A Quantitative Analysis of Songs on Spotify

Sydney Castillo
Nicole Nobbay
Ben Zhou

Research Questions:

1. **Do the songs with the most popular artists share a similar audio profile in terms of individual audio features? Does one audio feature seem to remain the most constant among these songs?**
    a. **Answer:** In short, we were unable to determine if there is a specific audio profile that is generally liked among listeners.
2. **Will Spotify recommend a list of popular songs if we pass in audio feature parameters that are representative of already popular songs? In other words, if we take sample statistics from popular songs, will we be recommended songs of similar popularity? From the results, can we determine if the audio features contribute to a song's popularity, or do these two seem to be unrelated?**
    a. **Answer:** Similar to the previous question, we have determined that passing in variables that represent statistics computed from popular songs will *not* return recommendations of similar popularity.
3. **Can we predict an artist's popularity score based on their song's audio features?**
    a. **Answer:** No, it is not possible to predict a song based solely on its features.

Motivation and Background:

Our team decided on having a specific focus on Spotify music data due to a shared love for music. With a Spotify account, the API brings access to many capabilities allowing for analyses of music trends, and trends within the audio features of an individual song. At our first meeting we decided to dive deeper into our curiosity of characteristics of popular songs reflected in Spotify and how Spotify interacts with this data. If we are correct in that a certain audio profile is generally likable, this can then help us identify additional factors that make certain artists more popular than others. This insight can help to answer one of our main goals for this project of using our knowledge in this course to see if we can predict whether a song's artist will be popular based on their song(s)'s audio features. Understanding how these attributes of audio features are reflected in popular artists can reveal the commonalities of audio features in these most streamed songs. This research will provide us with a new perspective on overlooked features of songs, and will strengthen our knowledge on audio feature correlation with the overall popularity of an artist.

Dataset

The dataset we will be using contains information about over over 19,000 songs in Spotify. We found this dataset off of Kaggle, and we have provided a link to both our shared repository where we uploaded a copy of the dataset, as well as the original author's post on Kaggle. This dataset includes quantitative values for a song's audio features, as well as an associated popularity score. We gravitated towards this dataset over many others because it had nearly 20,000 observations, and almost 50 different columns we could potentially analyze.

While combing through our dataset, we quickly realized that there were many columns that were unnecessary to our analysis, and could therefore be removed. The only columns that ended up being necessary to our analysis were 'popularity,' 'name,' 'danceability', 'energy', 'key', 'loudness', 'mode', 'speechiness', 'acousticness', 'instrumentalness', 'liveness', 'valence', and 'tempo.'

The **'popularity'** column is rated on a scale of 0-100, and all of the audio features besides 'key,' 'loudness,' 'mode,' and 'tempo' were scored from 0 to 1. The **'key'** column indicates the key the track is in, and is scored from -1 to 11. The **'loudness'** column indicated the overall loudness of the track in decibels. This column is scored from -60 to 0. The **'mode'** column indicates the modality of a track (major or minor). Major is represented with a 1, and minor is 0. Finally, **'tempo'** measures the overall beats per minute (BPM) of the track.

The **'danceability'** column indicates how suitable the track is for dancing based on a combination of tempo, rhythm stability, beat strength, etc. The **'energy'** column represents the intensity and activity of the track. Songs with high energy levels tend to feel faster and louder (think death metal music). The **'speechiness'** column detects the presence of spoken words in a track. The higher the value is to 1, the greater the likelihood the track is solely spoken (think talk shows, podcasts, etc.). The **'acousticness'** column is a confidence measure of whether the track is acoustic or not. The closer the value is to 1, the more confident the track is acoustic. The **'instrumentalness'** column predicts whether a track contains vocals or not. The closer the value is to 1, the higher the likelihood of the track being completely instrumental. The **'liveness'** column detects the presence of a live audience in the recording. The higher the value, the greater the likelihood the track was performed live. Finally, **'valence'** describes the mood of the track. Tracks with higher valence scores will sound more cheerful and euphoric, whereas tracks with low valence will sound sadder and more negative.

Methodology

1. For this question, we were looking to analyze the audio features of songs with a popularity score of 80 or higher to determine if a certain audio feature(s) tended to cluster

around the same value. Our suspicion was that if multiple popular songs shared common values between audio features, then there may be a certain audio profile that is generally liked among listeners.

To answer this, we first needed to distill our dataset down into just the most usable columns. This immediately eliminates any columns that store links. We decided on keeping only the following columns: ‘popularity,’ ‘name,’ 'danceability', 'energy', 'key', 'loudness', 'mode', 'speechiness', 'acousticness', 'instrumentalness', 'liveness', 'valence', and 'tempo.' Once we cleaned our data, we had to filter our data into songs whose artists have a popularity score of 80 or higher using a mask.

The first and second plot focused on utilizing this filtered data to visualize the presence and value of the audio features for these specific songs. The first plot took in all audio features in subplots to generate one plot to easily compare the range of audio feature values. We narrowed the focus on the second plot even more to compare only a select few audio features to visualize their ratio. We compared ‘danceability’ against ‘energy’ values with the size of these points varying by ‘speechiness’ level to generate a small subsection of our data.

For our third plot, we wanted to visualize some sample statistics about our audio features. Since ‘key,’ ‘loudness,’ ‘mode,’ and ‘tempo’ were not scored from 0 to 1, we decided to cut these out of our analysis. We decided that since the range of possible values were not confined to 0 and 1 like most of the other features, it would be unreasonable to determine whether the spread of values were comparable to the spread of values whose features ranged from 0 to 1.

After we computed these statistics, we were better equipped to find patterns among values for each audio feature. For this question, we implemented plots using Plotly because we wanted our graphs to be interactive. We liked that users could hover over individual data points for further information. To visualize the sample statistics of interest (IQR, median, outliers, etc.), we used a box and whisker plot. Doing so transitioned well into our second research question.

2. For our second question, we had already cleaned our data, so we did not need to repeat this process. The first thing we had to do was create a function that would provide the user with an OAuth token. To do this, we had to collect our Client ID and Client Secret from our Spotify developer account, and encode this using the base64 library. Once our ID and Secret were encoded, we had to create two dictionaries: one that would indicate that our ‘Authorization’ level was ‘Basic,’ and another that would indicate that our ‘grant type’ was ‘client credentials.’ This function then returned a new OAuth token whenever it was called.

   Next, we created a function called compute_quartile that we could call in order to return a lower and upper boundary for a specific audio feature. To do this, we used the

percentile function from the NumPy library. This gave us an upper and lower boundary to pass in as parameters for our API request.
Once we had a way to generate our OAuth token and lower and upper boundaries, we created a final function called get_recommendations which we would use to actually make our GET request. To do this, we stored the base API URL as our endpoint. We also created a variable called 'query' that was an f string which contained the endpoint, sample genre, and the upper and lower boundaries for our audio feature.

We stored the resulting JSON in a new file, as well as a .txt file which contained a simplified list of recommendations.Once we received a response with recommendations based on the feature we passed in, we assessed the popularity of the responses to see if the recommended songs also had an average popularity score of 80 or higher. The information about the popularity of each song was included in the API's JSON response.

3. In order to try to predict an artist's popularity using song features we created 3 machine learning models that were each implemented differently in hopes of one of them getting convincing results. The three implementations we built were a DecisionTreeRegressor, a MLPClassifier, and a custom neural network using Keras. For all three implementations, we randomized the data and split them into training and testing by a factor of 80/20. The song features we used were {'danceability', 'energy', 'key', 'loudness', 'mode', 'speechiness', 'acousticness', 'instrumentalness', 'liveness', 'valence', 'tempo'}
Decision Tree Regressor:

This implementation was quite simple. We imported our DecisionTreeRegressor from sklearn.tree. We then fit the model with train data and called predict with testing input.
MLPClassifier :

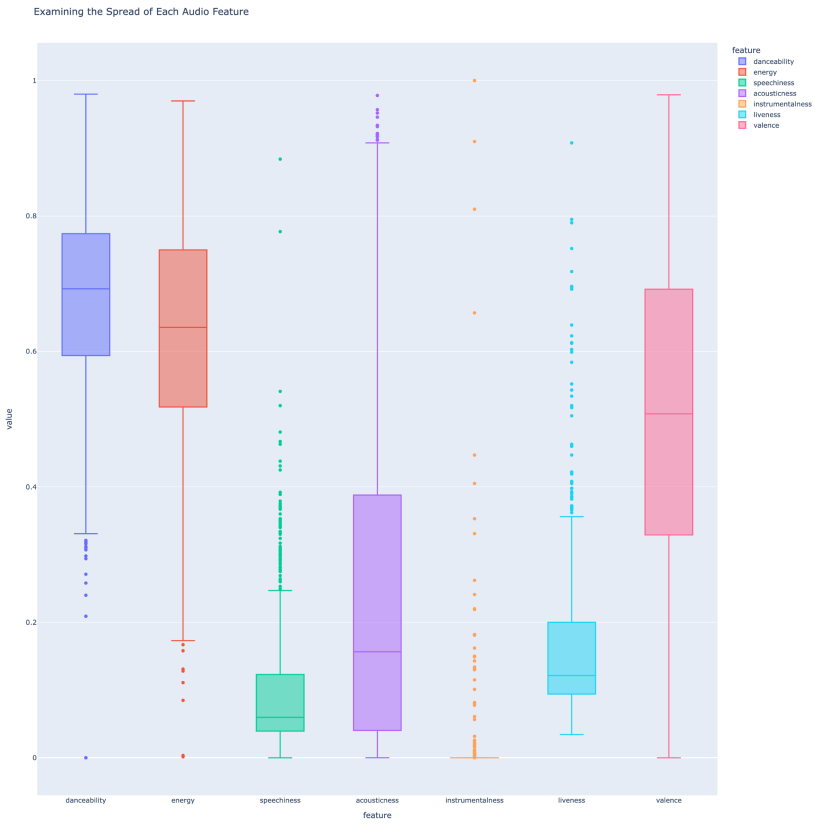
This was a slightly more complicated model since we are creating a neutral network. We first had to think about the number of hidden layers and max iterations to use. We tried our best to research and understand neutral networks and ended up settling upon 4 layers. As Jeff Heaton (author of *Introduction to Neural Networks in Java*) suggests 'the optimal size of the hidden layer is usually between the size of the input and size of the output layers'. As such, we set our hidden layer size to 4; a midpoint between our input layer of 8 and our output layer of 1. As for max iterations we found by testing that there was an upper limit to how many iterations we should use before our results no longer improved. We found this limit to be around 1000.
Keras Neural Network:

This was the most complicated and challenging of the three implementations. For this implementation we had control over each layer of our neural network and over the way we compiled it. In order to define the model architecture we had to do many layers of research that ultimately culminated in decisions we made to create our model. For model choice we went with a Sequential model. This is the one that TensorFlow suggests

for regression and is appropriate when there is only one input and output tensor. We next had to define the layers of our model. We decided to go with Dense layers because we wanted fully connected layers with nodes interacting since there are multiple song features and we need these variables to affect each other. The optimiser that we chose was the gradient descent algorithm 'adam'. This gradient descent is useful because it will automatically tune itself and be helpful in a wide range of situations. We then fit the model with the training data and defined the epochs and batch sizes we wanted to use. An Epoch is defined as "One pass through all of the rows in the training dataset."and a Batch is "One or more samples considered by the model within an epoch before weights are updated." Typically as both of these numbers increase your results should get better however for us that was not the case. We struggled immensely with getting good results with this model and it seemed that epoch and batch sizes had little effect on the results for us.

Results

Do the songs with the most popular artists share a similar audio profile in terms of individual audio features? Does one audio feature seem to remain the most constant among these songs?



We thought that if we found most audio features had a low interquartile range relative to units they are measured in (most, but not all, audio features are scored from zero to one), then this could indicate that songs with high popularity scores could fit a generally likable audio profile. Specifically, we hoped we would find an interquartile range of 0.075-0.1 for features rated from 0 to 1. Alternatively, if we found that the interquartile range for the audio features varied, then this could indicate that the correlation between song and artist popularity is not largely based on how the song sounds; factors that cannot be quantified, such as lyrics, may have a

greater impact on an artist's popularity score.

Through our box plot specifically, we were able to determine that the interquartile range of each audio feature varied, with only speechiness and instrumentalness falling within our range of interest (0.075-0.1). Because the features varied in their IQR, we came to the conclusion that songs with high popularity scores do not tend to cluster around one value for each feature. The only audio feature that seemed to remain the most constant was instrumentalness, but given Spotify's definition of this, we were already under the impression that none of the most popular songs would be fully instrumental.

With the two scatter plots, we were able to plot the value of the audio features to compare with each specific feature. The x-axis values helped us visualize what a common amount for that audio feature would be within popular songs that helped us to conclude that popular songs do not have one general audio profile, as there was little to no correlation between popularity and the features being examined.

Will Spotify recommend a list of popular songs if we pass in audio feature parameters that are representative of already popular songs?

For this question, we were curious to see if Spotify would recommend songs with artists of similar popularity (80 or higher) if we passed in lower and upper boundaries for audio features of interest according to the previous research question. To determine the results of this, we used Spotify's 'Get Recommendations' endpoint, which returns an array of track objects. We decided to use quartiles for lower and upper boundaries instead of mean and standard deviation, as quartiles tend to be more robust to outliers.

Through testing our function and generating multiple sets of recommendations, we found that we were receiving peculiar results. We were often receiving song recommendations from artists we knew to be popular, but the popularity score associated with the song would be shockingly low. For instance, one recommendation we received was 'Started From the Bottom' by Drake. Through our knowledge of pop culture and current music trends, we found it very shocking that the popularity score associated with this song was zero. On one hand, we trusted that the results we received were correct, as the API successfully returned recommendations to us. On the other hand, we knew some part of this had to be inaccurate. We tested this with another song: another recommendation we received was 'Dangerous Woman' by Ariana Grande, but once again, the associated popularity score was only five. We will discuss more on why this was occurring in our "Testing" section.

Since running the API request returned a new list of recommendations each time, it was hard to standardize the average popularity score of each set of results. The set of results we got also varied with which audio feature we were using as a parameter. Another thing that impacted our results was the seed genre we provided, but for the sake of consistency, we used 'pop' for each

recommendation call. Despite the various factors affecting the recommendations we received, we found that the average popularity score we received ranged from 55-65. This research question tested if we would be recommended songs with a popularity score of 80 or higher if we used audio feature parameters that also met this criteria, but we can conclude that these parameters are not enough. Similar to our conclusion from the previous research question, we believe that a song does not become popular because of its audio features. There are many qualitative factors that go into what makes a song popular, and inputting raw numbers that quantify the audio profile of the song is not enough information to return popular recommendations. Additionally, popularity is time-sensitive. Songs that were popular last year may have a much lower popularity score today based on streaming data. This logic may also apply to what general audio profile is popular.

Can we predict a song's popularity score based on its song's audio features?

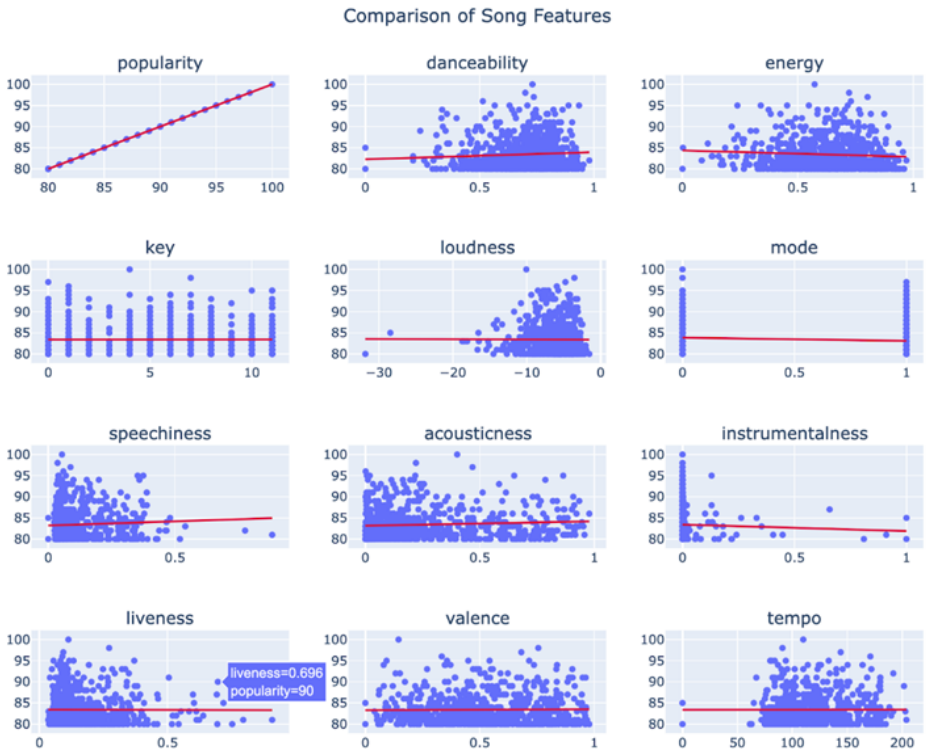
Results:

Unfortunately the conclusion that we came to is that, no, it is not possible to predict a song based solely on its features. All of our models had mean squared errors over 50%. Our keras model can very consistently come in around 120-130% MSE, the MLP model ranged around 130-140% MSE and, the sklearn model ranged around 220-230% MSE.

Why was the MSE so high?:

We believe the primary problem was that the popularity value that Spotify gives a song is independent of the individual factors that make up its audio profile. We believe that popularity is highly subjective and that what makes a song popular has many factors that cannot be quantified. For instance, an artist with an already large fanbase will likely yield songs that are more popular than an artist who has a small fanbase, even if their audio profiles are similar.

Not being able to quantify what makes a song popular can lead to many songs having the same popularity value which can lead the model astray, as they might have completely different features. Another problem is that many audio features generated by Spotify are pretty subjective. For example, "energy" and "danceability" are features we used, but it's unclear



what this exactly means and if all the numbers are even accurate to what they are supposed to represent. This makes it hard for a model to be fitted properly as the input data might even be misleading. Lastly, we believe that there isn't much of a correlation between the two values just by looking at the data. If we take a look at graphs of popularity value vs. audio features (Popularity on Y-Axis, feature on X-Axis), we see that most of these lines are nearly horizontal, indicating very weak relationships between the two axes. In conclusion, we believe that our results are fairly representative of the state of the data and as such, believe it is unreasonable to predict a song's popularity score based on audio features.

Impact and Limitations

Machine Learning Limitations:

We had some technical limitations with the way we designed our machine learning models. If we had better computers we could have run our models with higher epochs and batch sizes which could have resulted in more accurate models. Another limitation would have been with our data. If we had access to all of the millions of songs and would be able to run that through the algorithm it could have yielded better results. It's very possible the songs in our data set were biased and or inaccurate.

Societal Impacts:

Although our results didn't lend much insight into predicting song popularity, we believe this is a testament to popularity being related more to talent rather than formulas. We believe our results are valid because if there were a formula that could instantly produce hits, it would probably be more widely used, resulting in homologous music. In addition to the overproduction of homologous music, we do not doubt that this would result in many lawsuits and copyright issues. It is quite possible that our data was dated, or possibly just did not include very many data points for songs that were hits, but even then, we still stand by our findings.

We believe that our findings do more harm than good, as they encourage individuality in music creation. However, despite individuality being encouraged, this still leaves an impossibly tiny amount of space for smaller musicians to prosper in the industry.

Challenge Goals

1. The first challenge goal we will be working towards is **learning and implementing new libraries.** The first new library we will be using is a new machine learning library called Keras in order to develop an artificial neural network in hopes of creating a better model than one using sklearn. Because we planned on machine learning being another one of our challenge goals, in addition to learning Keras, our group will also be applying new plot formatting using Plotly. We are intrigued by the amount of flexibility with the plots, and liked that we were able to make interactive visualizations.

2. The second challenge goal we will be addressing is **processing and cleaning our API data.** Although the Spotify API offers a ton of rich information, the JSON formatting can be quite difficult to read. We anticipate that we will need to spend extra time figuring out how to neatly process the data we receive from our GET requests. In terms of neatly processing the JSON, we want to figure out how to extract just the most useful pieces instead of having to read through the entire file for the few pieces of information that are truly relevant to us.

3. The final challenge goal that we will be working towards is **developing a regression model using machine learning in order to predict the popularity of an artist on a scale of 0-100.** We will be using the library we learned in the class (sklearn) to build and train our model. This will also serve as a baseline comparison for the other models we plan on building. We used the Keras library to develop an artificial neural network in hopes of creating a better model than one using sklearn. We wanted to experiment with the nodes of our neural network in an attempt to build a better model than the one we have used in class.

Work Plan Evaluation

From our shared GitHub repository that kept track of files, data, and code, we each helped contribute to achieve full transparency within our team that is clearly reflected in our final project outcome.

1. At the start of our final project journey, we met in person for the first time to discuss our own goals for the project and what we wanted to accomplish. We established early on what our time availability was like that transferred into answering our challenge goals and research questions. For our work plan outline we wanted to first clean the data, then implement our work on the challenge goals and research questions to divide our time to create our finished product. From then on, we had Zoom meetings to share what we have worked on and if we needed help in creating a specific part of our project.

2. The first part of our work plan was to clean and prep the data. For the first two research questions, filtering and cleaning the data were almost identical in process, so we decided it would be best to create a function to do this in the question 1 script, and then import it into the question 2 script. With each research question, the data cleanup took varying amounts of time, significantly less than what we had originally proposed, our cleanup took approximately 2 hours for each member of our team. **(Expected: 12 hours, Actual: 6 hours)**

3. Our next step in our process was to compute sample statistics, and generate data visualizations. To better visualize the statistics we wanted to examine, we created 3 plot visuals to support our original research questions with the help of the implementation of new libraries. We decided to use plotly to create a box plot and two scatter plots to clearly represent the discoveries we found while analyzing the audio feature values. This task ended up taking more time then we had originally proposed due to the process of learning new implementations with this library. The process was spread throughout our day and resulted in approximately 8 hours to create these finished plots. **(Expected: 6 hours, Actual: 8 hours)**

4. Once we had some solid visuals we could build off of, we started working on our second research question. This involved using the Spotify API, which none of us were familiar with. It took us quite a while to figure out how to authenticate our credentials. Once we were able to authenticate our credentials, we used the box plot to determine which audio features had the least amount of spread. We used these features to test our second research question. Overall, using the API and processing the resulting JSON data took quite about 5-6 hours of total work. **(Expected: N/A, Actual: 5-6 hours)**

5. Next, we had to start training our machine learning model. We first proposed the idea of implementing a regression model using Machine Learning that transitioned 3 different Machine Learning models that included a DecisionTreeRegressor, a MLPClassifier, and a custom neural network using Keras. This process was constantly being refined to produce convincing results that took a majority of our research and time during our work process. Spanning the course of days that translate to over 10 hours to reflect our main mission and goal, though less than what we expected. **(Expected: 15 hours, Actual: 10 hours)**

6. Finally, we took the last few days before the deadline to clean up our code and make sure what we have written is high quality. Additionally, we will use this time to confer about our findings, and determine what the results of our research questions were. It is important to go over our data analysis to showcase our Python knowledge that we have learned and strengthened over the course of taking this class. Making sure our final project reflects our values and learning that we now possess is one of our team goals, and therefore took us about 6 hours to complete. **(Expected: 6-8 hours, Actual: 6 hours)**

Testing

As discussed in the results for our second research question, our group was skeptical about the popularity scores we were receiving in our recommendations. After consistently receiving strange results like the ones described in question two, we decided to revisit the API documentation. Initially, we were under the assumption that the ‘popularity’ column

corresponded to an artist's overall popularity score as outlined here in the Spotify developer documentation. What made this more confusing was that this definition could also be found in the 'Get Track' documentation (note that this endpoint is significant to our analysis for this question, as the 'Get Recommendations' endpoint returns an array of track objects). However, after reading further into this documentation, we realized that the 'popularity' score can either correspond to an artist's overall popularity, or an individual track's, depending on the context it is returned in. The popularity score we were referencing in the data, as well as the popularity score being returned to us, corresponded to the individual track rather than its artist.

To verify this, we created two additional functions: one that would return information about specific artists, and another that would return information about specific tracks. To test this, we used the URI (unique identifier) for one artist/track from each of the files we created through our get_recommendations function. The four tracks we tested were 'IDOL' by BTS (popularity score = 40), 'Dangerous Woman' by Ariana Grande (5), 'Started From the Bottom' by Drake (0), and 'Firework' by Katy Perry (71). Through the information we received from the 'Get Several Tracks' endpoint, we were able to verify that the track popularity scores were 40, 5, 0, and 71 respectively. Through the information we received from the 'Get Several Artists' endpoint, we found that the popularity scores for each artist were 96, 93, 98, and 88 respectively, which validated the pop culture knowledge we held of these artists. The sample files can all be found in the test-api-files folder.

Collaboration:
None.