DS-GA1001 Capstone Report

Group 30: Zhiwen Cao, Qihang Tang, Vivian Yan

Author contribution: All members contributed to each subquestion of this project on both codes and the report. We cross-validated our results of all 10 questions.

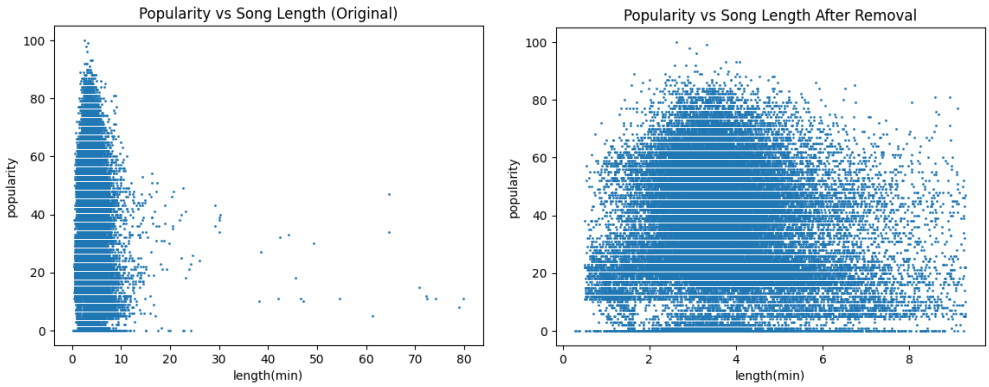
Statements:
Spotifydata: no missing values
starRating data: missing ratings

- Data transformation: The original dataset uses *duration* to represent the length of songs in ms. For better interpretation and readability, we converted it in minutes and replaced the column with new values.
- Data cleaning: We concatenated the artist name, album name, and track name as a new column called *id*. We kept the first occurrence of duplicate id such that id becomes unique in the transformed dataset. After transforming the original dataset, our cleaned dataset has a final shape of (45828, 21). We use the unique songs to deal with ratings as well by filtering the starRatings dataset and only maintaining user ratings for unique songs among the first 5k songs. While there are a lot of missing values, there are no users or songs with no ratings. So no rows or columns need to be dropped.
- Data Format Transformation: We transformed the rating matrix into a regular dataframe with columns of user_id, song_id, rating to fit the input format of the algorithm we used in Q10.
- Dimension reduction: we use a pipeline of a standard scaler and PCA to reduce the dimension and project our data (specifically in Q6).

Random seed number: 16385369

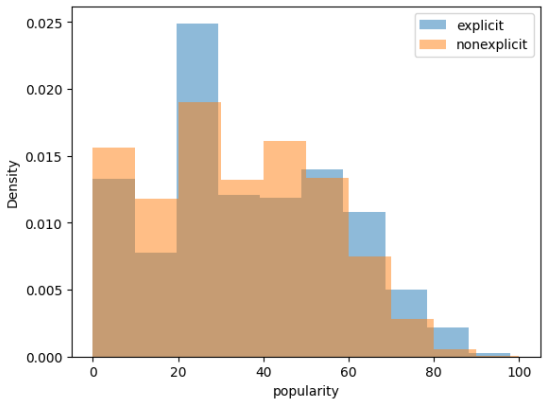
- When splitting the training set and test set and in other ML algorithms, we used `ramdom_state=seed`
- For cross-validation, we applied `np.random.seed(seed)`
- In the setting of Neural Networks, we used `tensorflow.random.set_seed(seed)`

Q1. To analyze the relationship between song length (duration) and popularity, we plot population against duration and compute the pearson correlation between the two features, which yields **r=-0.0675 and p-value=1.65e-47**. However, we observe some extreme values in durations in the scatterplot. By computation, we find 99% of the songs have a length around 9.30 minutes, so we remove the rows with song length greater than the threshold and repeat the process. The new pearson correlation gives **r=-0.0627 and p-value=8.76e-41**. Although the correlation is relatively small, it is still significant. Hence, we conclude a weak **negative** relationship between the song length and popularity.



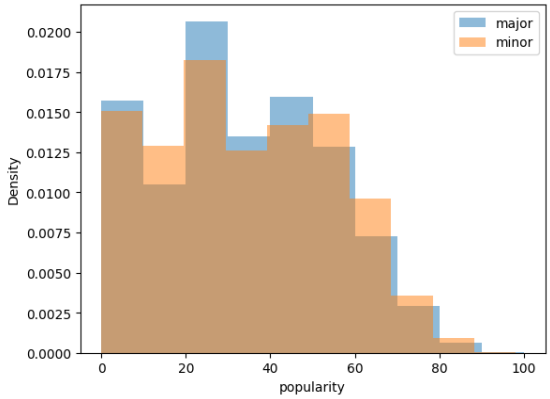
Q2. It is reasonable to reduce the popularity data to sample means and there are two sample means (explicit and implicit) are compared in the design. Since we do not know the population parameters and variances are non-homogeneous by the levene test, we use a one-sided **Welch t-test** and get the **test_statistic=8.96, p_value=2.12e-19**. Given our p-value<0.005, we reject the null and conclude that explicitly rated songs are **more popular** than implicitly rated songs.

```
LeveneResult(statistic=34.78128193243141, pvalue=3.7148394554060796e-09)
Unequal Variance
```

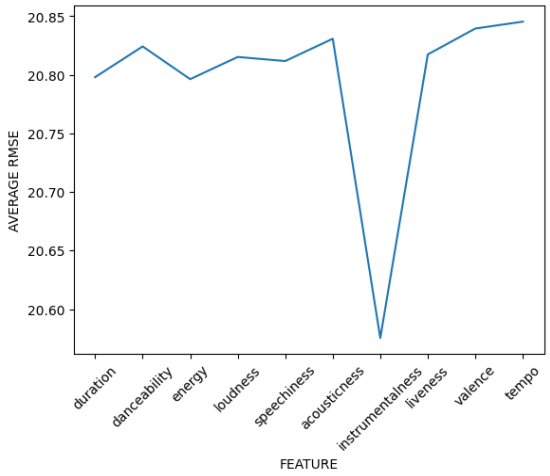


Q3. Same assumptions are made for this question as the previous one, we use a one-sided **Welch t-test** and get the **test_statistic=-5.37, p_value=0.99**. Given our p_value>0.005, we fail to reject the null hypothesis and conclude that songs in major keys are **not more popular** than songs in minor keys.

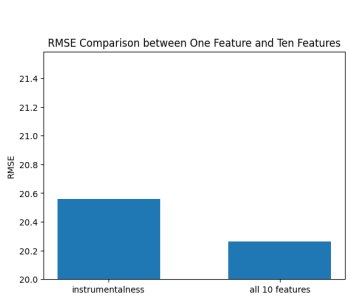
```
LeveneResult(statistic=38.745812280143504, pvalue=4.869159292351772e-10)
Unequal Variance
```

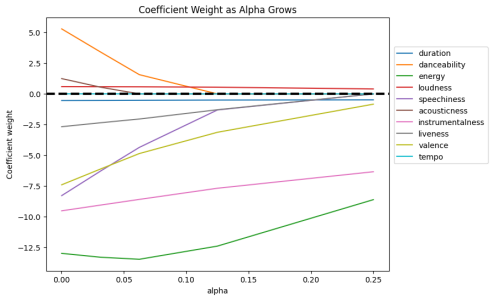


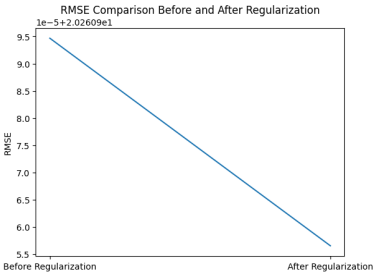
Q4. We split 80% of the data to be our training set and use the rest to test our prediction. For each of the selected features, we fit a linear regression model and apply a 10-folds cross validation on the training data. We plot the average RMSE against the features and select the feature with the smallest average MSE=423.35, which is ***instrumentalness*** to be our best predictor. Finally, we train a linear regression of popularity on instrumentalness and obtain a **RMSE of 20.56** on the test data.



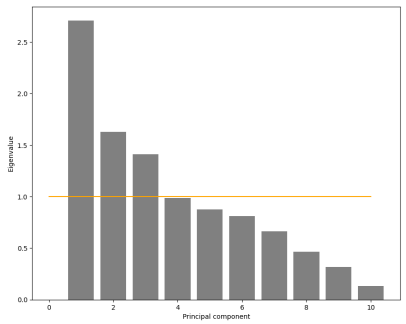
Q5. (1) Building a multiple regression model that uses all of the song features, the model gains a **RMSE=20.26099**, which **decreases by 1.44%** on the test data compared to RMSE from instrumentalness. But with too many features, our model might have overfitting issues. (2) Since **Lasso** will reduce the coefficient weights of some non-predictable features to zero when increasing alpha, we fit Lasso with its hyperparameter in the range of $[2.0^{-12}, 2.0^{-1}]$ to select features from the subset. By **grid search**, our **best alpha=0.002**, where most features are maintained (deduced from the plot), except for *tempo* with a coefficient of 4.92e-03. (3) With regularization, we obtain **RMSE=20.26096**, which **decreases by 1.88e-04%** from the unregularized RMSE. Since the improvement is trivial, we plot a line to illustrate the reduction in RMSE from regularization.

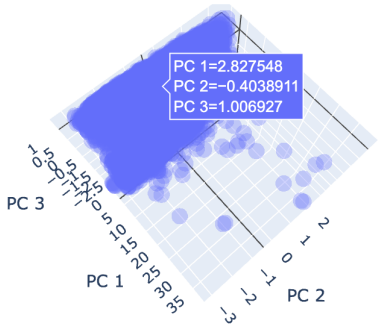






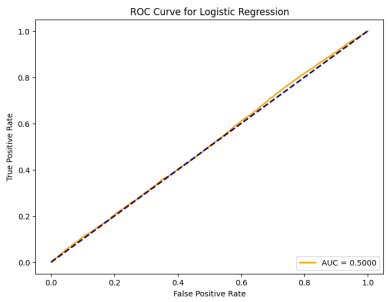
Q6. (1) Based on Kaiser's criterion, we notice that PCA gives us **3 meaning principal components** with respective eigenvalues greater than 1. (2) PCA finds eigenvectors whose eigenvalues show how much of the variance in the data is accounted for by the corresponding vector. Therefore, **57.49% of the variance** is covered by the 3 principal components. (3) From the 3D visualization of principal components, it is hard to eyeball the number of clusters. Applying K-means with Silhouette scores, we set the range of plausible k to be [2,11] and choose the **optimal number of clusters to be 3** with the highest sum of silhouette score. (4) The genre labels in column 20 suggest 52 unique genres/clusters. While our designed range for k doesn't include 52, we calculated the sum of silhouette scores when k=52, which is 13294.60. Since it is less than the peak of the silhouette scores at k=3, our observed clusters **don't match** with the genre labels.

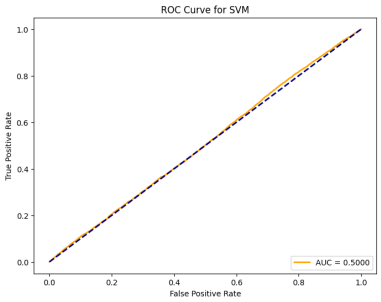


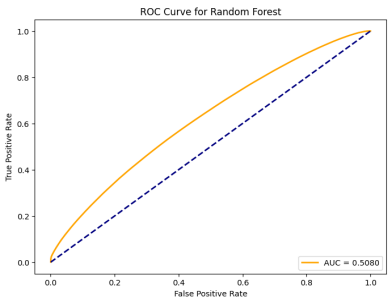


Q7. (1) Both logistic regression and SVM map continuous input to binary outcomes. In our situation, *valence* is real numbers and *mode* has two outcomes that can be represented as 0/1. However, logistic regression and SVM both give an **AUC=0.5**, indicating they are guessing randomly.

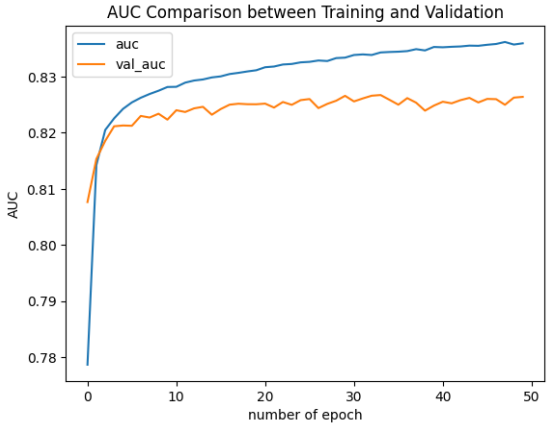
(2) To improve the classifier, we switch to a **random forest** classifier, which yields **AUC=0.508**. Random forest performs better than logistic regression and SVM, but with trivial improvements.



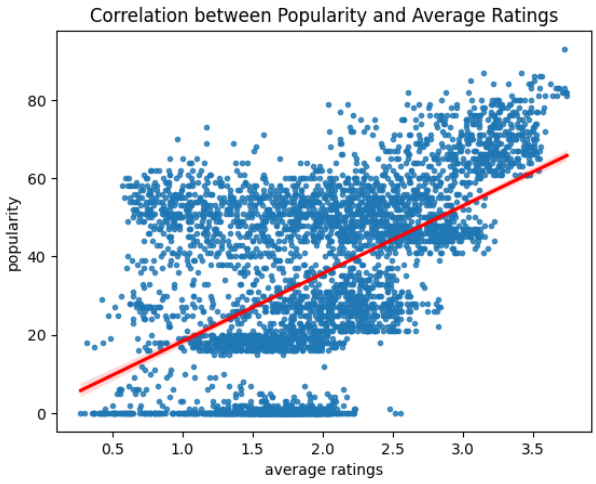




Q8. We use the 3 principal components from Q6 to predict the genre. Because it is a classification problem, we use label encoder and to_categorical to one hot encode y. We use a neural network with **3 hidden layers with 64, 32, and 32 perceptrons** in each layer respectively. Loss is categorical cross entropy, optimizer is ADAM, and metric is AUC. After training for 50 epochs, we test the model performance, which gives an **AUC=0.8264** and **loss=3.12**. The AUC comparison between training data (blue) and validation data (orange) is shown below.

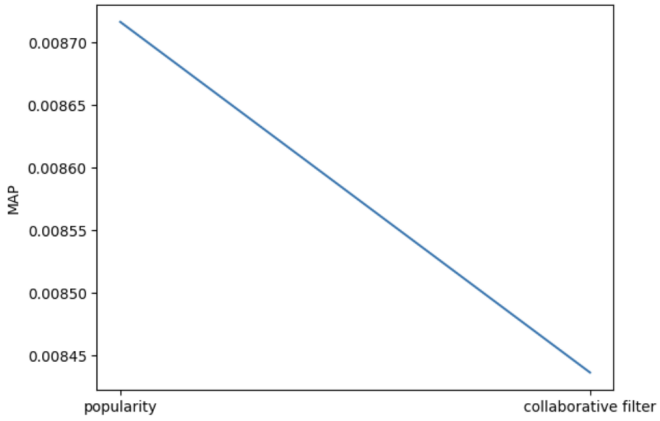


Q9. (a) Average star rating for each movie is obtained by taking the mean of each column of ratingdata. We plot popularity against average rating, and apply pearson correlation on these two features. The r value is **0.5281**, and p value is **1.036e-313**. The **positive** relationship is significant. The positive relationship between popularity and average star rating is also indicated by the clear pattern shown in the figure below.
(b) We sort the popularity in a descending order and extract the top 10 songs to recommend to all users: **Sweater Weather, Miss You, Daddy Issues, Softcore, abcdefu, Mr. Brightside, In the End, Creep, Feel Good Inc., Feel Good Inc., Losing My Religion**.



	track_name	popularity
2003	Sweater Weather	93
3300	Miss You	87
2000	Daddy Issues	87
2002	Softcore	86
3004	abcdefu	86
2106	Mr. Brightside	86
3006	In the End	85
2053	Creep	85
3255	Feel Good Inc.	84
2203	Losing My Religion	84

Q10. We assume that for each user songs that have a rating are considered relevant. The top 10 rated songs for each user are considered favorite songs. Therefore the recommender system should recommend these 10 songs to each user. After extracting the top 10 rated songs for each user and evaluating against the prediction from popularity based model from Q9, we have a **MAP of 0.0087**. In order to make personalized recommendations, we train an item based collaborative filtering on the ratingdata. By using the library **lenskit**, we can make 10 recommendations for each user. Evaluated against top 10 rated songs for each user, we get a **MAP of 0.0084**, which shows a **3% decrease** compared with that of the popularity based model. The personalized model performed a little worse than the popularity based model.



Extra Credit. Since energy tries to quantify how "hard" a song goes. Intense songs have more energy, while softer songs have lower energy. It is reasonable to assume that harder songs will have faster tempo because faster pace generally comes with greater energy and propulsion. We would like to investigate **whether there is a relationship between energy and tempo**.

We plot energy against tempo, and apply pearson correlation on these two features. The **r value is 0.245, and p-value is 0**. We conclude there is a **positive** relationship between energy and tempo.