Movie Recommendations for Flights

Team 13
DS4A Data Engineering

Table of Contents

INTRODUCING...

Team 13

Nicole Ciccarello

Team Captain

Mike Martin

Cecil Belfon

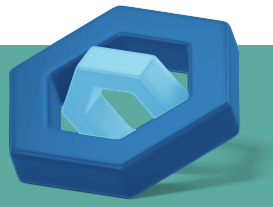
Nina Hamidli

Context

Business Problem

Airline companies have monetized almost every part of the travel experience, from luggage fees, to higher costs for extra legroom, to in-flight meals, all services that were once included with the price of the flight ticket. However, one service the airline companies have not yet charged are the in-flight movies and TV shows they offer to passengers. Offering digital entertainment is a small perk that keeps passengers entertained, or distracted, during the course of a flight, and almost all carriers offer a catalog of movie and pre-recorded TV options. According to Nerd Wallet, Delta Airlines, for example, can give passengers access to up to 300-movie titles, HBO, Showtime, and 12-channels of live TV. But of the millions of possible titles to choose from, how do airline executives decide what movies and pre-recorded TV shows to offer on flights?

The airline industry is elastic; many customers are price-driven and often choose a carrier based primarily on cost of travel to their final destination Team-13 hypothesizes that in order to help build brand loyalty, airline managers need to make better decisions on the no-cost offerings, such as mood lighting, easy self-service menu options, and a strong catalog of movie and pre-recorded TV offerings. This study will focus on the latter suggestion of entertainment; we will make recommendations to airline executives on what movies and pre recorded TV shows should be in their digital catalog for short, or regional, flights, cross-country flights, and international flights.



Business Impact

Selecting the right mix of titles on flights can save the airline companies a large amount of money in licensing fees. The largest international airlines sometimes pay more than $90,000 USD for the license to show a film over a period of two or three months. In the United States, airliners pay a flat fee every time a movie is watched by a passenger. Some airlines spend up to $20-million USD per year on content. Team-13 will analyze how movies and TV shows are selected and, where possible, offer the best choices for entertainment selections that can reduce airline cost, increase customer engagement, and provide a better travel experience via entertainment.

Project Overview

Scope

DS4A Data Engineering Team 13 designed and built an interactive portal where the end-user can search for movie titles by name and returns the title card, the IMBD rating. Automation and ongoing maintenance was out of scope for this project. From conception to delivery, this project lasted approximately three months.

Goals

- Identify connection between IMDB scores and movie popularity
- Analyze publicly available movie and flight data using Python, Redshift and AWS
- Our ultimate goal is to decrease airline operational spend and increase passenger loyalty

Impact

- Airline managers can select a tranche of movies that will fit the duration of the flight and minimize licensing costs.
- Airline carriers can save money by licensing films that fit flight duration and consumer profile information while increasing satisfaction.

Technical Framework

- Data Scoping
- Data Sourcing and Data Wrangling
- Exploratory Data Analysis
- Front-end Design and Data Visualization

Tools Used

- Python
- AWS S3
- AWS Redshift
- AWS EC2
- Selenium
- Plotly Dash
- Excel
- Slack

Data Analysis

Datasets (Movies)

We used two sets of data in our project. The first set would be our movie data. More specifically, we used the Internet Movie Database's Top 1000 Films list. Originally, we were supposed to use one of IMDB's many APIs to extract this data, but IMDB did not have an API specifically for their Top 1000 Films list. So we decided to use web scrapping methods to extract the movie data from their website.

We used Selenium, which is a tool used to automate web browsers to complete tasks, and its Chrome Driver feature to automatically get data of the Top 1000 Films from IMDB's actual website. The data that we extracted included the film's name, summary, directors, stars, genres, posters, and IMDB ratings.

Column Name	Type
index	int
id	varchar
length	int
genre	varchar
rating	float
about	varchar
directors	varchar
stars	varchar
poster	varchar
genre_1	varchar
genre_2	varchar
genre_3	varchar
clean_about	varchar
stars	varchar
director_1	varchar
director_2	varchar
star_1	varchar
star_2	varchar
star_3	varchar
star_4	varchar

Data Analysis

Datasets (Flights)

The second set of data that we used in our project is our flight data. We retrieved real-time flight information from AviationStack.com's flight API. With that API, we retrieved information on specific flights like the flight's departure and arrival locations, departure and landing times, information about each carrier, and the flights' statuses.

Column Name	Type
index	int
flight_date	date
flight_status	varchar
departure.airport	varchar
departure.timezone	varchar
departure.iata	varchar
departure.icao	varchar
departure.scheduled	time
departure.estimated	time
arrival.airport	varchar
arrival.timezone	varchar
arrival.iata	varchar
arrival.icao	varchar
arrival.scheduled	time
arrival.estimated	time
airline.iata	varchar
flight.number	int
flight_length_min	int

Data Analysis

Data Wrangling

We performed a number of data cleaning and data enhancement procedures on the movie dataset. In the movie dataset, the raw data had a Director column that included not only directors, but stars as well. To keep things organized, we created a Python script that extracts the stars data into its own column. If a film has more than one director, then the script will also extract the names of all directors of that film into the director_1 and director_2 columns.

The same is done for stars, with newly created star_1, star_2, star_3, and star_4 columns. The same logic is used for the genres column. If a film has more than one genre, then the script will extract the genres into columns genre_1, genre_2, and genre_3.

Last but not least, we cleaned the "about" column, which lists information about each film. We removed unnecessary punctuations, converted the strings to lowercase, and we removed integers, mentions, and "likes." You can view a sample of this code below.

We did this because we wanted to ensure that each director, star, and genre were easier to find in our database when conducting analytical queries. We also wanted to ensure that each film's "about" column can be easily parsed in case we needed to find any specific substrings within the "about" strings.

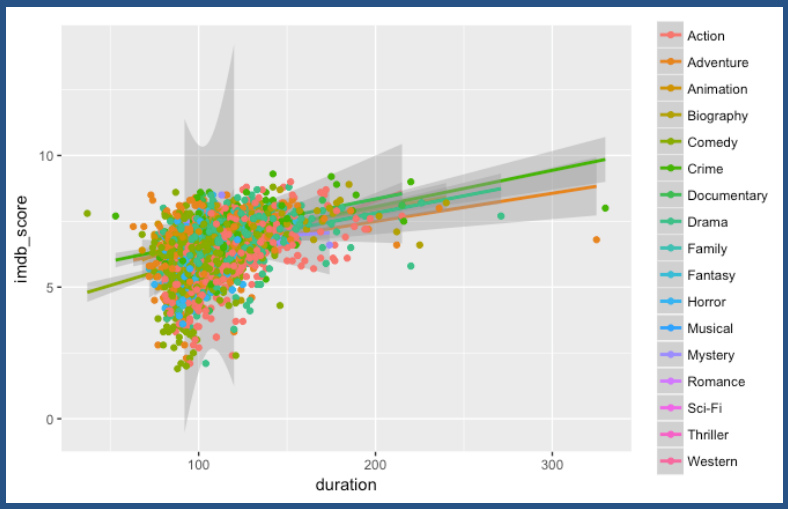
```
70  # Function to clean the About column. This function will remove punctutation, convert to lower-case, remove
    numbers, remove mentions and links, and remove stopwords
71  def clean_about_movies(df):
72      # Cleaning About Movies
73      stop_words = stopwords.words("english") # pull all stopwords from nltk
74      df["clean_about"] = df["about"].apply(lambda x: x.lower()) # Lower-case about movies
75      df["clean_about"] = df["clean_about"].apply(lambda x: re.sub("@[A-Za-z0-9_]+","",x))  # Remove @mentions
        from Tweets
76      df["clean_about"] = df["clean_about"].apply(lambda x: re.sub("#[A-Za-z0-9_]+","",x))  # Remove #hashtags
        from Tweets
77      df["clean_about"] = df["clean_about"].apply(lambda x: x.replace("\n","")) # Remove \n from Tweets
78      df["clean_about"] = df["clean_about"].str.replace(r'[^\w\s]+','') # Remove punctuation from Tweets
79      df["clean_about"] = df["clean_about"].str.replace('\d+','') # Remove numbers from Tweets
80      df["clean_about"] = df["clean_about"].replace(r'http\S+', '', regex=True).replace(r'www\S+', '', regex=True)
        # Remove Links(URL) from Tweets
81      df["clean_about"] = df["clean_about"].str.strip() # Remove Whitespace from Tweets
82      df["clean_about_no_stopwords"] = df["clean_about"].apply(lambda x: ' '.join([word for word in x.split() if
        word not in stop_words]))  # remove stopwords
83      df["name"] = df["name"].str.lower() # lower-case movie_name
84      df["index"] = range(0,0+len(df)) # new incremental index for df
85      df["clean_about"].replace('',np.nan,inplace=True) # replace blanks cells with np.NaN
86      df.dropna(subset=["clean_about"],inplace=True) # Drop cells with NaN tweets
87      return df
88
89  new_df = clean_about_movies(df)
90
```

Data Analysis

Exploratory Data Analysis

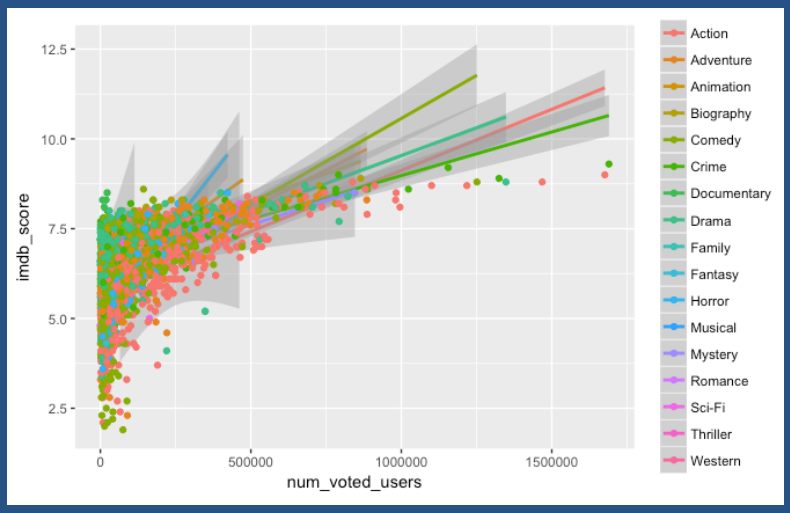
1

In our analysis of the data, we noticed that the longer the movie (duration) the higher the IMDB score. We also see that the majority of IMDB scores fall within the 5.0 to 7.5 range with a duration of 100- to approximately 150-minutes. This cluster is where we will find movie titles with presumably lower licensing costs but with a high degree of viewer satisfaction. Airline managers can search our database for movies of high IMBD scores and then find similar genre to offer to their passengers.



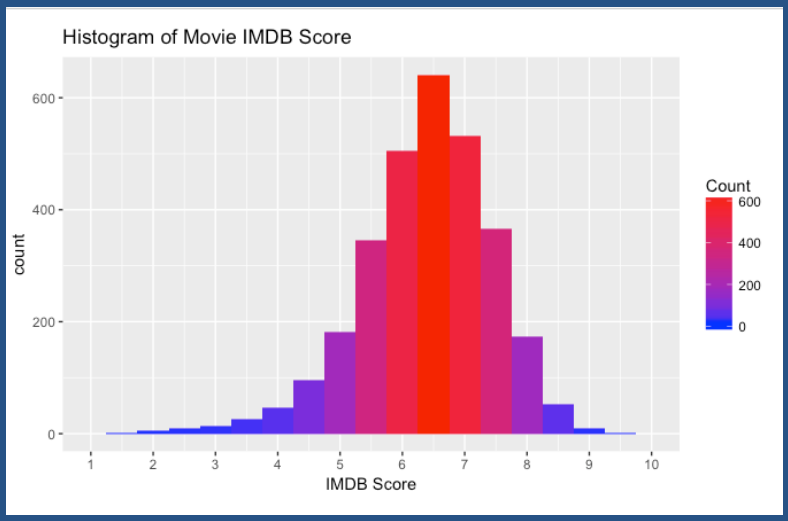
2

In the second chart we can see a similar trend where the scatter plot of markers is “up and to the right.” We concluded that when more viewers vote for a movie, then that movie will have a higher IMDB score. The movie itself may not necessarily be better that other movies within the 5.0 to 7.5 range and so, this finding further strengthened our assumption that a strong catalog of movies for the airline carrier does not have to include highly rated movies. Entertainment for the passenger can be accomplished with lower scoring movies which should lower the Airlines operational (in-flight entertainment) cost.



3

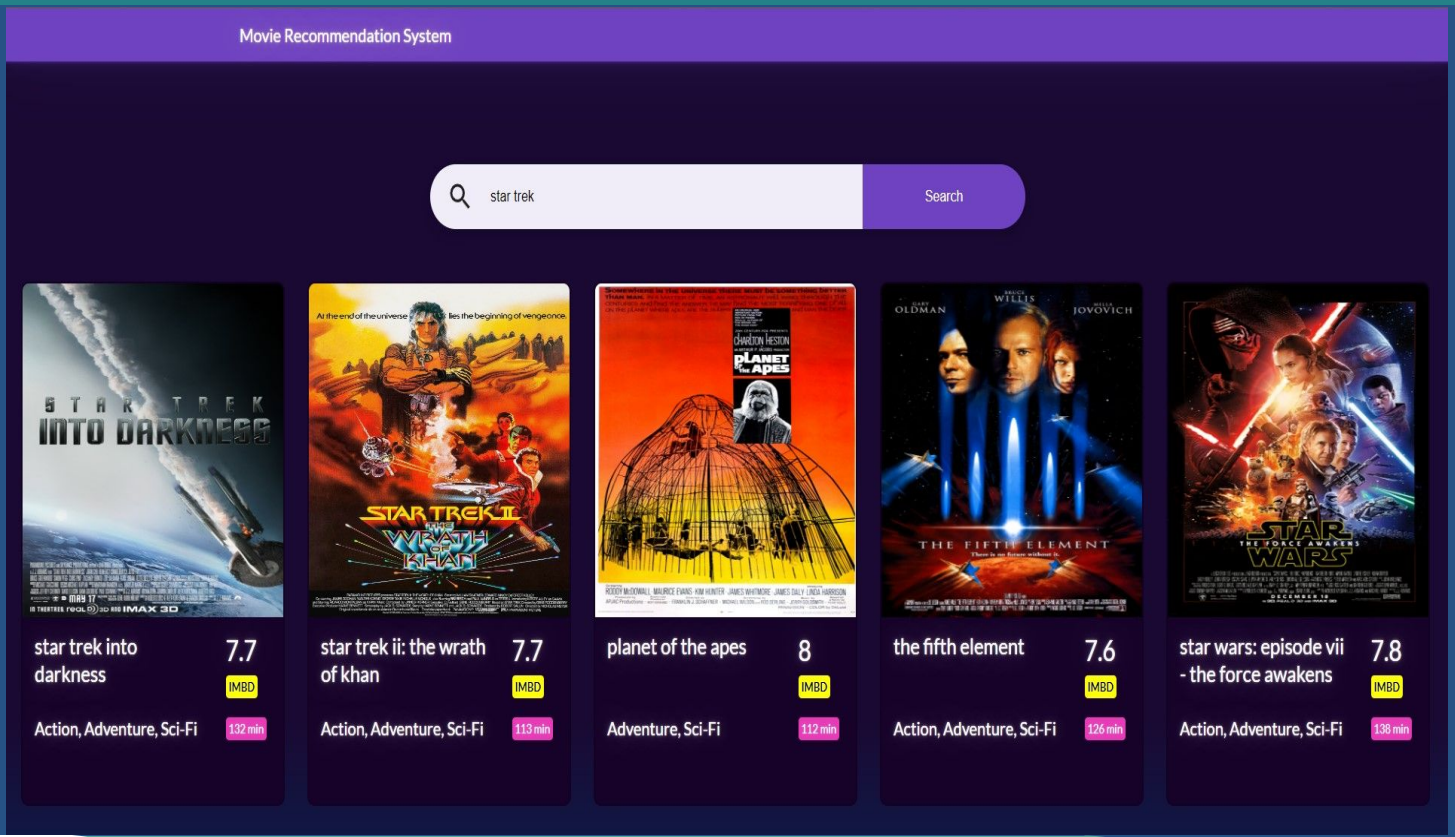
Our histogram gives a different view of the same data, majority of movies have an IMBD score between 5.0 and 7.5 and so, it is here where we recommend Airline managers focus when negotiating licensing contracts with Hollywood studios.



Dashboard

Use Case

We created a business client-facing portal where the end-user can search for movie titles by name. The app will then return five films that are most similar to the film the user typed in. Each individual result includes the movie's title, the IMDB rating, the genres, the runtime, and the movie poster. In a future version of our app, double-clicking on a result will present the end-user with more detailed information about the movie, including approximate licensing fees and recommended flight routes.



Dashboard

Data Engineering

1. Extracting the Data

We created an ETL pipeline for our Movie Recommendation App using three AWS EC2 Instances. In the first EC2 Instance, we extracted our movie data from IMDB's Top 1000 Movies list using Selenium's webscrapping feature. We also extracted our flight data by making API calls to AviationStack.com using Python's Requests, JSON, and Boto3 libraries. This Python script will then create a CSV file from this data, which will be sent to an AWS S3 Bucket being used as our data lake.

2. Transforming the Data

In the second EC2 Instance, we used a Python script to transform our raw data. The Python script first retrieves the raw data's CSV file from our data lake. Then it transforms our data by cleaning it in a number of ways, like cleaning up the movie data's genre, stars, directors, and about columns. More information on this process can be found in the Data Wrangling section on page 8.

3. Loading the Data (and Beyond)

After the data is cleaned, the Python script creates a new CSV file and sends it to a second S3 Bucket. Now the user is able to load the clean data into our database on AWS Redshift, which acts as our Data Warehouse. In our third EC2 Instance, we used a Python script to retrieve the clean data from our second AWS S3, and to prepare the data for our Movie Recommendation App by using Cosine Similarity, which we'll get into on page 11. Once the data is prepared, a second Python script is used to initialize our Plotly Dash app UI.

Dashboard
Data Engineering

What is Cosine Similarity?

Cosine Similarity measures the similarity between two sequences of numbers, or in this case, two vectors. Each vector represents a film and it includes the film's description, directors, and actors. The vectors that are closest to each other represents a strong similarity between the two films. So, in the image shown on the right, we have Vector A and Vector B, which represent two films. The angular distance between A and B is represented as theta. When the angular measurement of theta is small, it signals a strong similarity between Vector A and Vector B.

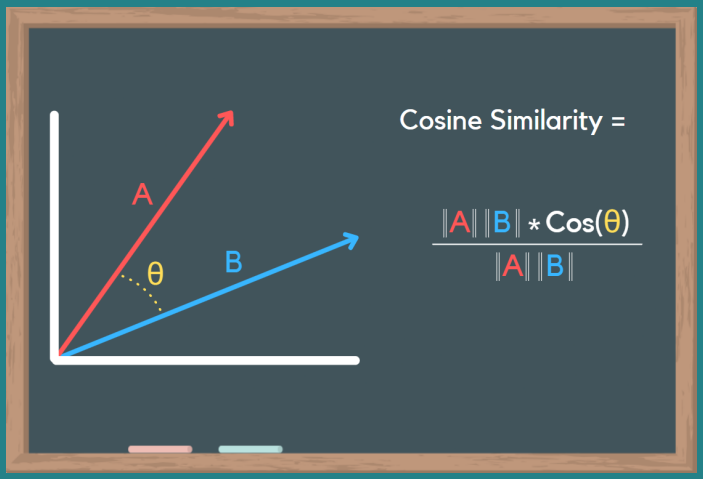
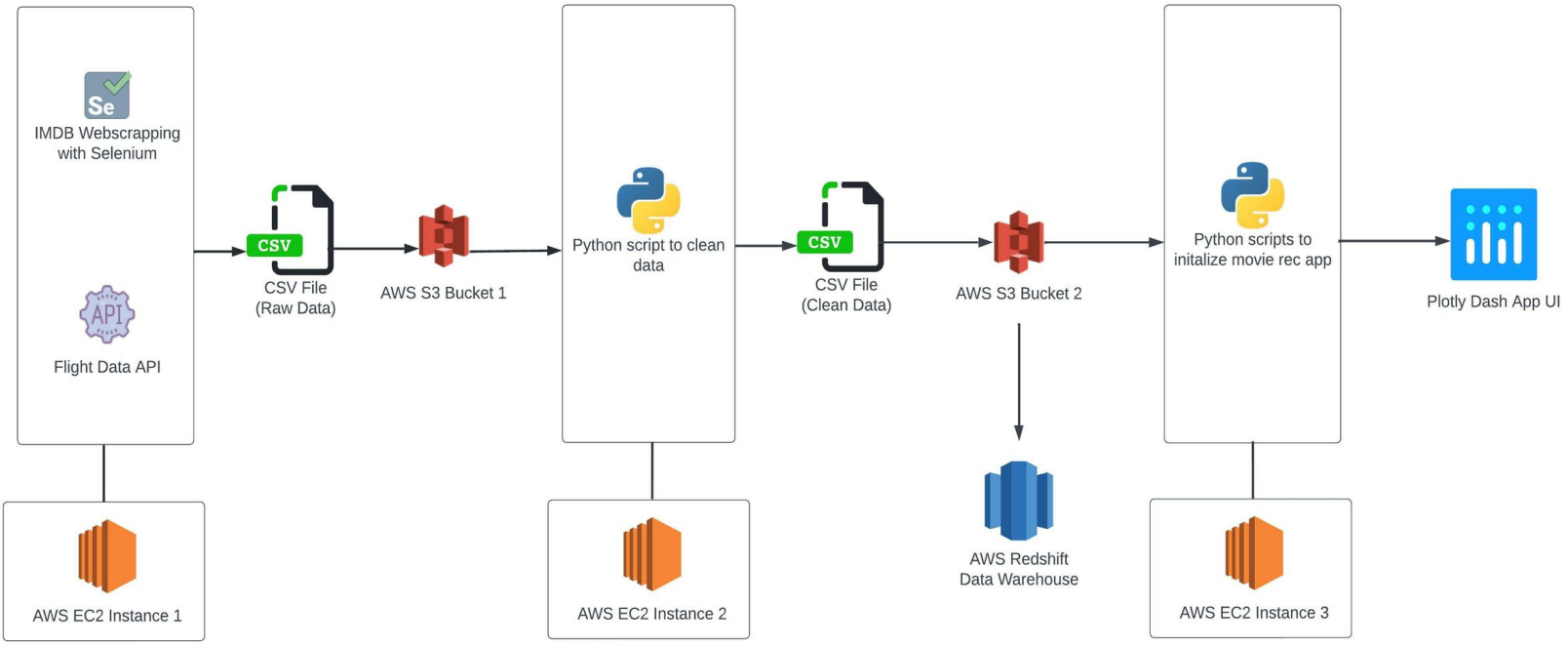


Image by Ben Chamblee, Towards Data Science

ETL Pipeline Diagram



Conclusions

What's Next?

Most of us have experienced airline travel where the In-Flight Experience (IFE) has played a critical role on our desire to travel again with the same airline brand. Can you imagine sitting on an International flight for 10+ hours looking at the seat in front of you with no entertainment?

Movie recommendation systems are an invaluable additions to the travel experience. It provides comfort and personalization. It also provides a customer retainage when the word spreads that “XYZ Airlines has the best movies!” Let’s not forget about keeping the kids happy, too.

In this digital age and increased travel, along with a growing desire for experiences, being able to recommend movies for domestic and international flights will increase user engagement. Most importantly, a personalized travel experience, an increase in in-flight satisfaction, and a diverse array of movie choices will entice the consumer toreturn and look forward to flying.

Continuous Entertainment Experience (CEE): Beyond the Arrival Point

- Passengers can complete a movie via their Airline ap
- Creation of user profiles where specific movies (within the catalog of movies licensed by the airline) are aligned with the passenger's preferences
- When flying one specific carrier (e.g. Delta), the passenger can pick up the movie they were watching on a connecting flight

References

The Internet Movie Database (IMDB)
https://www.imdb.com/

Aviation Stack
https://www.aviationstack.com/

Nerd Wallet, In-Flight Entertainment: Broken Down by Airline
https://www.nerdwallet.com/article/travel/in-flight-entertainment-the-complete-guide#:~:text=Movies%20an

Towards Data Science, What is Cosine Similarity? How to Compare Text and Images in Python
https://towardsdatascience.com/what-is-cosine-similarity-how-to-compare-text-and-images-in-python-d2bb6e411ef0

Acknowledgements

We would like to thank everyone who helped in making our project a possibility. They include, but are not limited to:

Correlation One

The DS4A Data Engineering Program Staff

Our Teaching Assistants Jefferson Bien-Aime and Jonathan Schlosser

Our Mentors Daniel Silva and Bandi Kirankumar

The members of Team 13