularity scores. There is also a larger drop in the number of songs with a popularity score of 70 or greater. This suggests that songs with scores above 70 are considered more “popular”. This aligns with our focus on “popular” songs since the number of songs decreases after this point. Additionally, when going through logistic regression, we had to test with different hyperparameter weights of the “popular” vs “unpopular” songs due to the fact that our original dataset was heavily skewed with “unpopular” songs. We tried tuning how much of an emphasis we placed on “popular” vs “unpopular” songs but the results were not ideal.