January 06, 2023

# C964: Computer Science Capstone Documentation

*Dalton Riley*

# Part A: Project Proposal for Business Executives

## Letter of Transmittal

January 12, 2023

John Doe, CTO

Music Streaming Company

123 Sharp St

Houston, TX

Dear Mr. Doe,

While the music industry is shifting towards streaming more than ever, we are still seeing less growth than our competitors. This could be related to several factors, but it is my express belief that the number one reason for this is the lack of a music recommendation service within our platform. The recommendation is becoming more commonplace than ever, and we are one of the last platforms to adopt it.

If we were able to deliver a data product that recommends music to our users, I believe we will see direct growth in monthly users. By incorporating a random forest algorithm into our vast music dataset, we will easily be able to achieve this. After the algorithm is in place, we can set up a new tab within the platform for the users to find new music that is similar to their tastes.

Recommending new music to our users will benefit us in several ways. It will keep our existing users more engaged and spending more time on our platform. With current users more engaged, it will be easier to sell premium services and increase revenue. This will also result in more users adopting our platform.

Upfront costs will considerably less than the amount we will gain from adopting this service. I estimate we will need a budget of around $100,000 to fund this project, and we will be looking at a time frame of around 3 months. I also believe I am the best fit to lead the development of this product. With a bachelor's degree in computer science and over five years of experience in developing data products that utilize machine learning, I am well-equipped to execute this task.

Thank you for your time. I look forward to your response.

Sincerely,

Dalton Riley

# Project Recommendation

## Problem Summary

Our company has managed to stay afloat in the shifting landscape of the music industry for almost ten years. While this is an accomplishment to be proud of, there is another shift in the landscape happening at this moment that is currently catching us off guard. Up until now, it has been enough to provide our users with the ability to listen to music and leave the rest up to them, but a recent technology, music recommendation, is making its way through our industry that will make that outdated. Creating a music recommendation system for our platform is paramount to our continued success.

Many of our competitors are already implementing music recommendation systems which is why we have seen a recent drop in our monthly users. This is we will create our system that will recommend music to our users based on their listening history. By implementing our recommendation system, we will see our revenue back to or higher than it was before.

## Application Benefits

The music recommendation system will benefit us in several ways. Recommending new music to our users will keep them more engaged on our platform. A more engaged user will talk to their friends about us, bringing in new users and more revenue. An engaged user will also be more willing to upgrade their accounts to our premium options, which will also result in more revenue. Lastly, it will keep us up to date with our competition. This means that we can limit users from choosing another service instead of ours.

## Application Description

In its final iteration, the system will be a tab on our platform that users can click on to find music they have not heard before. The deliverable for this project, however, will be a Jupyter notebook with a small sample dataset. The training dataset will represent a user's listening history and we will use a random forest algorithm to group songs with similar characteristics. If a song in the test dataset has similar characteristics to the liked songs in the training set, it will be recommended.

## Data Description

The data will come from Spotify's API and is provided by Brice Vergnou (2021) in his GitHub repository. The data includes song names and artists, as well as qualifiers for the songs. Below you can find a direct quote from the repository explaining the data qualifiers, "From Spotify's API documentation :

- **acousticness**: A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
- **danceability**: Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is the least danceable and 1.0 is the most danceable.
- **duration_ms**: The duration of the track in milliseconds.
- **energy**: Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy.
- **instrumentalness**: Predicts whether a track contains no vocals. “Ooh” and “aah” sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly “vocal”. The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.
- **key**: The key the track is in. Integers map to pitches using standard Pitch Class notation. E.g. 0 = C, 1 = C♯/D♭, 2 = D, and so on.
- **liveness**: Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides a strong likelihood that the track is live.
- **loudness**: The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing the relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typically range between -60 and 0 dB.
- **mode**: Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.
- **speechiness**: Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audiobook, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.
- **tempo**: The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, the tempo is the speed or pace of a given piece and derives directly from the average beat duration.
- **time_signature**: An estimated overall time signature of a track. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure).
- **valence**: A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).

And the variable that will measure the success of the prediction:

- **liked**: 1 for liked songs, 0 for disliked songs". (Vergnou, 2021)

## Objectives and Hypothesis

The main objective of this project is to correctly recommend songs that a user will like. Creating data visualizations to further improve our understanding of the data points is another objective. After these objectives have been accomplished our main outcome will be to create a simple user interface to bring all these components together into one cohesive application.

I hypothesize that a random forest algorithm will be the best fit for our problem. It groups similar data points into categories, which is perfect for predicting music preference.

## Methodology

We will be using the agile methodology throughout the development of this project. We chose this methodology because it offers adaptability, good communication, and an iterative approach that will provide the best quality. Using this method, we can break the project into six phases:

1. Concept: In this phase, we will create the scope for our project, and outline key requirements. We will also analyze the estimated cost to determine if it is worthwhile to move forward with the project.
2. Inception: In the inception phase, the team who will work on the project is chosen. They will then create a simple user interface and a framework to start working on the bulk of the project. Requirements are fully determined during this phase.
3. Iteration: In this phase, the team will go through multiple cycles to complete the project. The first cycle should yield a simple, but functional iteration of the product. Each additional cycle adds more functionality until the project is complete.
4. Release: In this phase, quality assurance is run on the product to ensure there are no bugs. User guides are finalized, and the product is released.
5. Maintenance: Once the product is available to the public, the team will stay ready to provide training or support for any problems that may occur
6. Retirement: If for any reason the product is no longer needed, the project will be retired.

## Funding Requirements

Due to the availability of many free resources, the environment, licensing, and tools will be at no cost to us. We will be creating the project in Jupyter Notebooks using free python libraries, and the data set comes from Spotify's API. The only area that will require funding requirements is the team that will be developing the project. The overall estimated budget will be $100,000. At the estimated timeframe of 3 months, and the average pay rate of our developers at $40 an hour, we are looking at $19,200 per team

member assuming a 40-hour work week. With 3 team members, this comes out to about $60,000. The project manager will be making $48 an hour which comes out to $23,000. This leaves $17,000 for ongoing maintenance after release.

## Data Precautions

The datasets we will be using in this project are in the public domain and were downloaded from Kaggle.com. Due to this, no precaution is needed when dealing with the raw datasets. However, any cleaned data, insights, or visualizations will be considered proprietary information and precautions must be taken when handling them. No information should be shared with anyone outside the company and NDAs will be required for anyone working on the project.

## Developer's Expertise

The developer chosen to head this project has a Bachelor's in computer science. They also have five years of experience working with machine learning algorithms and creating data products like this project. With experience and education, I believe the developer is a perfect fit for our project.

# Part B: Project Proposal

The project proposal should target your client's technically savvy IT (Information Technology) professional leadership. Use appropriate industry jargon and sufficient technical details to describe the proposed project and its application. Remember, you're establishing the technical context for your project and what it will accomplish for the client. Typically, this section is 8 – 10 pages. **Write everything in the future tense.**

## Problem Statement

A new feature is becoming commonplace in the world of music streaming, music recommendation. By utilizing machine learning algorithms, we can now predict what music a specific user will like with good accuracy. With this, we can then recommend music that users have not heard before and keep them entertained and using our product.

Unfortunately, we are one of the last in our domain to adopt this new technology. As of now, we are losing monthly users, and all our competitors that have seen a rise in their monthly users have a music recommendation system. Utilizing music recommendations in our platform will almost certainly result in more monthly users and more monthly revenue.

## Customer Summary

- Describe the client (or customers).
- Describe why your proposed *application* (a *data product* in the task directions) will resolve the problem successfully.

While we will be building this data product for our company, the main customer will be the end user. This will be someone using our platform to stream music; preferably looking for something new or fresh to liven up their music playlist. This application will solve their problem successfully by analyzing their listening history and recommending music with similar qualities.

It will solve our company's problems as well. Right now, we are looking at a downturn in business because we are falling behind our competitors in which features we are offering. By adopting this feature, we will see an upturn in business and will be back or above our past performance.

## Existing System Analysis

Currently, our platform offers the ability to stream music for a fixed monthly price, with the added features of creating personalized playlists, liking others' playlists, and searching songs by title and artist. In the past, this was perfectly adequate to keep up with the competition. With the rise of new technologies, however, it is quickly becoming less than enough to stay in business. With the addition of a music recommendation system to our platform, we will be back to the current level our competitors have set for what our consumers expect from a music streaming platform.

This new system will be non-invasive to our current platform. Nothing in the current system will need to be changed except for the addition of a new tab in our user interface. Since we already record users' listening histories, everything in the new data product will utilize what we already have in place.

## Data

The data will be collected from Spotify's API. The testing and training dataset that will be used in the production of this project will be downloaded from a GitHub repository by Brice Vergnou(2021). According to Vergnou (2021) and Spotify documentation, the following is a description of the dataset:

"From Spotify's API documentation :

- **acousticness**: A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
- **danceability**: Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is the least danceable and 1.0 is the most danceable.
- **duration_ms**: The duration of the track in milliseconds.
- **energy**: Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy.
- **instrumentalness**: Predicts whether a track contains no vocals. "Ooh" and "aah" sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly "vocal". The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.
- **key**: The key the track is in. Integers map to pitches using standard Pitch Class notation. E.g. 0 = C, 1 = C♯/D♭, 2 = D, and so on.
- **liveness**: Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides a strong likelihood that the track is live.
- **loudness**: The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing the relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typically range between -60 and 0 dB.
- **mode**: Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.
- **speechiness**: Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audiobook, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections

or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.

- **tempo**: The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, the tempo is the speed or pace of a given piece and derives directly from the average beat duration.
- **time_signature**: An estimated overall time signature of a track. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure).
- **valence**: A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry)."

In addition to these attributes, there is an added column of like and dislike, with a 1 denoting the song is liked and a 0 denoting that is disliked. Our product will take the qualities found in the dataset and create categories of music with similar qualities. It will then filter through the testing dataset and create categories as well. If a category from the training set has mostly liked songs, it will recommend music from that category from the test set.

Inevitably, there will be anomalies within the data that we use. If any entry has incomplete or invalid attributes, we will simply remove the entry. Outlying data points will be kept due to the nature of music preference. After the release, we will switch to our records of user listening histories.

# Project Methodology

To develop this product, we will be using the agile methodology. This is a good fit for our product because it provides a development environment that is adaptable and fosters good communication. In agile methodology, there are generally 6 phases that a project goes through. Below I will outline each phase and what that entails.

1. Concept:
    - The project manager or product owner will outline the scope of the project.
    - They will sit with senior management to discuss key requirements, and scope expectations, and prepare documentation.
    - Product owner will estimate the time and cost of the project.
2. Inception:
    - Product owner will choose assets to be a part of the production team.
    - The team will create a very minimal user interface and start laying the groundwork for the back end to set a foundation to start fully completing the product.
    - Stakeholders will give feedback to fully recognize all requirements that need to be met.
3. Iteration:
    - The team will start doing 1–2-week sprints to complete the product.
    - The first sprint should have a very minimal but functional product as the objective.

- In further iterations they will build out the functionality with the requirements and stakeholder feedback as objectives.

4. Release:
    - The team will run quality assurance on what is now a product that meets all stakeholder requirements.
    - If any bugs are found, they will be fixed by the team.
    - The user guide will be created, and documentation will be completed.
    - The product will be released.
5. Maintenance:
    1. With the product deployed, the team will stay ready to fix any bugs that may arise from the product being distributed to a wide market
6. Retirement:
    2. If the product ever becomes outdated or unneeded, the product will be retired and the support will be discontinued.

# Project Outcomes

Throughout the development of this product, there will be many deliverables at every level. The first deliverables will be the statement of scope and the requirements document in the concept phase. In the inception phase, the deliverables will be the minimal user interface and code infrastructure. Iteration will provide the main deliverable, which will be a fully functional product with a user interface, capable of recommending music to users accurately. Another deliverable from this phase will be the product documentation and user guide.

# Implementation Plan

Once we have a functional product, we will need to integrate it into our current platform. Our general plan will be to create a tab that the users can click on to take them to the recommendation system. There they can see playlists created for them based on their current playlists. In addition, we will have functionality that will make it possible to long-click on a song and we will recommend music based on that.

After we have the user interface fleshed out, we will slowly roll out the new tab to a small group of users at a time; no more than 1000 at first, and eventually snowballing upwards as more bugs are found and dealt with. We will keep releasing it to new users until everyone has the new version. During this, we will be accepting feedback to find any latent bugs within our new system.

# Evaluation Plan

At each phase of our development, our work must be validated and verified. After the team has finished the sprint dealing with the simple user interface and code framework, we will perform integration testing to make sure they interact correctly. After each sprint, the new functionality will undergo unit testing, and the whole system will undergo integration testing to ensure that the new code works well on its own and as a part of the system. We will also have constant feedback from senior leadership to make sure we are still on track.

At the end of the project lifecycle, we will be looking for the algorithm to have at least a 70% success rate. This means we are looking for at least 7 out of 10 songs that are recommended to have a 1 in the liked field. Once this is achieved and the product is complete, we will begin user testing to ensure the system is user-friendly and the algorithm works at a larger scale.

## Resources and Costs

Most of the costs from this project will come from the development team. We estimate that it will take 3 months to complete this project. With a 40-hour work week, this comes out to 480 hours of work per resource. Below you will find the detailed and final cost for all human resources for this project.

Human Resources:

- Project Manager ($48 an hour for 480 hours) comes to a total of $23,040
- 3 Developers ($40 an hour for 480 Hours) comes to a total of $57,600
- A total cost of $80,640 will be needed for the development team.

For the development environment, we will be using Jupyter notebook and free python libraries on computers that the company already owns. The only costs coming from this will be the cost of utilities such as internet, electricity, etc. We are allotting a budget of $2000 for all utility costs. The application will be run on the database that we already own and run and will require no extra funding.

## Timeline and Milestones

The project is estimated to run for 3 months. Below is a detailed table of how long each step in the development will take.

| Milestone | Start and End Dates |
|---|---|
| Requirements and Scope | January 1 – January 7 |
| Mock GUI and Framework | January 8 – January 15 |
| Algorithm Acceptance | January 16 – February 13 |
| GUI completed, Algorithm Integrated | February 17 – February 28 |
| User Testing | March 1 – March 7 |
| Roll out completed | March 8 – March 31 |
| Deployment | Completed |

# Part C: Application

The files needed to run the application are as follows:

- C964 Application.ipynb
- C954 Documention.DOC
- train.csv
- test.csv
- predictions.csv

# Part D: Post-implementation Report

## A Business (or Organization) Vision

Before we finished this product, we were falling behind our competitors. The market started leaning towards music recommendation, but up until that point, we had no system in place to recommend music to our users. We had the data to accomplish this, but until now it was going largely unused.

The system we have created will take our users' data and process it to recommend music in an informed way. This ensures that we are staying up to date on what features consumers are looking for in a music streaming platform. By leveraging decision trees that put songs into different classifications, we can understand which songs are similar to each other and create a better experience for our users.

The user will be able to use this directly from the app, however, it is currently being used through Jupyter notebooks. To use it, we have datasets downloaded. We run the test and training datasets through the algorithm. You can then type the id of a song to see if the system will recommend it.

## Datasets

The system is using a training and testing dataset acquired from Vergnou(2021). A link to the data can be found in the sources section and a sample will be shown below.

| | danceability | energy | key | loudness | mode | speechiness | acousticness | instrumentalness | liveness | valence | tempo | duration_ms | time_signature | title |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.486 | 0.881 | 2 | -5.623 | 0 | 0.0474 | 0.024400 | 0.000000 | 0.429 | 0.6670 | 144.997 | 480707 | 4 | A Little Piece of Heaven |
| **1** | 0.356 | 0.960 | 4 | -3.487 | 1 | 0.1040 | 0.014200 | 0.000000 | 0.209 | 0.0857 | 125.921 | 204733 | 4 | Helena |
| **2** | 0.487 | 0.949 | 2 | -4.217 | 0 | 0.0798 | 0.000429 | 0.000029 | 0.231 | 0.4060 | 110.020 | 352427 | 4 | Afterlife |
| **3** | 0.366 | 0.963 | 11 | -5.301 | 0 | 0.1420 | 0.000273 | 0.012200 | 0.115 | 0.2110 | 137.114 | 366213 | 4 | Knights of Cydonia |
| **4** | 0.412 | 0.920 | 9 | -4.852 | 0 | 0.0575 | 0.000456 | 0.002600 | 0.110 | 0.4550 | 93.162 | 227440 | 4 | Hysteria |

The training dataset has an additional column named like. It denotes if the song was liked or disliked and is what is used to train the model. After running the test dataset through the system, a copy of the dataset named prediction is created with the additional like column. This will show if the system thinks the user will like or dislike the song.

There were no null values, which are shown in the figure below, and no unneeded columns as every attribute about a song are important when deciding if the user will like it.

```
training_set.isnull().sum()
```

```
danceability        0
energy              0
key                 0
loudness            0
mode                0
speechiness         0
acousticness        0
instrumentalness    0
liveness            0
valence             0
tempo               0
duration_ms         0
time_signature      0
title               0
first_artists       0
all_artists         0
like                0
dtype: int64
```

Once the data was ready to be processed by the algorithm, we dropped the title and artist fields to keep the values numerical. We then split the dataset further using the function train_test_split() and dropped the like column from the test subsection to keep the test unbiased.

## Data Product Code

For the development of this system, we used Jupyter notebook and python. The specific libraries we used were, pandas for reading and writing data, matplotlib and seaborn for data visualization, and scikitlearn for data processing and prediction. First, we imported the datasets using the pandas function “pd.read_csv()”. Once the datasets were loaded into the application we utilized the .head() function from pandas to view the first 5 entries from each dataset. We also used .info() to see a detailed view of the columns, .decribe() to see the count and five-number summary, and the function .isnull().sum() to see if there were any null values in the dataset.

For the descriptive methods, we leveraged seaborn and matplotlib to create three visualizations. First, we created a heatmap that showed the attribute’s relationship to whether they were liked or disliked. This showed that songs that were higher in “acousticness” and “instrumentalness” were more likely to be liked. Because of this correlation, our second visualization was a joint histogram showing the relationship between “acousticness” and the liked column specifically. Finally, we created a KDE plot for each attribute about the like column. This showed the shape of density that songs fell into when comparing how high a value was and if it was liked or not. Many provided only vague insights, but the assumption of high acoustic qualities being liked was further supported.

For the non-descriptive method, we used a random forest classification algorithm from sklearn. We first split the training dataset using the train_test_split() function to perform training and testing on our forest classifier. We fit the data to the x and y training subsections using the .fit() method. We then found the mean absolute error of the current model to find the performance.

Once the performance was acceptable, we ran the testing dataset through the model. We then created a third dataset which was a copy of the testing dataset with an added like column. With the results from model.predict(testing_set), we filled in the like column. Finally, we were able to use this new CSV file with predictions to search songs by index to see if the algorithm would recommend them.

# Objective (or Hypothesis) Verification

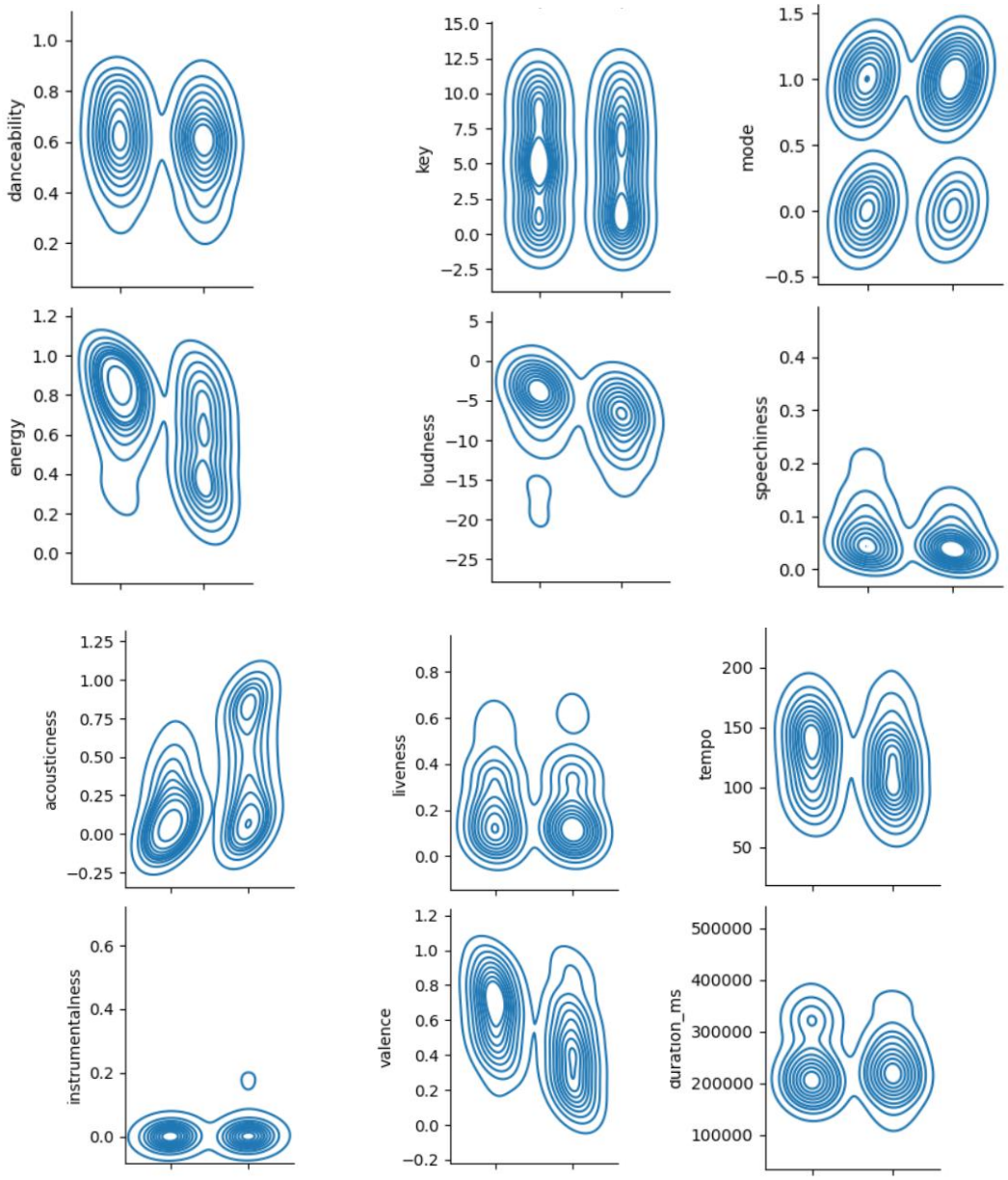
The objective of this project was to create a system that could accurately recommend music based on a user's listening data. The hypothesis before going into this project was that a random forest classification algorithm would be a good fit and would have a success rate of 70% or above.

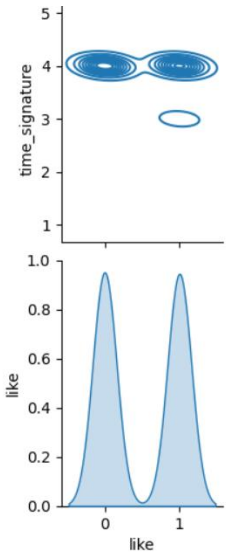
After training our random forest classifier, we came to a mean absolute error of 0.18. This means the model was predicting correctly 82% of the time. Since this is above our projected outcome, I believe our hypothesis was correct.

# Effective Visualization and Reporting

- Describe how the descriptive method(s) and visualizations supported your non-descriptive method(s) development process. Items discussed should include:
    - Data exploration.
    - Data analysis.
    - Data summary.
    - Analysis application of three visualizations (include examples of all three).

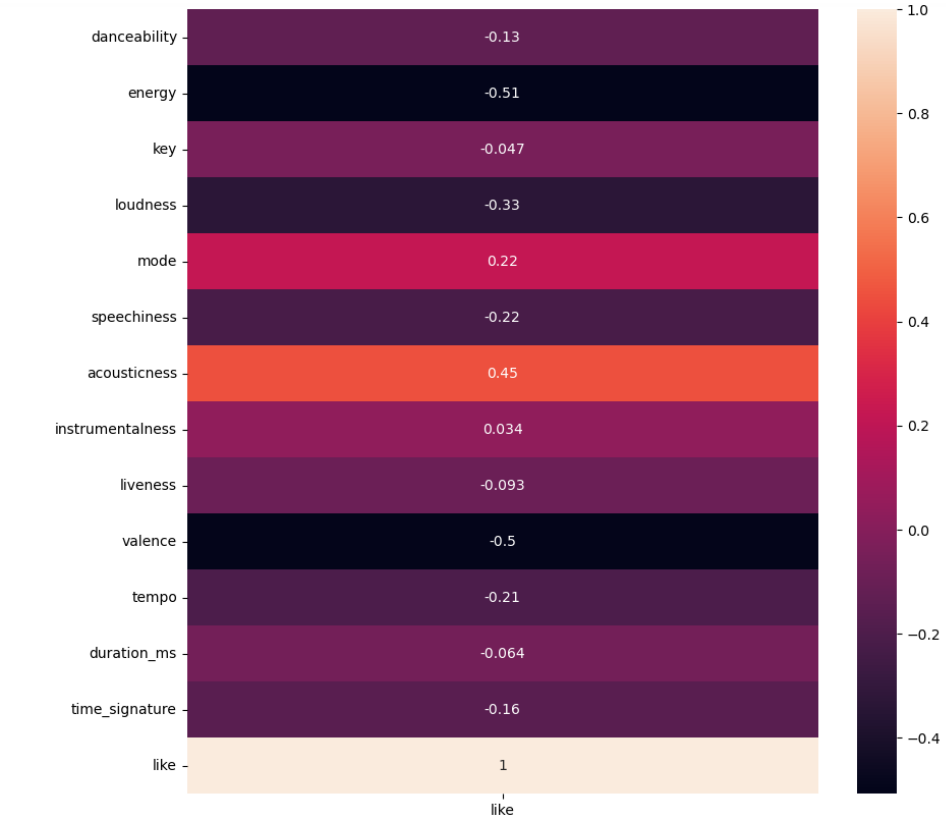
To see an overview of how the data related to our key value, if a song is liked or disliked, we created a KDE chart for each column as it related to the like column.


danceability
key
mode
energy
loudness
speechiness
acousticness
liveness
tempo
instrumentalness
valence
duration_ms



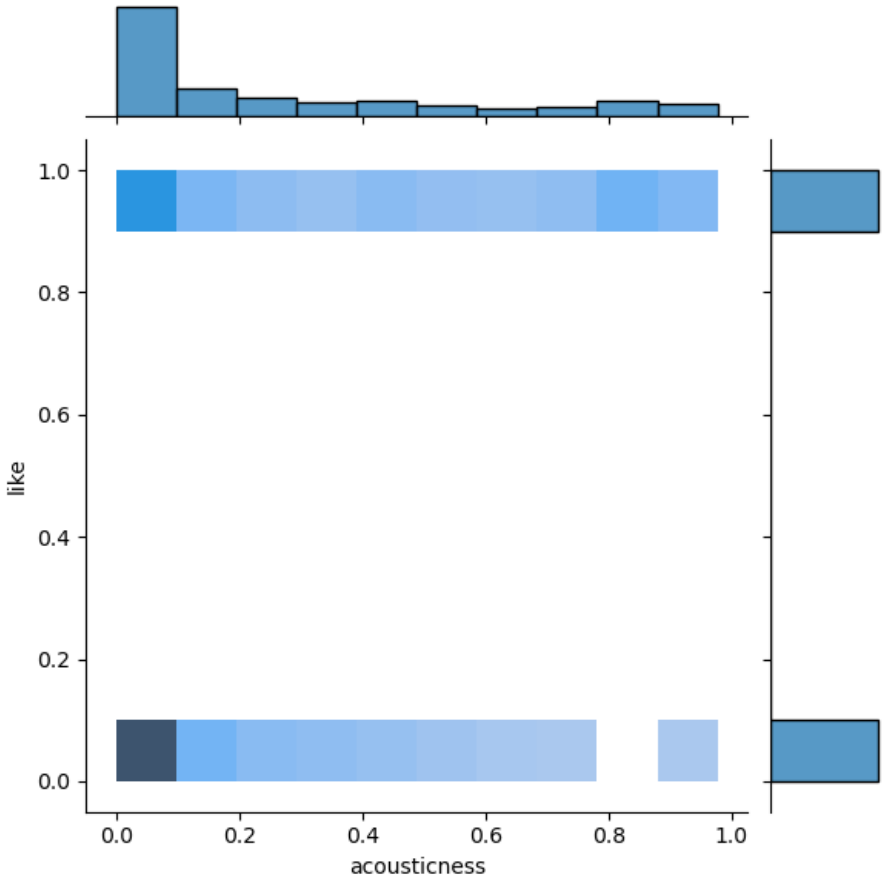

With these images, we were able to see how each attribute affected whether a song was liked or disliked and the distribution of this interaction throughout each song. Many of the attributes seem to have little effect, but in the acousticness, mode, and instrumentalness charts, we can see that the like column has slightly more in the higher values. We can also see that the valence, loudness, and speechiness graphs, show a trend where lower values are preferred.

To see this information in a slightly different way we created a heatmap with all the attributes compared to the like column. This can be seen below.



This also shows that there is a positive correlation with acousticness, intrumentalness, and mode and a negative correlation with the other attributes. This further solidifies our assumptions.

The strongest positive correlation seemed to be with acousticness, so we created a histogram to show its impact on how a song is liked.




While it is slight, we can see that there is a larger distribution of songs with higher acousticness being liked. From this data exploration, we were able to determine that there were correlations between these values and an algorithm that classified songs into smaller categories based on numerical values would lead to an accurate prediction.

# Accuracy Analysis

To analyze the accuracy of our application we utilized the library sklearn. It has a function that calculates the mean absolute error. To do this it takes the difference between the expected value and the actual value. It then takes the absolute value of this difference to ensure all values are positive. Then it takes the average of all differences to find the average error rate of what is being tested. We ran the code "print(mean_absolute_error(y_vld, training_prediction))" where the "y_vld" is the actual values from a subset of the data and "training_prediction" are the predictions.

# Application Testing

As each piece of functionality was completed, it underwent unit testing to ensure the desired result was achieved. An example of when this was used was after the prediction dataset was created and the predictions were written to the CSV file. While the functionality seemed to be fine at first, it was found that every time the predictions were written to the file an extra blank column was added to the front as well. This was causing issues with the user interface because we were using the indexes of the two-dimensional array to find results and the indexes were being pushed forward after every write. To fix this we added a drop statement after every write statement for the prediction CSV file.

Once all units of the application were completed, we performed an integration test to ensure everything was working properly together. We found that while everything was running fine on its own when put together some errors were showing up. An example of this was an error resulting from objects being passed into the prediction algorithm when it was only expecting numerical values. To fix this, we added the parameter "index=False" to the to_csv function call.

# Application Files

The files needed to run the application are the main file which includes the entire application and the CSV files which include the datasets. They are as follows:

- C964 Task 2 Submission
    - test.csv
    - train.csv
    - predictions.csv
    - C964 Application.ipynb

In addition to these files, you will need to have python and Jupyter notebooks downloaded, along with the following libraries: pandas, seaborn, matplotlib, and sklearn.

# User Guide

Step 1: Download the files C964 Application.ipynb, test.csv, train.csv, predictions.csv, and C964 Documentation Word document. Also make sure you have python and Jupyter notebooks downloaded, as well as the following python libraries: pandas, seaborn, matplotlib, and sklearn.

Step 2: Open the file C964 Application.ipynb by opening Jupyter Notebooks, navigating to the directory you saved the file in, and clicking open.

Step 3: Left-click in line 1 and press ctrl-enter.

Step 4: Go to your download of the C964 Task 2 Submission. Right-click on the file train.csv and choose copy as path. In line 2 of the application, you will see the following code: training_set = pd.read_csv(r, encoding= 'unicode_escape'). Right after the r in between the parentheses, paste the train.csv path.

Step 5: Go to your download of the C964 Task 2 Submission. Right-click on the file test.csv and choose copy as path. In line 2 of the application, you will see the following code: testing_set = pd.read_csv(r, encoding= 'unicode_escape'). Right after the r in between the parentheses, paste the test.csv path.

Step 6: Left-click in line 2 and press ctrl-enter.

Step 7: Left-click in line 3 and press ctrl-enter.

Step 8: Left-click in line 4 and press ctrl-enter.

Step 9: Left-click in line 5 and press ctrl-enter.

Step 10: Left-click in line 6 and press ctrl-enter.

Step 11: Left-click in line 7 and press ctrl-enter.

Step 12: Left-click in line 8 and press ctrl-enter.

Step 13: Left-click in line 9 and press ctrl-enter.

Step 14: Left-click in line 10 and press ctrl-enter.

Step 15: Left-click in line 11 and press ctrl-enter.

Step 16: Left-click in line 12 and press ctrl-enter.

Step 17: Left-click in line 13 and press ctrl-enter.

Step 18: Left-click in line 14 and press ctrl-enter.

Step 19: Go to your download of the C964 Task 2 Submission. Right-click on the file predictions.csv and choose copy as path. In line 15 of the application you will see the following code: prediction_set = pd.read_csv(r, encoding= 'unicode_escape'). Right after the r in between the parentheses, paste the predictions.csv path.

Step 20: Left-click in line 15 and press ctrl-enter.

Step 21: Left-click in line 16 and press ctrl-enter.

Step 22: In line 17 you will see the following code: prediction_set.to_csv(r, index=False). Right after the r in the parentheses, paste the prediction.csv path. Left-click in line 17 and press ctrl-enter.

Step 23: Left-click in line 18 and press ctrl-enter.

Step 24: In line 19 of the application you will see the following code: song_index = . Enter a number from 0 to 270 to see how the algorithm classified a song from the dataset.

Step 25: Left-click in line 19 and press ctrl-enter.

Step 26: Left-click in line 20 and press ctrl-enter.

Step 27: Left-click in line 21 and press ctrl-enter. This final line will print out the song name, artist and like value associated with the index you chose in line 19.

## Summation of Learning Experience

During my time working on receiving a bachelor's in computer science from Western Governors University, I learned a variety of things that helped me throughout this project. My experience working on various programming projects helped me develop the necessary skills to plan, code and debug a project of this nature. Due to a project involving python, I had the necessary foundation for the language to complete most of this application.

While I had a strong foundation from my degree, there were many things that I had to figure out during this project's development. Where I found the most help was in the documentation for the libraries I was using. Since I have a strong foundation in computer science, if I was struggling to understand a concept or functionality of a method, I could investigate the documentation and see how it was working to find the answer I needed. Many libraries also offer simple tutorials to help you get started, including pandas, seaborn, matplotlib, and sklearn.

Working on this project has taught me that learning new skills never ends, and learning is a lifelong process. No matter how much you know in a specific language or field, there are always areas that you won't know or new technologies that show up. This is simply a part of our field, and you must be willing and able to continually learn to stay at your best.

# Sources

Vergnou, B. (2021). spotify_recommendation. Retrieved from https://github.com/Brice-Vergnou/spotify_recommendation/blob/master/data/data.csv.