INFORMATION SYSTEMS EDUCATION JOURNAL

**Volume 20, No. 3**
June 2022
ISSN: 1545-679X

## *Special Issue*
## *Teaching Cases*



iscap
Information Systems & Computing Academic Professionals

The **Information Systems Education Journal** (ISEDJ) is a double-blind peer-reviewed academic journal published by **ISCAP** (Information Systems and Computing Academic Professionals). Publishing frequency is six times per year. The first year of publication was 2003.

ISEDJ is published online (https://isedj.org). Our sister publication, the Proceedings of EDSIGCON (https://proc.iscap.info) features all papers, panels, workshops, and presentations from the conference.

The journal acceptance review process involves a minimum of three double-blind peer reviews, where both the reviewer is not aware of the identities of the authors and the authors are not aware of the identities of the reviewers. The initial reviews happen before the EDSIGCON conference. At that point papers are divided into award papers (top 15%), other journal papers (top 25%), unsettled papers, and non-journal papers. The unsettled papers are subjected to a second round of blind peer review to establish whether they will be accepted to the journal or not. Those papers that are deemed of sufficient quality are accepted for publication in the ISEDJ journal. Currently the target acceptance rate for the journal is under 40%.

Information Systems Education Journal is pleased to be listed in the Cabell's Directory of Publishing Opportunities in Educational Technology and Library Science, in both the electronic and printed editions. Questions should be addressed to the editor at editor@isedj.org or the publisher at publisher@isedj.org. Special thanks to members of ISCAP/EDSIG who perform the editorial and review processes for ISEDJ.

# INFORMATION SYSTEMS EDUCATION JOURNAL

## Editors

*Teaching* Case

# Bracketology: Predicting Winners from Music March Madness

Kevin Mentzer
kevin.mentzer@nichols.edu
Nichols College
Dudley, MA 01571, USA

Zachary Galante
zachgalante62@gmail.com
School of Information
University of California, Berkeley
Berkeley, CA 94720, USA

Mark Frydenberg
mfrydenberg@bentley.edu
Computer Information Systems Department
Bentley University
Waltham, MA 02452, USA

## Abstract

Organizations are keenly interested in data gathering from websites where discussions of products and brands occur. This increasingly means that programmers need an understanding of how to work with website application programming interfaces (APIs) for data acquisition. In this hands-on lab activity, students will learn how to gather data from several prominent websites using APIs and then build predictive models using that data. Unlike popular challenges on competition sites such as Kaggle where challenges often supply the data, this project emphasizes the data acquisition step of the analytics lifecycle. Working with data from Spotify, YouTube, and Twitter, students will fill out a music based March Madness bracket to predict the winner of the annual Locura De Marzo, a popular middle and high school Spanish competition. By becoming familiar with the data available from each site, through the analysis of the JSON formatted data returned by the APIs, students will be able to explore which features of a song might lend themselves to higher voting by high school students in order to build better prediction models.

**Keywords:** Bracketology, March Madness, Python, Data Science, Data Analytics, APIs, Hit Song Science

### 1. INTRODUCTION

The most well-known March Madness event is the annual NCAA basketball tournament that occurs each spring when 68 college teams compete for the national championship. The competition is commonly displayed as a bracket showing each division with the head-to-head matchups comprising the tournament. The winner of each head-to-head matchup moves on to the next round while the loser goes home. The study of predicting winners in this format is called

Bracketology. However, March Madness isn't just for basketball: similar tournaments, usually based on public voting, have occurred for science fiction TV shows (*Io9's March TV Madness*, n.d.), the Cooking Channel's Best College Eats (*Bracket Battle Best College Eats*, n.d.), and the Consumerist's Worst Companies in America (*Here Are Your Contestants For The 2013 Worst Company In America Tournament!*, 2013) where Electronic Arts trounced Bank of America in the final round earning them the title of Worst Company in America in 2013. In each of these tournaments the general public was asked to vote for the best (or worst) in each head-to-head matchup.

Consider the following challenge: if you were given 16 songs and asked to predict which song middle and high school students would select as their favorite, do you think you could do it?

This project will look at another, non-sports related, bracket style tournament run by middle school Spanish teacher Señor Ashby titled Locura De Marzo (*Locura De Marzo 2021*, n.d.). In this event, 16 songs go head-to-head in a bracket style competition for best song.

Using data that you will gather from Spotify, YouTube, and Twitter, you will develop models to fill out the Locura De Marzo bracket and to see how well you can predict the winner. The model will be trained using data from prior years' competitions.

## 2. LEARNING OUTCOMES

Before starting this project you should be familiar with the following

- A basic level of Python as you would obtain in an Introduction to Python class.
- Familiarity with installing libraries in your Python environment.
- Familiarity with Python Pandas (refer to this site for a good overview of Pandas: https://www.learndatasci.com/tutorials/python-pandas-tutorial-complete-introduction-for-beginners/).
- General familiarity with Spotify, YouTube, and Twitter, from a user's perspective.
- A general understanding of predictive modeling techniques such as linear regression and decision trees.

After completing this project, you will be able to:

- Automate data gathering from websites using several different APIs
- Import Python libraries and call methods in those libraries
- Generate a correlation matrix to explore your data for feature identification
- Build basic predictive models including a linear regression and random forest model
- Interpret results from the regression and random forest models
- Create a bracket-style elimination model

## 3. PROJECT DESCRIPTION

In this project, you will build a Python application that carries out the process of acquiring data, building a model, obtaining characteristics, and performing linear regression and machine learning analysis to predict the winner of a music tournament. Your instructor may provide a starter code file to help you develop or run this solution.

**Discussion Questions**

Each step of the project ends with questions for your consideration as you complete this project. Discuss the questions with your group, prepare responses for class discussion, or share your responses in a format as specified by your instructor.

Figure 1 provides an overview of the work flow and data flow for this project.

Your solution will follow these steps:

Setup Step (Step 0). If you do not already have free accounts on Spotify, YouTube, and Twitter, then you need to create those accounts. Next you will create Developer Accounts for these social media sites. You will need these to gather data about the songs and artists.

Step 1. Collect data on the past results of the Locura de Marzo tournaments. Manually enter it into a Pandas DataFrame. Features include the song title, artist, and performance of that particular song in the tournament.

Step 2. Call Application Programming Interfaces (APIs), passing collected song titles, and artist information to gather additional information. The Spotify API provides more advanced features on the song such as liveliness and tempo. The YouTube and Twitter APIs collect additional data about the song and artist's exposure on social media. The program uses this complete dataset to train the model.

Step 3. Create a correlation matrix to identify key features related to the target variables.

Step 4. Use the data and previous findings to train a Linear Regression model. Explore how machine learning can enhance this solution.

Step 5. Build a bracketed model solution to predict which songs will win in the tournament.

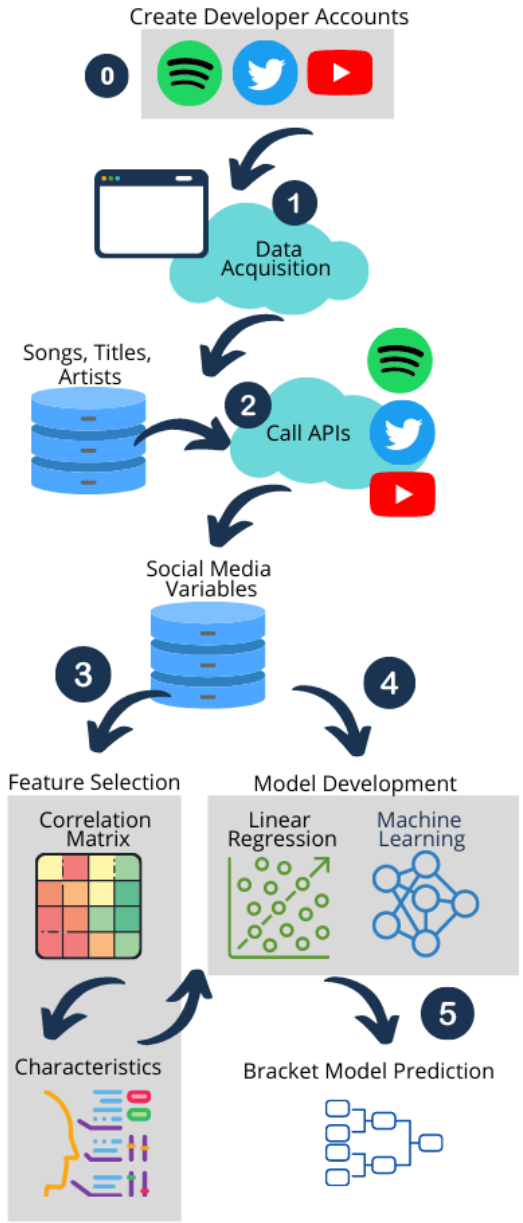



**Figure 1. Workflow and Data Flow for Bracketology Project**

**Setup Step (Step 0 on Figure 1). Create Developer Accounts**

Application Programming Interfaces (APIs) are callable functions that allow developers to access data from a website for use in other applications. Not all websites have APIs, and sometimes developers resort to screen scraping, or gathering information from a webpage based on its position on the page when an API is not available. Calling an information provider's API is always the preferred method when gathering large amounts of data. While most large social media sites offer APIs to developers, they frequently restrict the amount of type of data that can be gathered from the website.

To work with APIs, you usually have to create a developer account and get a license key, a unique identifier that allows the API provider to track the amount of information you request. These keys are unique for each user and the different websites have their own processes to obtain these keys.

In this project you will obtain data from Spotify, YouTube, and Twitter by calling their APIs. Follow the instructions in the paragraphs below to create developer accounts so you can access each service's API.

Follow the steps on these sites to create your developer accounts for the three social media sites:

- https://developer.twitter.com/en
- https://developer.spotify.com/
- https://developers.google.com/youtube/

***Things to watch out for***

When filling out the questions to submit to the various social media platforms, make sure you explain that you are requesting access as a student in order to do a social media project for your class. You may want to include your class name and your professor's name. It is preferred that you use an email address with a .edu extension in order to support your documentation.

***Discussion Questions***

1. *Looking at the Spotify API documentation, discuss which pieces of data could help with building a predictive model for song success.*
2. *Looking at the YouTube documentation, which fields could be used to predict popularity of a song?*

**Step 1. Data Acquisition**

Gather information about songs, titles, artists, and song performance in prior Locura de Marzo

tournaments from the SenorAshby.com website. Manually enter it into a Pandas DataFrame.

***Things to watch out for***

Some years the songs may have total number of votes while other years may include percent of votes received or even simply whether the song won or not. Consider how you will compare various years and make sure you use the same measure across years. This may require you to convert total vote count to a percentage or some other transformation.

***Discussion Questions***

1. *Looking at the data you were able to gather from the competition website, what fields provide the most insight and which fields do you think you could ignore?*
2. *Considering the data you examined in the API documentation along with the competition website, how would you ensure that the song on the competition website is the official song on the other sites (Spotify and YouTube)?*

**Step 2. Call APIs**

Call APIs from Spotify, Twitter, and YouTube to obtain additional information about these songs.

Your application will also need code to access and interact with the data that the APIs provide. Open-source libraries written in Python exist that provide this capability. The next sections describe those libraries and the data they make available.

To include a library in your Python application, you need to install it in your environment and then specify an *import* statement to include it in your project.

**Spotipy.** To interface with Spotify we will use the Spotipy library. Review the documentation for the Spotipy project here: https://spotipy.readthedocs.io/en/2.18.0/. Scroll down to the API Reference section of the documentation and you can see that there are methods for getting data related to Artists, Albums, Playlists, and Tracks, just to highlight a few.

Appendix 1, Figure 1 shows a code sample for accessing the Spotipy API along with the expected results. You can use this code to validate that your developer keys are working properly. The output from each of these calls is returned in a JSON format. This is a common format used with APIs, so if you are unfamiliar with the format then we suggest you read more about it at https://www.json.org/.

Since we are concerned with predicting which song will win in a head-to-head matchup with another song, we will focus on song level data (called a Track in the music industry). Looking at the documentation for the Track method we see the following:

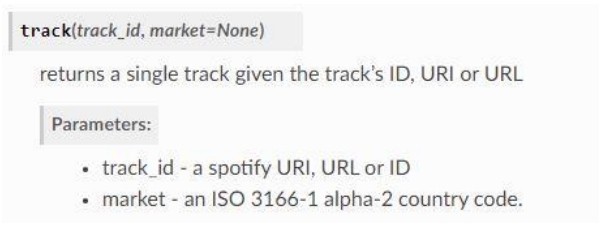



**Figure 2. Spotipy Track Method**

In order to make the call to the track method, we need to supply the track_id and an optional market. The track_id is Spotify's unique id for a specific song in their database.

The Locura de Marzo website makes track retrieval easier by supplying a playlist of each song in the competition. This allows us to call the API that retrieves the song details using that playlist which eliminates the need to loop through each song individually. This also reduces the calls to Spotify to 1 call per year of the competition. It is best practice to minimize the number of calls to the APIs in order to reduce your chances of hitting rate limits.

| Spotify Features | Description |
|---|---|
| Release Date | Date song was released |
| Length | Length of the song in milliseconds |
| Popularity | Scale of how popular the song is (0-80) |
| Acousticness | Confidence measure if the song is acoustic |
| Danceability | Suitability for dancing |
| Energy | Measure of intensity and activity |
| Instrumentalness | Predicts whether a track contains no vocals |
| Liveness | The presence of an audience in the recording |
| Loudness | Loudness in decibels |
| Speechiness | Presence of spoken words |
| Tempo | Tempo in beats per minute |
| Key | Main group of pitches of a track |
| Mode | Modality of a track (1= Major, 0 = Minor) |

**Table 1. Spotify Features**

The code in Appendix 1, Figure 2 validates that the list of songs can be retrieved using the playlist as supplied by the competition website.

Appendix 1, Figure 3 shows a code sample for retrieving track details for each of the songs in the playlist. The getTrackInformation() function is called once for each song in each playlist. The result of this call gives us the features found in Table 1.

**Tweepy.** In order to access Twitter information for each artist we need to find the handle for each. This is a manual lookup process. An artist's Twitter handle can be found in their Twitter profile, as shown in Figure 3.

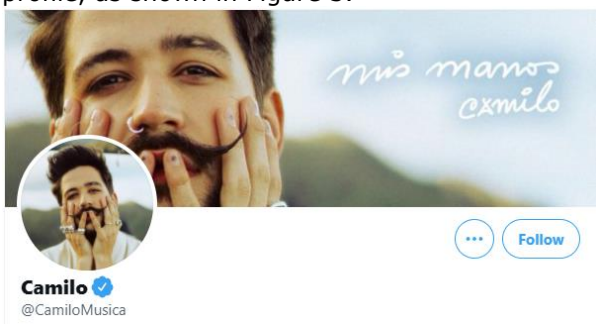



**Figure 3. Finding a Twitter Handle**

We can see that the artist's handle is @CamiloMusica. We'll use this handle below to interface with the Twitter APIs. We use the Tweepy library to interface with the Twitter APIs. Review the documentation for the Tweepy project here: https://docs.tweepy.org/en/latest/.

See Appendix 1, Figure 4 for a code snippet to collect Camilo's number of followers, a value which we will use in our model. The result of this call gives us the following features from Twitter:

| Twitter Feature | Description |
|---|---|
| Number Followers | Number of followers for an account |

***Table 2. Twitter Features***

**YouTube.** Your solution/notebook will obtain information about music videos for each song from YouTube. Google provides an official Python library for YouTube found here: https://developers.google.com/youtube/v3

The code in Appendix 1, Figure 5 validates that your YouTube credentials are correct and you are able to retrieve the total number of views for the music video for *Vida De Rico* by Camilo. Table 3 shows some of the features available for each music video.

At this point you should have been able to validate that your developer accounts for the three websites are working properly. The next step is to collect the data for each song for both the training date (i.e. prior year competitions) and the testing data (current year challenge).

| YouTube Features | Description |
|---|---|
| View Count | Number of views for the video |
| Like Count | Number of likes for the video |
| Dislike Count | Number of dislikes for the video |
| Comment Count | Number of comments for the video |
| Tags | Tags associated with the video |

**Table 3. YouTube Features**

***Discussion Questions***

1. *If you could only pick a single piece of data to predict popularity, which would it be and why?*
2. *Based on your personal knowledge of the songs, what values surprise you most? Why?*

**Step 3. Feature Selection**

After gathering data about each song from the three different sources (Spotify, YouTube, Twitter), the next step is to decide which fields to use in our models.

One way to approach this step is to consider which variables are highly correlated with our target variable. In this case our target is the percentage of votes. By examining the correlation matrix (see Appendix 2, Figure 1) we look for fields that score high (using the absolute value, so high positive or high negative) in relation to the total number of votes. We see that the feature that has the highest positive correlation with our target variable is YouTube View Count, and Like Count. Seed number appears to be our strongest negatively correlated variable followed by mode and acousticness.

***Discussion Questions***

1. *Research and discuss the limitations related to correlation matrices.*
2. *Discuss what other features you think should be included in the analysis and why they should be included.*

3. *Are there features that were selected that you think shouldn't be used in the analysis and why?*

**Step 4. Model Development**

One way to select a modeling technique would be to examine how others have solved similar challenges in a related area. In this case we can consider how others have approached trying to predict the winner of the NCAA March Madness Tournament.

Prior literature has shown that a team's Seed number is one of the best predictors of success. Our correlation matrix suggests that Seed number is important in this tournament as well. Boulier and Stekler (1999) found that by using seed alone one is able to predict winners with a 73.5% accuracy. Stekler and Klein (2012), however, stress that this approach appears to work well only for the early rounds of the tournament. One reason this is the case is that an incorrect prediction in the first round will feed into subsequent rounds, which is known as a forward propagation error.

Using a similar approach, we can look at the 2020 results (see Appendix 2, Figure 2) to see how well Seed number predicted the win. There are 11 head-to-head matchups in the 2020 competition and the higher Seed won in 11 of those 17 matchups for an accuracy of 64.71%. This can be considered an improvement over a 50% probability were we to simply use a random guess. However, using Seed number alone in predicting each head-to-head matchup assumes that we know each head-to-head matchup while this is only known (prior to the competition) for the first round.

Let us see if we can improve upon this model using the features we identified earlier as highly correlated with percent of vote. We can use a multivariate linear regression to predict percentage of vote. The equation for the multivariate linear regression is shown in Equation 1.

$$Y_i = \alpha + \beta_1 x_i^{(1)} + \beta_2 x_i^{(2)} + \ldots + \beta_n x_i^{(n)}$$

***Equation 1. Multivariate Linear Regression***

Multivariate Linear Regression is a supervised machine learning technique since the target variable is known and we build the model using that target (in this case, the target is percent of vote). Code to set up the linear regression is shown in Appendix 1, Figure 6. We use the results of the regression to predict the winners of the 2021 challenge discussed next.

***Discussion Questions***

1. *Using Seed number alone, how accurate would the model be in predicting the 2019 competition?*
2. *Find another example of how others have tried to predict the winner of the NCAA March Madness tournament and discuss the model(s) used and whether they were effective or not?*
3. *How might machine learning algorithms enhance your solution?*

**Step 5. Bracket Model Prediction**

Looking at the regression coefficients for each (see Appendix 1, Figure 7) of the independent variables we can state the following:

- As the Locura de Marzo Seed number increases, the win percentage decreases
- As the number of YouTube views increase, the win percentage also increases
- As the number of Twitter Followers increase, the win percentage also increases
- As the Spotify tempo increase, the win percentage decreases.

The first three appear to make sense. Keep in mind that the number 1 seed is a lower number that than the number 16 seed even though it would be considered the higher ranked song, so an increase in Seed number means a lower rank song. Therefore, as the Seed number increases (song has a poorer Seed number), the predicted percentage decreases.

Finally we can consider the tempo. A song's tempo is the beats per minute. Our model suggests slower paced songs are more likely to win.

Next we are able to use our model to predict the winners of each matchup in the 2021 challenge. You can see our round one predictions in Appendix 2, Figure 3.

Completing the rest of the rounds allows us to create the complete predicted bracket as shown in Appendix 2, Figure 4. Comparing our results (Figure 4a) to the 2021 actual results (Figure 4b) we can see that we correctly predicted 5 of the 8 matchups in round 1 yet only 1 of the final 4 and neither of the final matchup. Clearly there is room for improvement.

***Discussion Questions***

1. *Would you consider the regression model an improvement over random guessing? An*

*improvement over using just the Seed number? Explain your reasoning.*

2. *What would be a logical next step in trying to improve the model?*

## 4. CONCLUSIONS AND NEXT STEPS

There are many ways to improve upon this model.

First, you can expand on the data being used to build the model. There are many other variables being retrieved from the websites that weren't used in the development of the initial model. Alternatively, you could consider data that we haven't yet retrieved such as a sentiment analysis of Tweets that mention each song or artist.

Second, our model was based on the results from the first round of data and the strongest showing in the first round was predicted to win the entire tournament. This model doesn't take into account that some songs had weak or strong competitors in the first round which influenced our results. Building a model that takes into account the strength of each competitor would be a logical future step.

Third, we have built and trained a multivariate linear regression model. Based on some of your own research in answering the discussion questions, you have probably already found alternative techniques (i.e. Decision Trees, Random Forest, Neural Networks, etc.) that may be a better fit for a challenge such as this one.

We caution you from being over exuberant, or disappointed, if your model performs extremely well, or poorly, for any given year. With such a limited amount of data to train the models, we are unable to split our data into an appropriate training and testing dataset. As such, any model is most certainly going to be overfitted based on the prior year's results.

Finally, we request that you do not reach out to Señor Ashby or anyone else responsible for the Locura de Marzo challenge. The website is to support Spanish language learning and not machine learning.

## 5. REFERENCES

Boulier, B. L., & Stekler, H. O. (1999). Are sports seedings good predictors?: An evaluation. *International Journal of Forecasting*, *15*(1), 83–91.

*Bracket Battle Best College Eats*. (n.d.). Cooking Channel. Retrieved July 7, 2021, from https://www.cookingchanneltv.com/recipes/packages/bracket-battle-best-college-eats

*Here Are Your Contestants For The 2013 Worst Company In America Tournament!* (2013, March 18). Consumerist. https://consumerist.com/2013/03/18/here-are-your-contestants-for-the-2013-worst-company-in-america-tournament/

*io9's March TV Madness: Pick the greatest science fiction TV show ever made!* (n.d.). Gizmodo. Retrieved July 7, 2021, from https://gizmodo.com/io9s-march-tv-madness-pick-the-greatest-science-ficti-452908411

*Locura De Marzo 2021*. (n.d.). SeñorAshby.Com. Retrieved July 7, 2021, from http://www.senorashby.com/locura-de-marzo-2021.html

Stekler, H. O., & Klein, A. (2012). Predicting the outcomes of NCAA basketball championship games. *Journal of Quantitative Analysis in Sports*, *8*(1).

# Appendix 1. Code Samples

**Calling the Spotify API (spotipy)**

The Spotify API will gather data on the songs included in the Locura de Marzo tournament. The API allows for the collection of advanced measures for each song, including the loudness and danceability. To run this code, you will need your own client_id and client_secret values from your Spotify developer account.

```
# The highest ranked song for 2021 is Vida De Rico by Camilo.
# Use this song ID to see what data is available to us in regards to each song and
validate developer credentials.
# Import the necessary libraries
import spotipy
from spotipy.oauth2 import SpotifyOAuth
from spotipy.oauth2 import SpotifyClientCredentials

# MAKE SURE YOU INSERT THE CLIENT ID AND CLIENT SECRET BELOW
# Set up a connection using the Spotify API credentials
auth_manager = spotipy.oauth2.SpotifyClientCredentials(client_id='INSERT CLIENT
ID', client_secret='INSERT SECRET')
sp = spotipy.Spotify(auth_manager=auth_manager)

# Songs have metadata associated with them along with audio features. We make a call
 for each.
meta = sp.track('73nAK3HgQK8dak83Y2WQ8F')
features = sp.audio_features('73nAK3HgQK8dak83Y2WQ8F')

print(meta)
print(features)
```

Output:

```
{'album':{'album_type': 'single', 'artists': [{'external_urls': {'spotify':
'https://open.spotify.com/artist/28gNT5KBp7IjEOQoevXf9N'} ,
'href':
'https://api.spotify.com/v1/artists/28gNT5KBp7IjEOQoevXf9N',
'id': '28gNT5KBp7IjEOQoevXf9N', …

[{'danceability': 0.824, 'energy': 0.457, 'key': 6, 'loudness': -5.428, 'mode': 1,
'speechiness': 0.0543, 'acousticness': 0.167, 'instrumentalness': 0, 'liveness':
0.041, 'valence': 0.95, 'tempo': 87.977, 'type': 'audio_features', 'id':
'73nAK3HgQK8dak83Y2WQ8F', 'uri': 'spotify:track:73nAK3HgQK8dak83Y2WQ8F',
'track_href':
'https://api.spotify.com/v1/tracks/73nAK3HgQK8dak83Y2WQ8F',
'analysis_url': 'https://api.spotify.com/v1/audio-
analysis/73nAK3HgQK8dak83Y2WQ8F', 'duration_ms': 187427, 'time_signature':
4}]
```

***Figure 1. Calling the Spotify API Code and partial results***

**Obtaining Track List using Playlist Information**

The following code sets up a function to retrieve a list of tracks using the playlist as supplied by the Locura de Marzo website.

```
#creating a function to get the trackID
def getTrackIDs(playlist_id):
    ids = []
    playlist = sp.playlist(playlist_id)
    for item in playlist['tracks']['items']:
        track = item['track']
        ids.append(track['id'])
    return ids

# Senor Ashby supplied playlist for 2021 - you can find this list ID in the URL
when you click on the playlist link.
playlist = getTrackIDs('5uR6hSLXusr18AERyVmnbE')

print(playlist)
```

```
Output:
['73nAK3HgQK8dak83Y2WQ8F', '0GARcbxLIOmzrs0lHpuvmi', '4N7yGB3c8GXPMEeoc15Ekr',
'02dsc9B5N8BFatjGcGhk1u', '1pqnQ41XbfKjaFu6M0eGJp', …
```

Figure 2. Obtaining Track IDs for songs in a playlist.

**Obtaining Track Details using Playlist**

The following code shows how to get tracks from a playlist and add them to a list. The getTrackFeatures function obtains the features for each track.

```
def getTrackFeatures(id):
    meta = sp.track(id)
    features = sp.audio_features(id)

    # meta
    name = meta['name']
    album = meta['album']['name']
    artist = meta['album']['artists'][0]['name']
    release_date = meta['album']['release_date']
    length = meta['duration_ms']
    popularity = meta['popularity']

    # features
    acousticness = features[0]['acousticness']
    danceability = features[0]['danceability']
    energy = features[0]['energy']
    instrumentalness = features[0]['instrumentalness']
    liveness = features[0]['liveness']
    loudness = features[0]['loudness']
    speechiness = features[0]['speechiness']
    tempo = features[0]['tempo']
    key = features[0]['key']
    mode = features[0]['mode']
    time_signature = features[0]['time_signature']

    track = [name, album, artist, release_date, length, popularity, danceability, ac
ousticness, energy, instrumentalness, liveness, loudness, speechiness, tempo, key, m
ode, time_signature]
    return track

track_labels = ['name', 'album', 'artist', 'release_date', 'length', 'popularity',
'danceability', 'acousticness', 'energy', 'instrumentalness', 'liveness',
'loudness', 'speechiness', 'tempo', 'key', 'mode', 'time_signature']

playlist_features = []
for i in range(len(playlist)):
    track = getTrackFeatures(playlist[i])
    playlist_features.append(track)

df_playlist = pd.DataFrame(playlist_features, columns = track_labels)
```

Figure 3. Obtaining Track information for songs in a playlist.

**Calling the Twitter API**

The Twitter API provides much more data than just the number of followers but that is the only feature we will collect for each artist.

```
import tweepy
# assign the values accordingly
consumer_key = "INSERT CONSUMER KEY"
consumer_secret = "INSERT CONSUMER SECRET"
access_token = "INSERT ACCESS TOKEN"
access_token_secret = "INSERT ACCESS TOKEN SECRET"
# authorization of consumer key and consumer secret
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
# set access to user's access key and access secret
auth.set_access_token(access_token, access_token_secret)
# calling the api
api = tweepy.API(auth)

print(f'Number of followers: {api.get_user("@CamiloMusica").followers_count}')
```

Output:
```
Number of followers: 1483544
```

Figure 4. Calling the Twitter API

**Calling the YouTube API**

This code block shows how to obtain information about videos in a Play List.

```
# importing necessary packages
from apiclient.discovery import build
from apiclient.errors import HttpError
from oauth2client.tools import argparser

# setting up the YouTube api with the appropriate credentials
DEVELOPER_KEY = 'INSERT DEVELOPER KEY'
YOUTUBE_API_SERVICE_NAME = "youtube"
YOUTUBE_API_VERSION = "v3"
youtube = build(YOUTUBE_API_SERVICE_NAME, YOUTUBE_API_VERSION,developerKey=DEVELOPE
R_KEY)

# setting up the video request and then executing
vid_request = youtube.videos().list(
        part="statistics, snippet",
        id='qKp1f7Vn9dM')

vid_response = vid_request.execute()

print(f'Total Number of Views: {vid_response["items"][0]["statistics"]["viewCount"]
}')
```

Output:
```
Total Number of Views: 606345247
```

Figure 5. Obtaining Video Information from a Play List.

**Linear Regression**

This code sample shows how to set up a linear regression model. This code assumes that you have a dataframe called train_df with all of the data from prior years and a test_df with data about the current year's competition.

```
from sklearn.linear_model import LinearRegression

linear_model_data = train_df[['seed','viewCount', 'followers', 'tempo', 'vote_perce
ntage']]
inputs = linear_model_data.drop(columns  = ['vote_percentage'])
target = linear_model_data.vote_percentage

# training the model on the previous results
regression = LinearRegression().fit(inputs, target)

# getting the predictions for the 2021 tournament
tournament_2021_data = test_df [['seed','viewCount', 'followers', 'tempo']]
predictions_2021 = regression.predict(tournament_2021_data).tolist()

# adding those predictions to the 2021 DataFrmae
test_df ['predicted_win_percentage'] = predictions_2021
```

Figure 6. Setting up a linear regression.

**Regression Coefficients**

This sample code shows how to display the regression coefficients.

```
# Show Regression Coefficients for Locura de Marzo Seed #, YouTube View Count (per
# Million views), Number of Twitter Followers (per Million
# followers), and Spotify Tempo
print(regression.coef_)

Output:
array([-2.41596475e-02,  3.01212645e-05,  2.48028881e-03, -9.50443717e-
04])
```

Figure 7. Showing Regression Coefficients.

# Appendix 2. Additional Figures

## Correlation Matrix

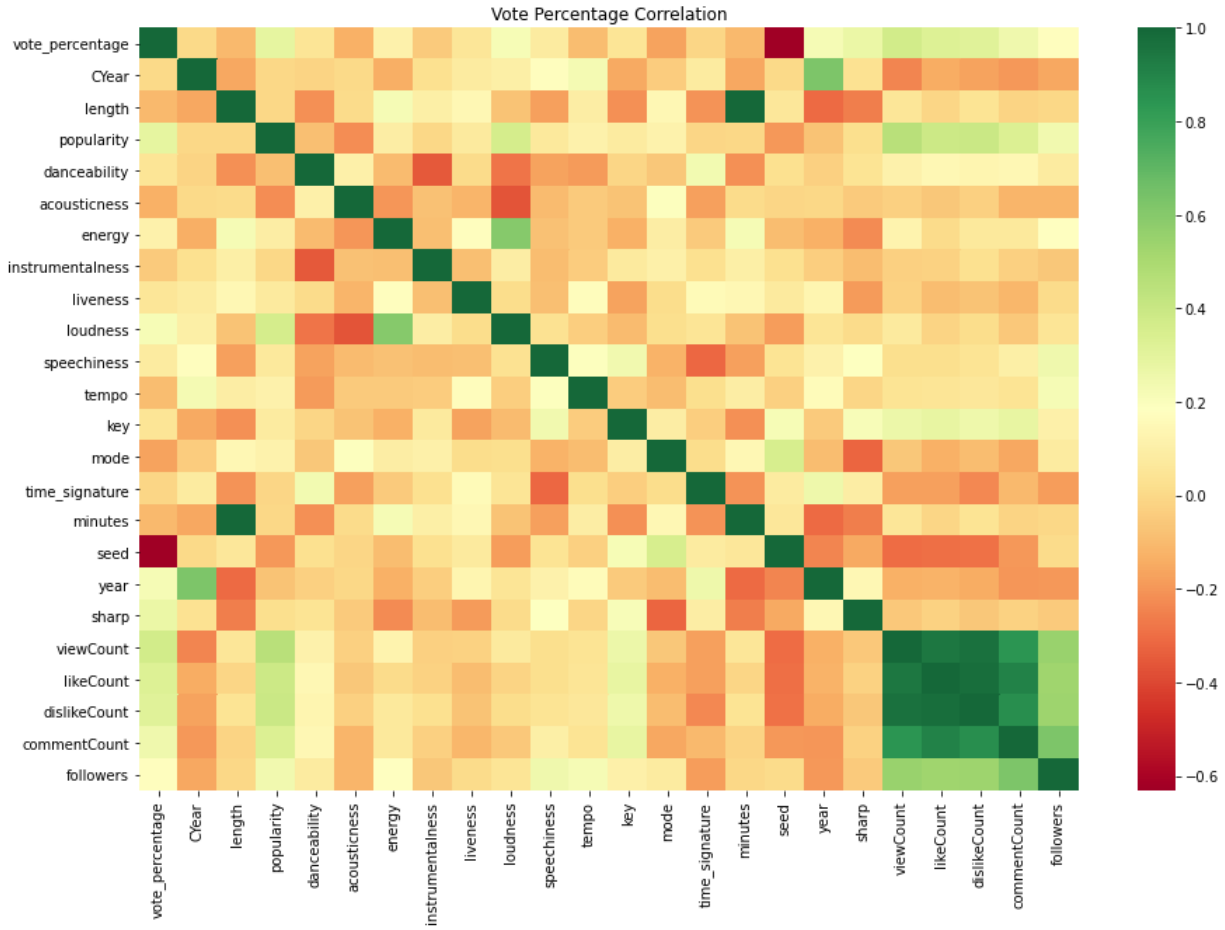



***Figure 1. Correlation Matrix***

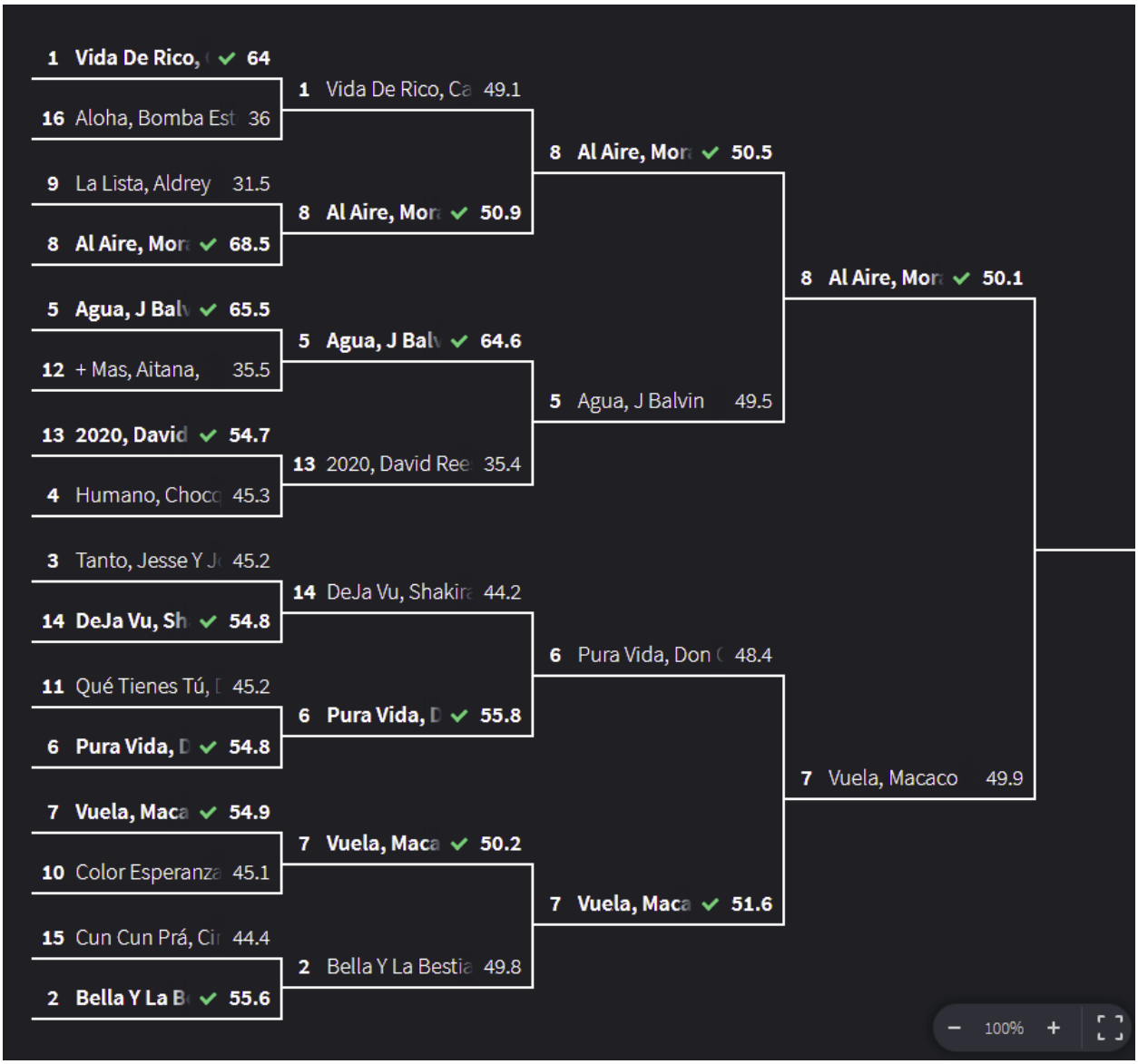

***Figure 2. 2020 Results Compared with Seed Number Prediction***

Round 1 Results

| | Winning Song | Winning Seed | Losing Song | Losing Seed |
|---|---|---|---|---|
| 0 | Vida de Rico | 1 | Aloha | 16 |
| 1 | La Bella y la Bestia | 2 | Cun Cun Prá | 15 |
| 2 | Deja vu | 14 | Tanto | 3 |
| 3 | Humano | 4 | dos mil veinte | 13 |
| 4 | Agua (with J Balvin) - Music From "Sponge On T... | 5 | + (MÁS) | 12 |
| 5 | Pura Vida | 6 | Qué Tienes Tú (feat. Jesús de Reik & Mau y Ricky) | 11 |
| 6 | Color Esperanza 2020 | 10 | Vuela | 7 |
| 7 | La Lista | 9 | Al Aire | 8 |

***Figure 3. Linear Regression Round 1 Predictions***


(a) Predicted Bracket




(b) Actual results

Figure 4. (a) Predicted Bracket and (b) Actual Results.