Introduction to Data Science
Capstone Project
Group28: Ceci Chen (zc1634), Lia Wang (rw2618), Maggie Xu (jx1206)

Data Preprocessing Overview

1. Dimension Reduction:

For dimension reduction, we utilized Principal Component Analysis (PCA). Initially, our dataset contained 10 feature variables, making it complex and computationally intensive to analyze. By applying PCA, we reduced the dimensionality to 3 principal components, capturing approximately 57.36% of the total variance in the dataset. This step simplified our models and reduced the risk of overfitting.

2. Data Cleaning:

Handling Missing Values: We first scrutinized the missing values in the spotify52kData dataset and identified zero missing data. For the starRatings dataset, we filled the missing value with 0 which means that there is no interaction between user and song. We choose this approach to construct the popular base model for simplicity and avoiding potential bias that could mislead the final result.

Duplicate Removal: When calculating the top 10 songs with the greatest hits, we handle duplicates by dropping rows with the same combination of 'track name' and 'artists'. We drop the duplicate for data consistency since duplicate entries for the same song could arise from different sources or errors in data aggregation.

3. Data Transformations:

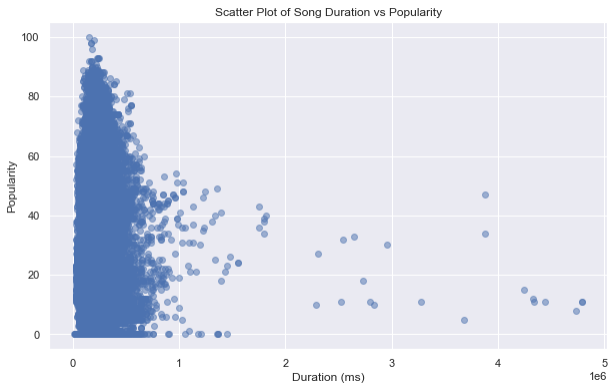
When building neural network models like a Multilayer Perceptron (MLP), we encode categorical variables (52 unique track genres) using LabelEncoder() and scale the feature data using StandardScaler() to make the data suitable for analysis.

4. RNG seeding

We seed our random number generator at 19329713 (N-number of Maggie Xu).

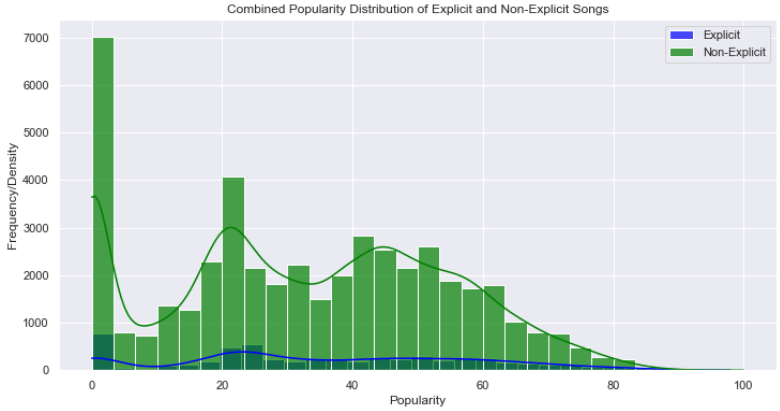
1) Is there a relationship between song length and popularity of a song? If so, is it positive or negative?

To conclusively determine if there is a significant relationship and whether it is positive or negative, a statistical analysis Pearson correlation test is conducted. This test would quantify the strength and direction of the relationship between song length and popularity. The Pearson correlation coefficient between song length and popularity is approximately -0.055, with a highly significant p-value 1.07e-35 (far less than alpha = 0.05). This suggests that there is a very weak negative correlation between song length and popularity. In other words, as the length of a song slightly increases, its popularity tends to decrease marginally.



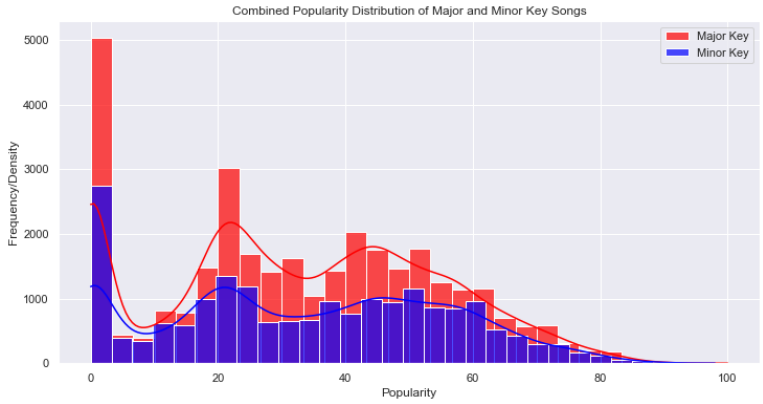
2) Are explicitly rated songs more popular than songs that are not explicit?

We first extracted explicit and not explicit rated songs and put them in the separate dataset. And then, we conducted the one tail t-test comparing the popularity of explicitly rated songs to non-explicit songs. With H0: explicit rated popularity = not explicit popularity and H1: explicit rated popularity > not explicit popularity. The test yields a t-statistic of approximately 9.83, with a highly significant p-value 4.25 e-23(far less than alpha = 0.05). This result indicates that we would reject the Null hypothesis and conclude that explicit rated songs are more popular than songs that are not explicitly rated.



3) Are songs in the major key more popular than songs in the minor key?

We first extracted the major(mode = 1)and minor(mode=0) key songs and put them in the separate dataset. The one tail t-test comparing the popularity of songs in major keys to those in minor keys. With H0: major popularity = minor popularity and H1: major popularity > minor popularity. The test yields a t-statistic of approximately -4.82, with a insignificant p-value 0.99 > alpha = 0.05. This result indicates that we fail to reject the Null hypothesis and conclude that there is no significant difference in popularity between major and minor key songs.



4) Which of the following 10 song features: duration, danceability, energy, loudness, speechiness, acousticness, instrumentalness, liveness, valence and tempo predicts popularity best? How good is this model?

We prepared the data used to fit the model by extracting these 10 song features and popularity from our original spotify data. We first tried to use the linear regression models to make the prediction. By creating a for loop, we use each feature in the 10 song features as a single predictor to fit the model and make a prediction on the value of popularity. By calculating the R^2 score between actual popularity value and predicted popularity value with each feature to evaluate the linear regression models, we got a relatively small R^2 for all 10 features. As shown in the left table below, feature “instrumentalness” has the largest R^2 score of 0.021017, which predicts popularity best. This means that for the 10 features here, they may not have a linear relationship with popularity and the linear regression models do not have a good performance here to make predictions.

Based on this, we then tried to use the random forest regression models for predictions. By creating a for loop, we still use each feature in the 10 song features as a single predictor to fit the model and make a prediction on the value of popularity. To evaluate the random forest models, we calculated the R^2 score between actual popularity value and predicted popularity value with each feature and found out that feature “duration”

has the largest R^2 score of 0.643402 as shown in the right table below, which predicts popularity best. For the random forest models here, we have much higher R^2 scores for all features, which means that they have a much better performance than the linear regression models on predictions and better capture the relationship between each feature and popularity.

	Feature	R^2 Score
6	instrumentalness	0.021017
3	loudness	0.003625
2	energy	0.003128
0	duration	0.002987
4	speechiness	0.002355
7	liveness	0.001922
1	danceability	0.001381
8	valence	0.001279
5	acousticness	0.000688
9	tempo	0.000007

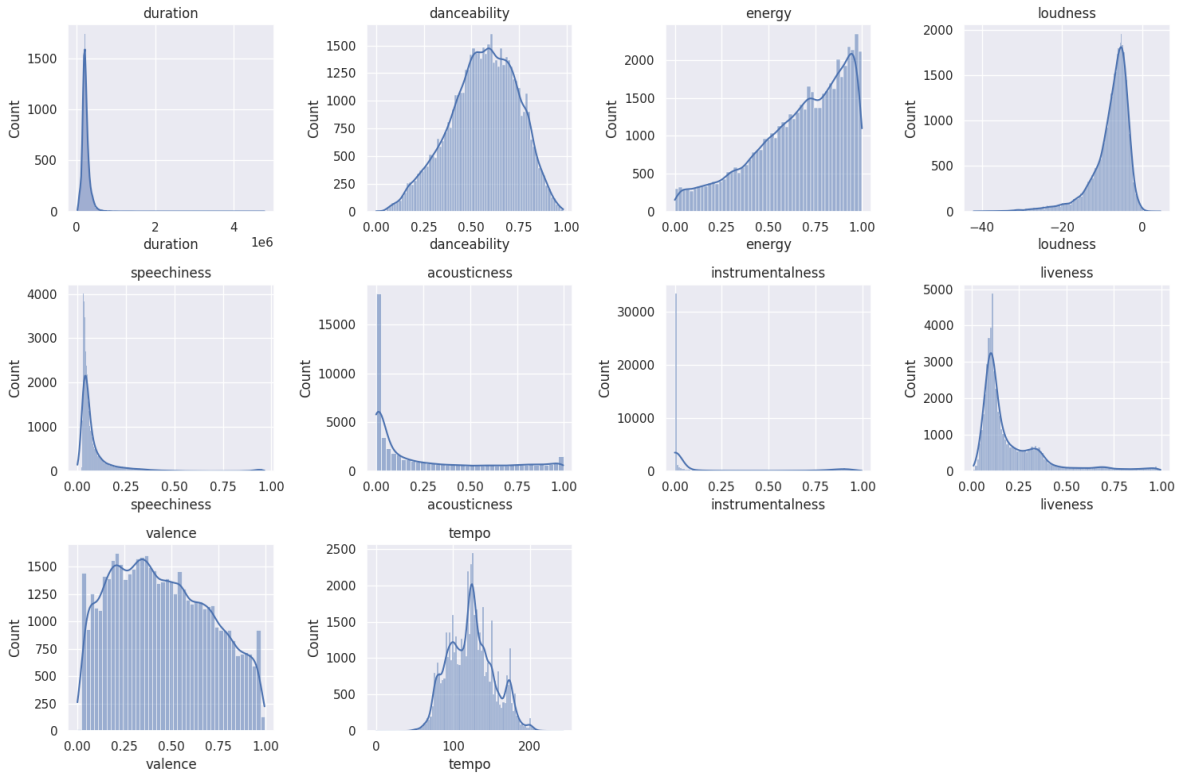
R^2 table for Linear Regression Model

	Feature	R^2 Score
0	duration	0.643402
9	tempo	0.606115
3	loudness	0.376893
5	acousticness	0.159760
6	instrumentalness	0.123671
2	energy	0.075572
8	valence	0.066103
7	liveness	0.062653
4	speechiness	0.059608
1	danceability	0.049276

R^2 table for Random Forest Model

5) Building a model that uses all of the song features mentioned in question 4, how well can you predict popularity? How much (if at all) is this model improved compared to the model in question 4). How do you account for this? What happens if you regularize your model?

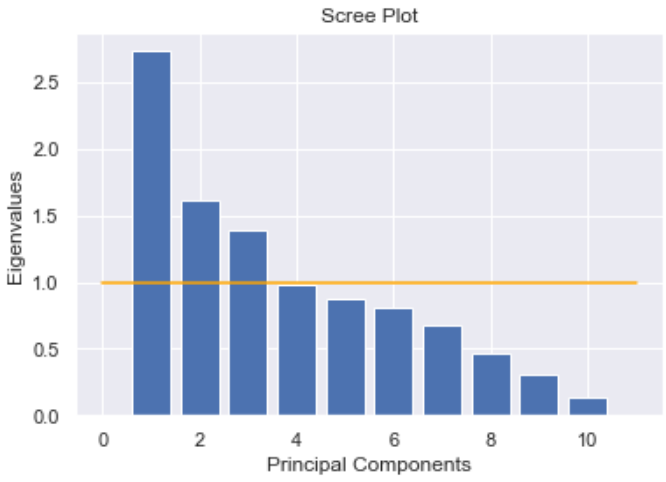
Below are the distribution plots of each feature.



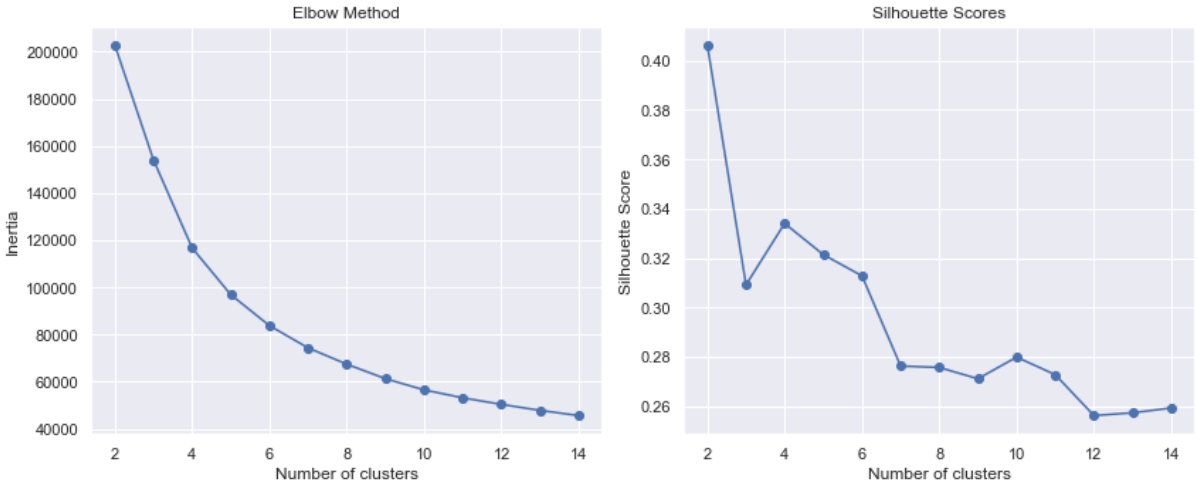
We build two models, multiple linear Ridge regression and Random Forest, and assess the performance of the models using COD and RMSE. We prepared the data by doing a 20-80 test split to train the model. For the linear model, without regularization, we got a R^2 of 0.0463 and RMSE of 21.1632315. Then we decided to use ridge regression because we want to avoid problems of multicollinearity, overfitting since the dataset has 10 features. The performance of the model can be summarized with a R^2=0.046 and RMSE = 21.1632. The performance doesn't improve much compared with the previous question (highest R^2=0.021 for the linear model), which might indicate that the relationship is nonlinear, so we decided to build a Random Forest model again. The R^2 for our Random Forest model is around 0.4, and the RMSE is 16.781. This model has a lower R^2 than the random forest model with duration as the predictor variable in question 4, and it is also lower than the R^2 for tempo as the predictor variable. We think that the reason might be that adding more variables increases the model's complexity. Since according to question 4's results, some of these additional variables are not strongly predictive, they might not contribute to, or could even detract from, the model's ability to explain the variability. Thus, we have a lower R^2 in this question.

6) When considering the 10 song features in the previous question, how many meaningful principal components can you extract? What proportion of the variance do these principal components account for? Using these principal components, how many clusters can you identify? Do these clusters reasonably correspond to the genre labels in column 20 of the data?

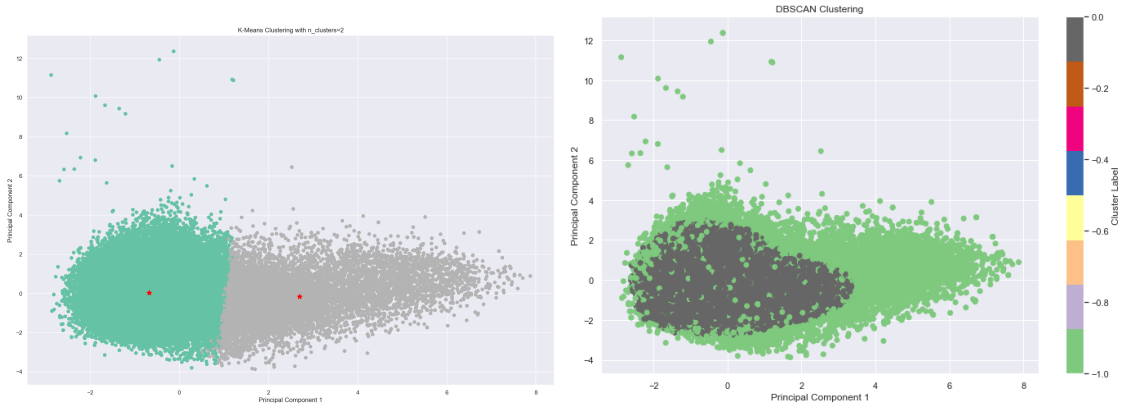
We conducted PCA for the 10 song features in the previous question. According to the outcome Scree Plot below, the orange line is the Kaiser criterion line where eigenvalue = 1, and we can see that there are three principal components with eigenvalues greater than 1. We decided to extract these three PCs for computations and found that these three components accounted for 57.36% of the total variance in the dataset.



We then find the number of clusters using the Elbow method by setting the range of possible number of clusters from [2,14]. To make sure our estimation is more accurate, we also computed the Silhouette Scores to determine the optimal number of clusters. According to the two plots below, we can see that when cluster = 2, the Silhouette score is the highest (greater than 0.4), and the scores show a decreasing trend for the larger numbers.

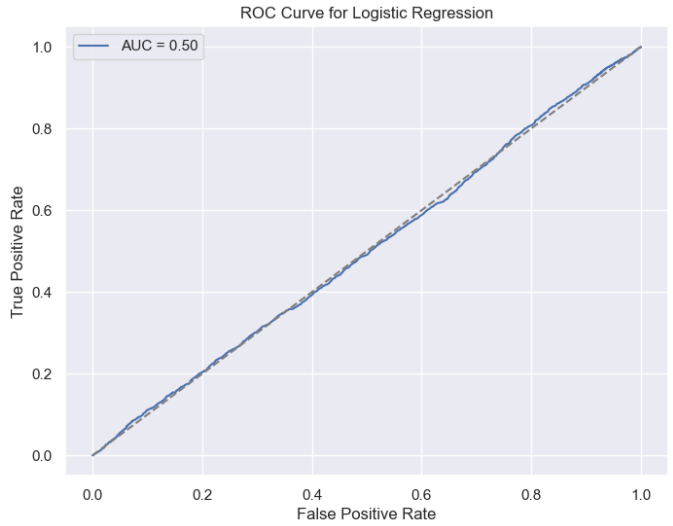


With 52 unique genres in the dataset, neither method suggests a close correspondence to this number of clusters. The high number of genres indicates that the genre classification may be too fine-grained or that the features may not capture all the nuances that distinguish 52 separate genres. We also visualized the clusters in the below scatter plot on the left, and we think that it is reasonable that the dataset does not have an inherent cluster. We then use DBSCAN to confirm our belief that 1 is the optimal number of clusters for our dataset, just as the plot shows on the right.

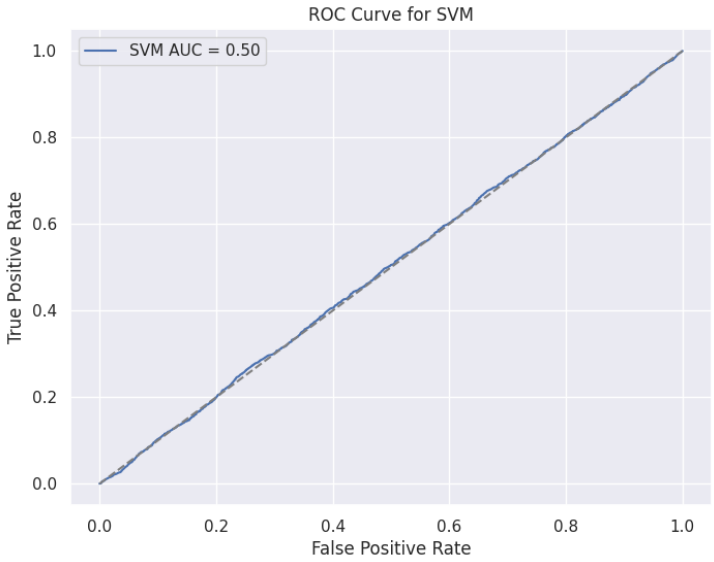


7) Can you predict whether a song is in major or minor key from valence using logistic regression or a support vector machine? If so, how good is this prediction? If not, is there a better one?

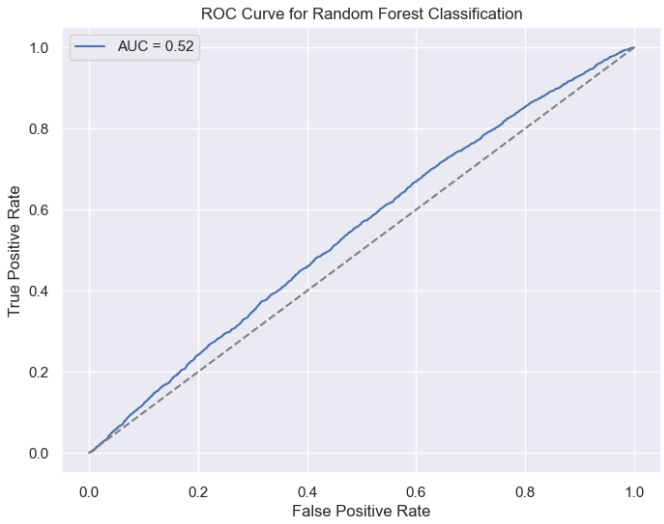
We prepared the data by extracting the valence column from the original spotify dataset our predictor X and extracting mode column as target y. After doing an 80/20 training and testing split on X and y, we first fit the training data with a logistic regression model (*LogisticRegression()*). After making predictions with test data, we calculate the AUC score to evaluate the model and plot out the ROC curve for this logistic model. We got an AUC Score of 0.5 here, which is very low and means that the logistic regression model is not doing a good performance in this case as the plot shown.

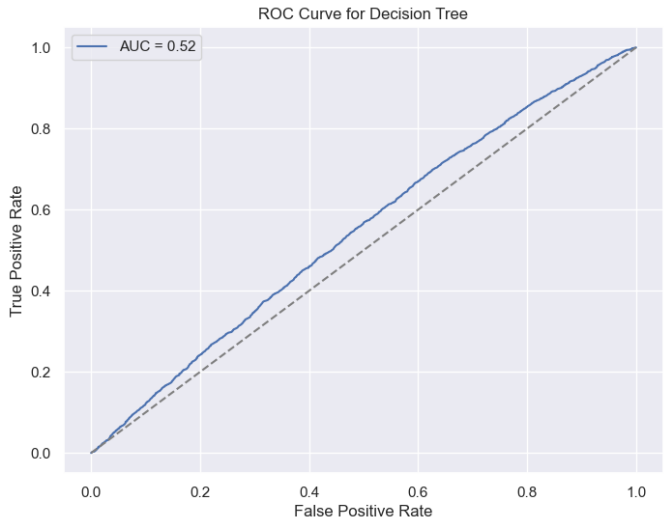


We then fit the training data with a support vector machine. By making predictions with test data, we also calculate the AUC score to evaluate the model and plot out the ROC curve for SVM. With an AUC score of 0.50, the SVM model also got a relatively very low AUC score. By plotting the ROC curves as below, we can see that SVM does not perform well in this case either.



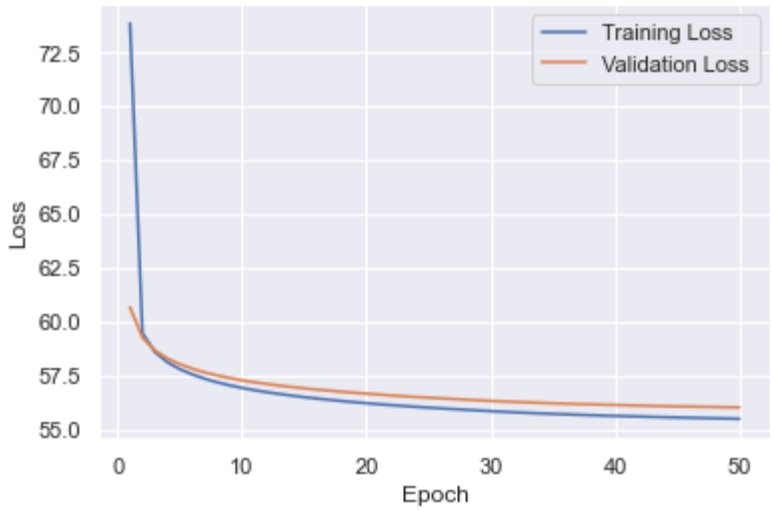
Since the logistic regression model and support vector machine are not doing good for predictions, we also try to fit the training data to a Random forest model and a Decision tree model. After making predictions, By making predictions with test data, we calculate the AUC score to evaluate these two models and plot out the ROC curve for both of them. Both of the models got an AUC score of 0.52, which is better than the logistic regression model and SVM. According to the ROC Curves below, the ROC curve for both models also includes more area than the logistic regression model and SVM. As a result, they are having better performance than logistic regression models and SVM in this case.





8) Can you predict genre by using the 10 song features from question 4 directly or the principal components you extracted in question 6 with a neural network? How well does this work?

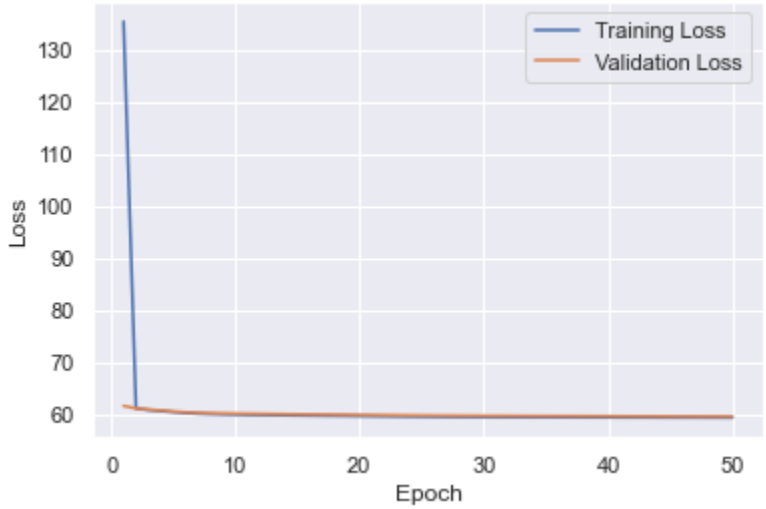
We first develop a set of functions to build the neural network. 'Module' function is a base class for neural network modules, with abstract methods forward and backward. The 'LeastSquareCriterion' is a loss function, assuming one-hot encoded targets, computes mean squared error between predictions and labels. The 'Linear' function is a linear (fully connected) layer, implementing forward and backward propagation. The 'ReLU' function is an activation function (Rectified Linear Unit) layer, implementing forward and backward propagation. The 'MLP' function is a Multi-Layer Perceptron class representing the neural network. It contains two linear layers separated by a ReLU activation function. Then, we develop a training function 'train_model', which trains the neural network model using a given training dataset, labels, and validation data. It implements the training loop with forward and backward passes and updates the model weights.



Feature Graph

After building all the necessary functions, we first predict the genre by using 10 song features from question 4. Scale the feature and encode the 52 unique track genre and split the dataset into training and test set. Then, we created instances of MLP and LeastSquareCriterion using feature and encoded track genres and trained using the train_model function. After training, the model's accuracy: 0.2456 is evaluated on the test set.

We then use the principal components you extracted in question 6 (which are 3 components) as a predictor to predict the genre. And have the model's accuracy:0.1352 in the test set. By comparing two model's accuracy, it concludes that the predicted genre by using the 10 song features from question 4 directly with a neural network has higher accuracy than using the PCA.

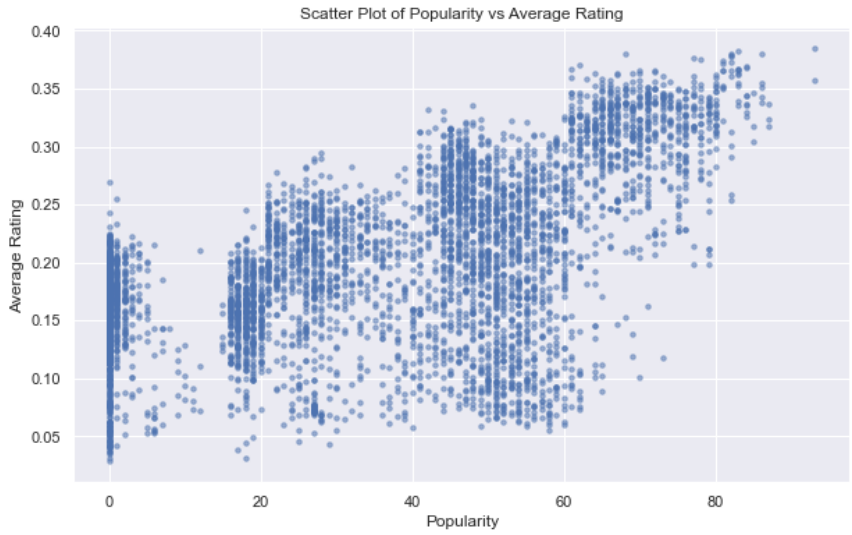


PCA Graph

9) In recommender systems, the popularity based model is an important baseline. We have a two part question in this regard: a) Is there a relationship between popularity and average star rating for the 5k songs we have explicit feedback for? b) Which 10 songs are in the "greatest hits" (out of the 5k songs), on the basis of the popularity based model?

To use the popularity based model, we prepare our data by first filling the missing value in the star rating dataset with 0, which means that there is no interaction between user and song, and calculating the average star rating for each song (mean value of each column). Then, we extract the first 5000 rows with columns of 'album_name', 'songNumber', 'track_name', 'popularity', 'artists'. By adding a new column 'average_star_rating' with values of average star rating for each song to the current dataframe, we are ready for the following questions.

a) We first use a simple correlation function to calculate the correlation between popularity and average_star_rating and get a correlation of 0.566, which means that there is a relatively moderate positive correlation between these two variables. As the plot shown below, we can also see an obvious positive trend from the points.



b) Since we find out that there is duplication in 'track_name' and 'artists', to find the top 10 songs with greatest hits, we first further handle our data by drop rows with duplicate combination of 'track_name' and 'artists' (drop_duplicates(subset=['track_name', 'artists'])). Then, we sort the data frame by popularity with descending order to have the top 10 songs. As a result, our 10 songs with greatest hits on the basis of popularity based model are songNumer of 2003, 2000, 3004, 2002, 2053, 3006, 4002, 3255, 2057 and 2367 with specific song information shown in the table below.

	album_name	songNumber	track_name	popularity	artists	average_star_rating
2003	I Love You.	2003	Sweater Weather	93	The Neighbourhood	0.3849
2000	Wiped Out!	2000	Daddy Issues	87	The Neighbourhood	0.3233
3004	abcdefu	3004	abcdefu	86	GAYLE	0.3802
2002	Hard To Imagine The Neighbourhood Ever Changing	2002	Softcore	86	The Neighbourhood	0.3380
2053	Pablo Honey	2053	Creep	85	Radiohead	0.3179
3006	Hybrid Theory (Bonus Edition)	3006	In the End	85	Linkin Park	0.3468
4002	Cigarettes After Sex	4002	Apocalypse	84	Cigarettes After Sex	0.3296
3255	Demon Days	3255	Feel Good Inc.	84	Gorillaz	0.3352
2057	Elephant	2057	Seven Nation Army	84	The White Stripes	0.3439
2367	The Colour And The Shape	2367	Everlong	84	Foo Fighters	0.3372

10) You want to create a "personal mixtape" for all 10k users we have explicit feedback for. This mixtape contains individualized recommendations as to which 10 songs (out of the 5k) a given user will enjoy most. How do these recommendations compare to the "greatest hits" from the previous question and how good is your recommender system in making recommendations?

We created a recommender system using collaborative filtering to predict the preferences of a user. We removed the duplicates from the spotify data and aligned each song with the ratings from the star ratings dataset. We then built a user similarity matrix and used it to find the top 10 song recommendations for each given user.

	songNumber	track_name	artists	popularity	Average Rating
1897	1897	Bienvenida	Jorge Drexler	18	1.55
2293	2293	Dragula	Rob Zombie	0	1.55
1895	1895	Sincronia	Samuca e a Selva	16	1.40
2460	2460	Are You Gonna Be My Girl	Jet	76	1.30
2480	2480	Miss World	Hole	0	1.20
991	991	Little Life	Frank Turner	25	1.15
565	565	Sappy - Early Demo	Kurt Cobain	50	1.05
2346	2346	Corazones Rojos	Los Prisioneros	0	1.00
37	37	Throwing Good After Bad	Brandi Carlile	0	1.00
2613	2613	Cough Syrup	Young the Giant	71	1.00

Above is an example of the recommendations for the user with index 0. We can see that the songs are very different from the top hits songs in the previous question, which might suggest that song preference is personal and subjective if the system performs

well. To evaluate the performance, we extracted the actual ratings and compared them with the predicted data. We calculated average precision and recall as the metrics for evaluation, and we set a threshold=4 (starts) as the indicator of whether the individual liked or disliked the recommended song. The result is precision = 0.999 and recall = 0.55, which indicates that there's a 99.94% chance that the user would like the recommended song (rate it 4 stars) and that the system correctly identifies about 55.1% of the actual songs.

Extra Credit: Can we use the number of beats per measure and the key to predict danceability of the songs?

We first prepared the data by extracting the 'tempo' and 'key' column from the original spotify dataset as our predictor X and extracting 'danceability' column as target y. Then, we do a 80/20 training and testing split on X and y for further model fitting. By plotting out the relationship between temp, key and danceability as the figure shown below, we are able to know that they are not in a linear relationship. Since danceability is a continuous variable, we use the random forest regression model to make the predictions. After doing the above steps, we evaluate our model by R^2 score and RMSE. Here, we got a R^2 score of 0.198 and RMSE of 0.158. As a result, we can make the conclusion that we can use the number of beats per measure and the key to predict danceability of the songs via the random forest regression model, where the model can also have a relatively good performance on this prediction.