December 29, 2024

# C964: Computer Science Capstone

Movie Recommendation Service

## Task 2 parts A, B, C and D

Part A: Letter of Transmittal........2

Part B: Project Proposal Plan........5

Project Summary........5

Data Summary........6

Implementation........6

Timeline........7

Evaluation Plan........8

Resources and Costs........8

Part C: Application........10

Part D: Post-implementation Report........11

Solution Summary........11

Data Summary........11

Machine Learning........13

Validation........15

Visualizations........16

User Guide........18

## Part A: Letter of Transmittal

December 29, 2014

To Joey Wheeler, Lunabox CEO

834 Philmont Street

Brooklyn, NY 11214

Dear Mr. Wheeler,

I am writing to you today to present a new movie recommendation service for your company Lunabox with the goal of enhancing your user base's experience. We aim to increase user engagement by utilizing advanced algorithms that will be capable of providing personalized movie recommendations for each user that are tailored to their personal preferences. In doing so, we will create a more engaging and captivating viewing experience that will keep users satisfied, thus improving user retention.

In the current state of your product, Lunabox's streaming service lacks the ability to make suggestions for what to watch next.  This inability to personalize the experience to each user may be discouraging to users who find difficulty in making use of the streaming service. As a result, users may be more inclined to cancel their memberships if they do not receive much value out of it.

The proposed solution will be a movie recommendation service that gathers user data and feedback on movies. This data will be used to tailor its recommendations to users. When a user searches up a particular movie that they enjoy, a list of movies that have rated positively by users with similar preferences will be returned to the user. This will be done through the use of machine learning algorithms that will continuously learn from the interactions of Lunabox's user base over time. This solution aims to increase the usage of the streaming service, thus improving user satisfaction.

Should this solution be implemented, the recommendation service will provide many benefits to the organization. Some of the key benefits are listed below:

1. By offering a tailored user experience, users will log more activity on the streaming service, thus enhancing user engagement.
2. As users will be utilizing Lunabox more, they are more likely to see value in their subscription making them more likely to renew their membership. This in turn increases user retention.
3. Lunabox's direct competitors currently lack the ability to recommend movies to its user base. Implementing this gives them a direct edge over its competition, thus making Lunabox more appealing to prospective users.

Developing this recommendation system will require a software engineer, a machine learning specialist, and a quality assurance engineer on staff. Below is a brief summary of costs, timelines, and ethical concerns regarding the development process.

Costs: The cost of implementing the solution will be approximately $21,675 for a staff of three people comprising of a software engineer, machine learning specialist, and a quality assurance engineer each earning $85/hour.

Timelines: The estimated time to complete this project will be 85 hours of work with a product delivery date of 01/31/2025.

Ethical Concerns: Because this recommendation service tracks user data, it is important that we comply with data compliance laws and protect the handling of user data.

Me and my team have a long history of experience in software development as well as machine learning and are confident and highly motivated in delivering a product that will meet the highest of standards for both Lunabox’s team as well as its users. Having led many projects similar to this, I believe that I will efficiently and reliably meet all timelines as well as keep development costs within the allotted budget.

I look forward to talking to you further regarding this proposal! If you need to contact me, my phone number is listed below.

Sincerely,

Luigi Celestino

Project Manager

(123) - 456 - 7890

## Part B: Project Proposal Plan

### Project Summary

Lunabox as a streaming service currently has no way of making recommendations to its users for what movie to watch. The only way users can find movies is by already knowing them beforehand and searching it up on the platform. This creates a lackluster experience for users that also fails to differentiate Lunabox from its competitors.

Lunabox, as an organization, is less likely to grow its user base as a result of this. This proposal will outline a solution to resolving this issue, ultimately delivering a recommendation service for users that created tailored experiences that increase user engagement.

The following list will outline the project deliverables:

1. Project plan will be written that outlines each step of the development process.
2. Dataset that will be used.
3. Implementation of the recommendation system using Jupyter Notebook.
4. Interactive user interface.
5. Project documentation that includes a user guide.

This recommendation system will benefit Lunabox in many ways. Initially, the end users will be provided with a superior experience that is tailored to their preferences. As a result of this, user engagement will improve which will improve user retention rates as well. Because of this, users will be more likely to renew their subscriptions which results in higher revenue. Another benefit to Lunabox is that the recommendation system will provide them an edge against their competition, allowing them to stand out.

## Data Summary

The dataset that will be used for this project will be sourced from grouplens.org at the following link: https://files.grouplens.org/datasets/movielens/ml-latest-small.zip. This dataset has already been cleaned prior to the developmental process. It will be further cleaned during development as titles will be reformatted to exclude non-alphanumeric characters. The dataset is quite expansive and includes approximately 10,000 films with user-ratings and genres for each film as well as its own unique ID. As such, the movie is well-suited for the needs of the project as it can easily accommodate the movies offered by the Lunabox streaming service. There will be no ethical or legal concerns in the use of this dataset as it is free to use by the public and contains no sensitive information.

## Implementation

In the development of this product, we will use the waterfall methodology. Because this project has clearly defined objectives for each step during development and the team already has experience with similar projects, the waterfall methodology with its structured approach is a good methodology to employ.

The project will begin with setting up the development of the project. This will involve the planning of the project as well as setting up the environment that we will develop the product in such as installing the necessary libraries. The dataset will then be acquired and explored to identify patterns and to determine if any modifications to the dataset is required. After the data has been thoroughly explored, we can begin to develop and train the model that will be used for the movie recommendation system. Once the model has been developed, a user interface will be implemented that allows end users to input a movie which will then return a list of movie recommendations based on the user input. Finally, testing of the product can take place and the completed product will be made available for submission.

## Timeline

| Milestone or deliverable | Duration (hours or days) | Projected start date | Anticipated end date |
|---|---|---|---|
| Project plan will be written up | 2 days | 12/30/2024 | 01/01/2025 |
| Development environment will be set up | 1 day | 01/01/2025 | 01/02/2025 |
| Acquire dataset | 1 day | 01/02/2025 | 01/03/2025 |
| Explore the dataset (and modify if necessary) | 7 days | 01/03/2025 | 01/10/2025 |
| Development and training of of ML model | 10 days | 01/10/2025 | 01/20/2025 |
| Implementation of interactive user interface | 3 days | 01/20/2025 | 01/23/2025 |
| QA Testing | 4 days | 01/23/2025 | 01/27/2025 |
| Final Fixes | 3 days | 01/27/2025 | 01/30/2025 |
| Product Submission | 1 day | 01/30/2025 | 01/31/2025 |

## Evaluation Plan

During the development of this product, proper version control practices will be used to ensure that no faulty code makes its way to the live product. This will be done via GitHub to make sure that each chunk of code written by the team can be tested in its own independent environment that each team member will be working in.

Validation for the performance of the product will be evaluated in the future by users of Lunabox's streaming service once the product goes live. Because the product is supposed to recommend movies that the user will like, it is only possible to receive quantifiable data regarding the success of this directly from Lunabox's user base. This will be done by prompting users to rate the quality of the recommendations given to them. This user feedback will be logged and stored by the service. If less than 50% of users are satisfied with their automated recommendations, the service will be deemed unsuccessful and adjustments to the model will be required.

## Resources and Costs

The product will be developed entirely on Jupyter Notebook using Python 3. Because Jupyter Notebook is free, the only hardware/software costs come from the work stations for the three team members. Each work station will cost $1,500 each, or a total of $4,500.  Each team member on staff earns an hourly rate of $85/hour. With an expected project duration of a total of 255 work hours, the cost of labor will cost approximately $21,675. As a result of being hosted entirely on the Jupyter Notebook as a standalone product, it will cost nothing to be deployed. Overall, the total cost of the entire project amounts to $26,175.

|  |  |  |
|---|---|---|

| Resource | Description | Cost |
|---|---|---|
| Software Engineer | Will be responsible for the front and backend | $7,225 |
| Machine Learning Specialist | Handles the development of the ML model | $7,225 |
| QA Engineer | Ensures the product passes all required testing | $7,225 |
| Work stations | Three work stations, each costing $1,500 | $4,500 |
| Total | Cost of the entire project | $26,175 |

## Part C: Application

The application will be included as part of the submission in the folder.

*Because the application is run locally from a Jupyter Notebook, there are no security concerns and as such, no industry appropriate security features were implemented.

## Part D: Post-implementation Report

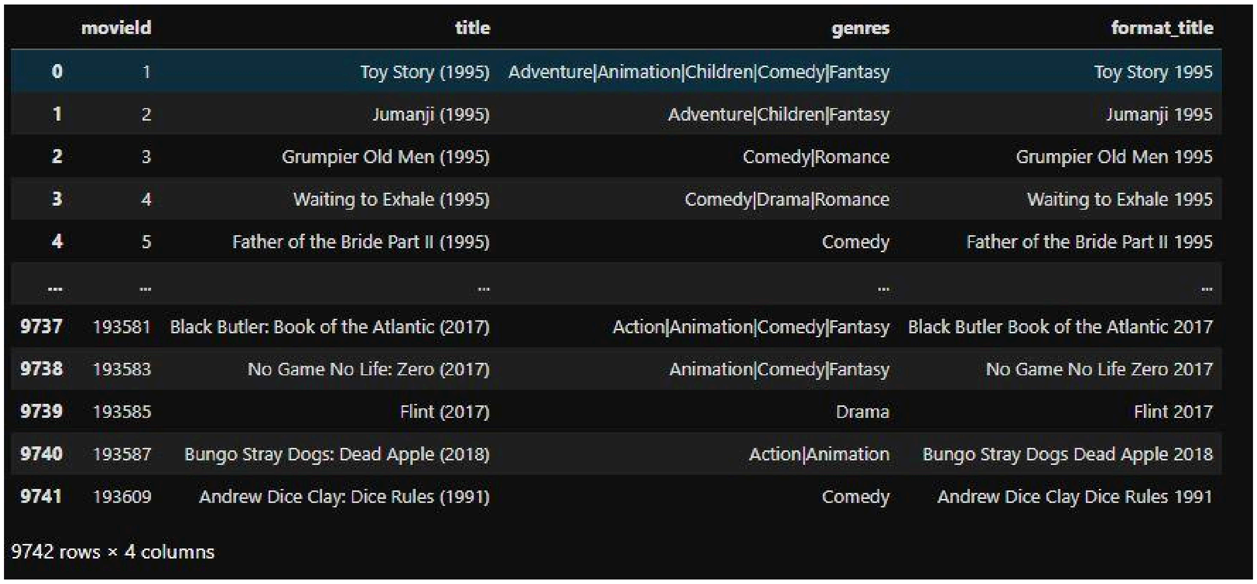
Create a post-implementation as outlined below. Provide sufficient detail so that a reader knowledgeable in computer science but unfamiliar with your project can understand what you have accomplished. Using examples and visualizations (including screenshots) beyond the three required is recommended (but not required). **Write everything in the past tense.**

### Solution Summary

The problem that Lunabox faced was the lack of a movie recommendation system for its users. As a result, users were faced with a lackluster experience. Without this feature, users were required to have their own knowledge of which movies to watch to discover new content. As a result, user engagement was not as high compared to their competitors. As a result, Lunabox contacted us to fix this issue. The solution that was created was a standalone tool that used machine learning algorithms to recommend to its users different movies to watch based on their preferences. It provided users with an interactive user interface that allowed for users to input a movie that they liked and through the use of user-based collaborative filtering, would output a list of movies that it recommended. The application provided a solution to Lunabox's issue by providing a feature that recommended movies to Lunabox's users.

### Data Summary

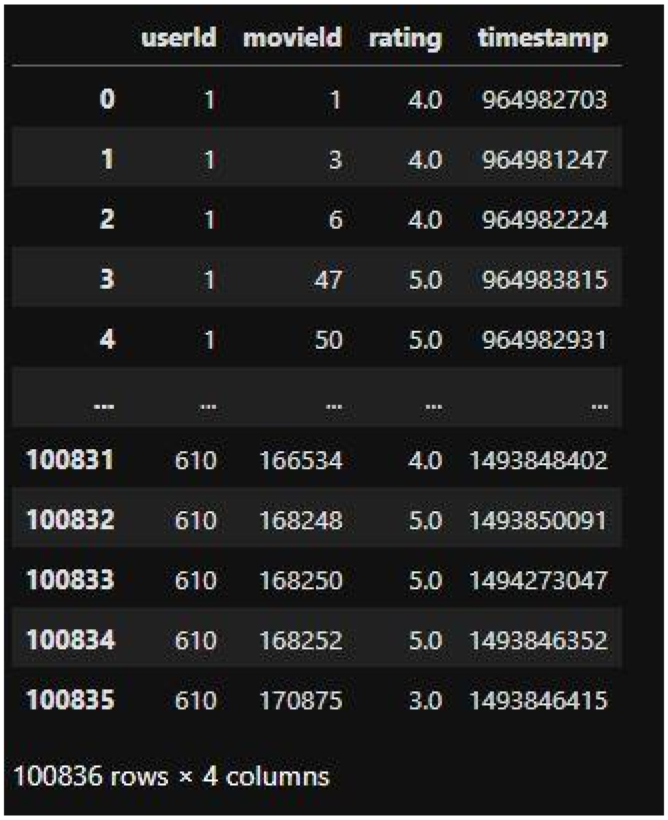
The dataset used for this project were CSV files describing movies and their features such as genres, as well as their user-reported ratings. It was sourced from grouplens' website, a research lab specializing in recommender systems. The data had already been cleaned prior to development as it was handled by grouplens' researchers. Nonetheless, exploration of the data still took place to determine if there were any anomalies or missing values. It was determined that the data did not have any null values, nor were there any anomalies that would require further investigation. However, further data cleaning did take place during development as the titles of the movies needed to be reformatted to provide more reliable search queries to the user. Figure 1 shows a section of the movies dataset that was used in the project while figure 2 shows the ratings dataset that corresponds to the movie dataset on their movieId.



| | movieId | title | genres | format_title |
|---|---|---|---|---|
| **0** | 1 | Toy Story (1995) | Adventure\|Animation\|Children\|Comedy\|Fantasy | Toy Story 1995 |
| **1** | 2 | Jumanji (1995) | Adventure\|Children\|Fantasy | Jumanji 1995 |
| **2** | 3 | Grumpier Old Men (1995) | Comedy\|Romance | Grumpier Old Men 1995 |
| **3** | 4 | Waiting to Exhale (1995) | Comedy\|Drama\|Romance | Waiting to Exhale 1995 |
| **4** | 5 | Father of the Bride Part II (1995) | Comedy | Father of the Bride Part II 1995 |
| **...** | ... | ... | ... | ... |
| **9737** | 193581 | Black Butler: Book of the Atlantic (2017) | Action\|Animation\|Comedy\|Fantasy | Black Butler Book of the Atlantic 2017 |
| **9738** | 193583 | No Game No Life: Zero (2017) | Animation\|Comedy\|Fantasy | No Game No Life Zero 2017 |
| **9739** | 193585 | Flint (2017) | Drama | Flint 2017 |
| **9740** | 193587 | Bungo Stray Dogs: Dead Apple (2018) | Action\|Animation | Bungo Stray Dogs Dead Apple 2018 |
| **9741** | 193609 | Andrew Dice Clay: Dice Rules (1991) | Comedy | Andrew Dice Clay Dice Rules 1991 |

9742 rows × 4 columns

Figure 1: The movies dataset



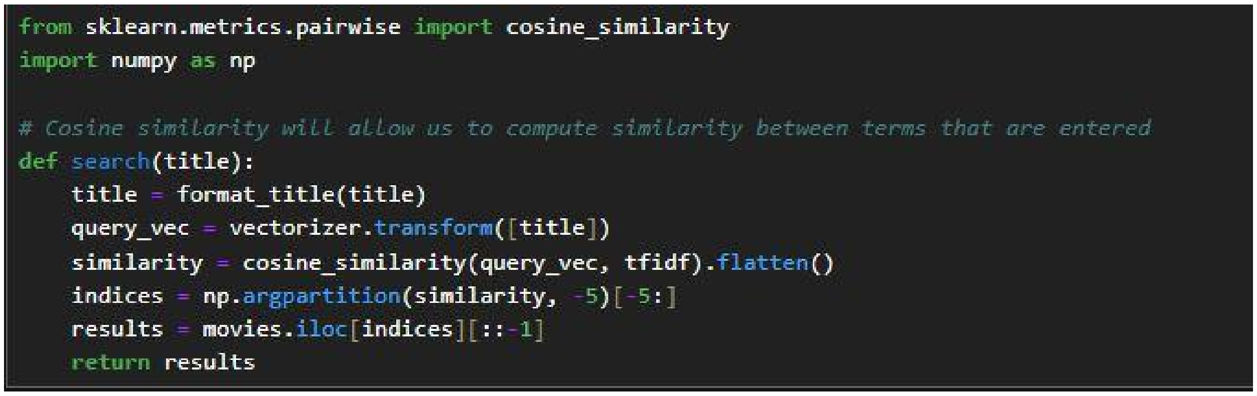
| | userId | movieId | rating | timestamp |
|---|---|---|---|---|
| **0** | 1 | 1 | 4.0 | 964982703 |
| **1** | 1 | 3 | 4.0 | 964981247 |
| **2** | 1 | 6 | 4.0 | 964982224 |
| **3** | 1 | 47 | 5.0 | 964983815 |
| **4** | 1 | 50 | 5.0 | 964982931 |
| **...** | ... | ... | ... | ... |
| **100831** | 610 | 166534 | 4.0 | 1493848402 |
| **100832** | 610 | 168248 | 5.0 | 1493850091 |
| **100833** | 610 | 168250 | 5.0 | 1494273047 |
| **100834** | 610 | 168252 | 5.0 | 1493846352 |
| **100835** | 610 | 170875 | 3.0 | 1493846415 |

100836 rows × 4 columns

Figure 2: The ratings dataset

## Machine Learning

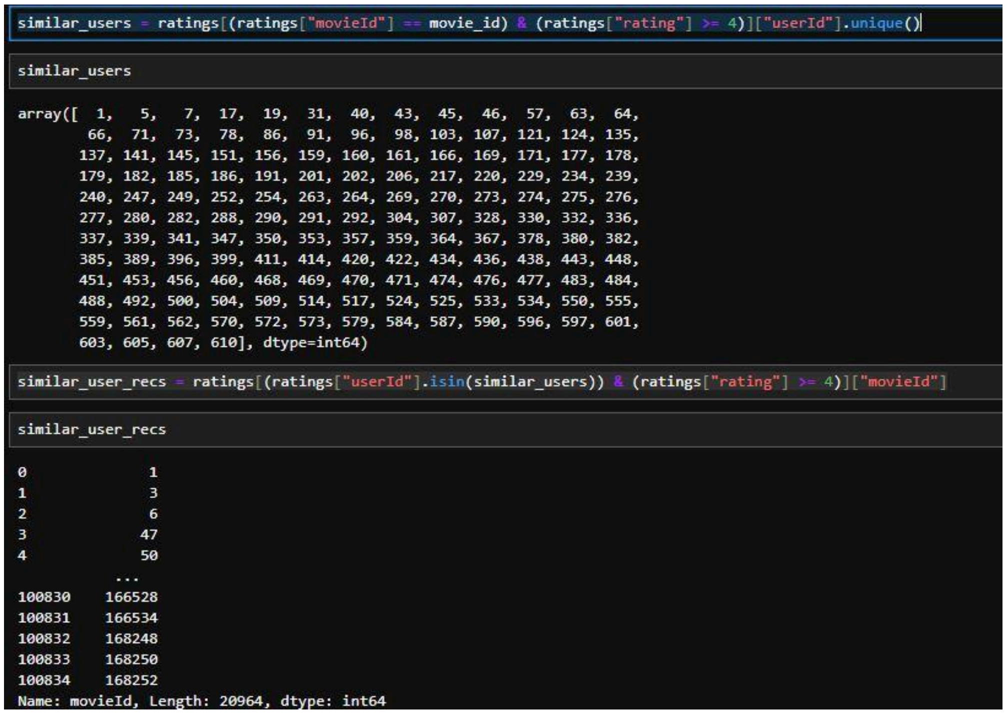
The recommendation system that was developed used user-based collaborative filtering and utilized cosine similarity scores to compute the similarity between the term that the user entered and the movies in the database. In order for this to happen, the movie titles were first vectorized so that cosine similarity could be used. Cosine similarity is a metric used to determine how similar two vectors are, or in this case two movies. This calculation was done through the usage of the sklearn library and returns a search result of the most similar movies.



```python
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np

# Cosine similarity will allow us to compute similarity between terms that are entered
def search(title):
    title = format_title(title)
    query_vec = vectorizer.transform([title])
    similarity = cosine_similarity(query_vec, tfidf).flatten()
    indices = np.argpartition(similarity, -5)[-5:]
    results = movies.iloc[indices][::-1]
    return results
```

Figure 3: Search function that uses cosine similarity to return similar movies

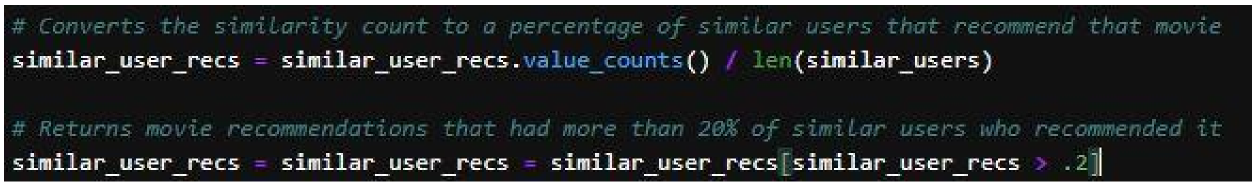
Upon returning a list of similar movies, the program determined a list of users from the database similar to the user of the tool. This was calculated by matching the movie ID that the user input to other unique users who rated that movie at least four stars.



```
similar_users = ratings[(ratings["movieId"] == movie_id) & (ratings["rating"] >= 4)]["userId"].unique()

similar_users

array([  1,   5,   7,  17,  19,  31,  40,  43,  45,  46,  57,  63,  64,
        66,  71,  73,  78,  86,  91,  96,  98, 103, 107, 121, 124, 135,
       137, 141, 145, 151, 156, 159, 160, 161, 166, 169, 171, 177, 178,
       179, 182, 185, 186, 191, 201, 202, 206, 217, 220, 229, 234, 239,
       240, 247, 249, 252, 254, 263, 264, 269, 270, 273, 274, 275, 276,
       277, 280, 282, 288, 290, 291, 292, 304, 307, 328, 330, 332, 336,
       337, 339, 341, 347, 350, 353, 357, 359, 364, 367, 378, 380, 382,
       385, 389, 396, 399, 411, 414, 420, 422, 434, 436, 438, 443, 448,
       451, 453, 456, 460, 468, 469, 470, 471, 474, 476, 477, 483, 484,
       488, 492, 500, 504, 509, 514, 517, 524, 525, 533, 534, 550, 555,
       559, 561, 562, 570, 572, 573, 579, 584, 587, 590, 596, 597, 601,
       603, 605, 607, 610], dtype=int64)

similar_user_recs = ratings[(ratings["userId"].isin(similar_users)) & (ratings["rating"] >= 4)]["movieId"]

similar_user_recs

0              1
1              3
2              6
3             47
4             50
           ...
100830    166528
100831    166534
100832    168248
100833    168250
100834    168252
Name: movieId, Length: 20964, dtype: int64
```

Figure 4: Returning the list of similar users based on the movie input

The list was then further refined to only include movie recommendations that had at least 20% of users who were deemed similar to the user of the tool. Furthermore, the similarity score was converted to a percentage to easily represent how strong of a recommendation the movie was.



```python
# Converts the similarity count to a percentage of similar users that recommend that movie
similar_user_recs = similar_user_recs.value_counts() / len(similar_users)

# Returns movie recommendations that had more than 20% of similar users who recommended it
similar_user_recs = similar_user_recs = similar_user_recs[similar_user_recs > .2]
```

Figure 5: Similarity score was converted to a percentage and the list of movies was further refined to only include movies with at least 20% of four star reviews from similar users

Following this, the program would then filter the results even further until the recommendation tool ultimately outputs a list of movies that it recommends the user watch based on their input.

Cosine similarity was a good fit for the implementation of the system because it would recommend movies that had similar names to the movie that the user was interested in. This is an obvious choice because naturally, if a person enjoyed a movie from a franchise, a sequel which bears the same franchise name should be recommended to the user. Collaborative filtering was also a good fit for this tool because we wanted the user's input to be tailored to their preferences so identifying patterns in movie preferences from similar users helped to create the recommendation tool.

## Validation

Because the performance metric of the recommendation system is gauged by user satisfaction, the means of validating the success of the tool is to directly request for feedback from the user. This will be done post-implementation when the product is already live for Lunabox's users to utilize. Following the usage of the tool, the user will be prompted for feedback regarding the quality of the recommendations and their feedback will be stored for analysis. If the satisfaction rate reported by the users who used the tool are less than the targeted 50%, the tool will be considered a failure and the system will need to be readjusted.

## Visualizations

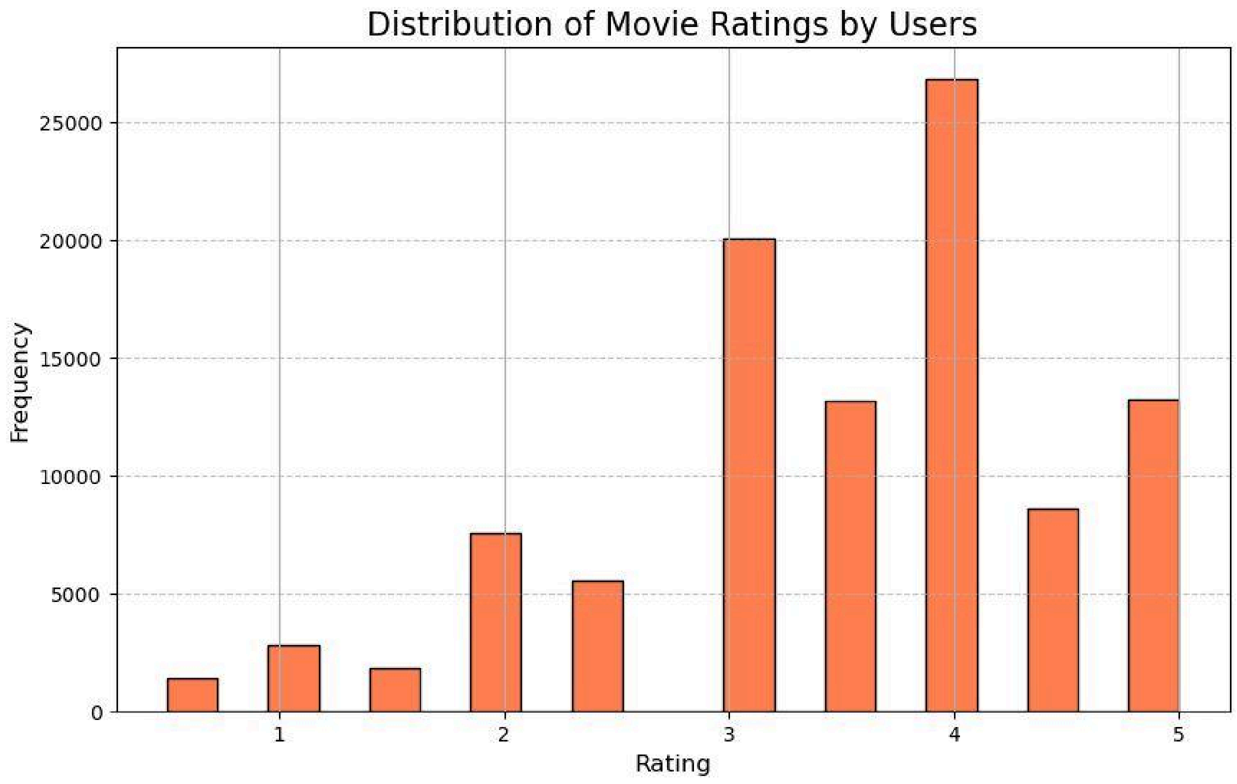


Figure 6: Bar chart of distribution of movie ratings reported by users

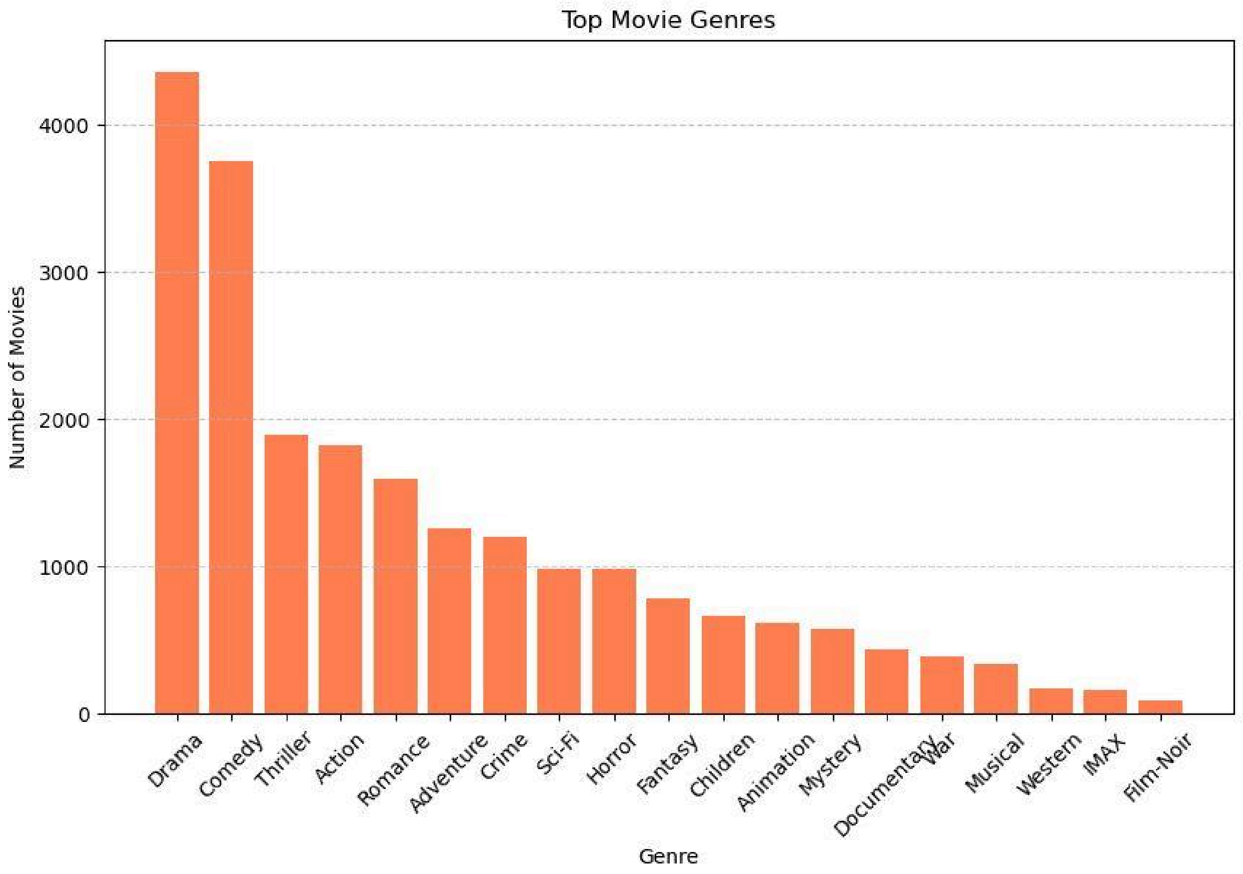


Figure 7: Bar chart of the top genres from the dataset

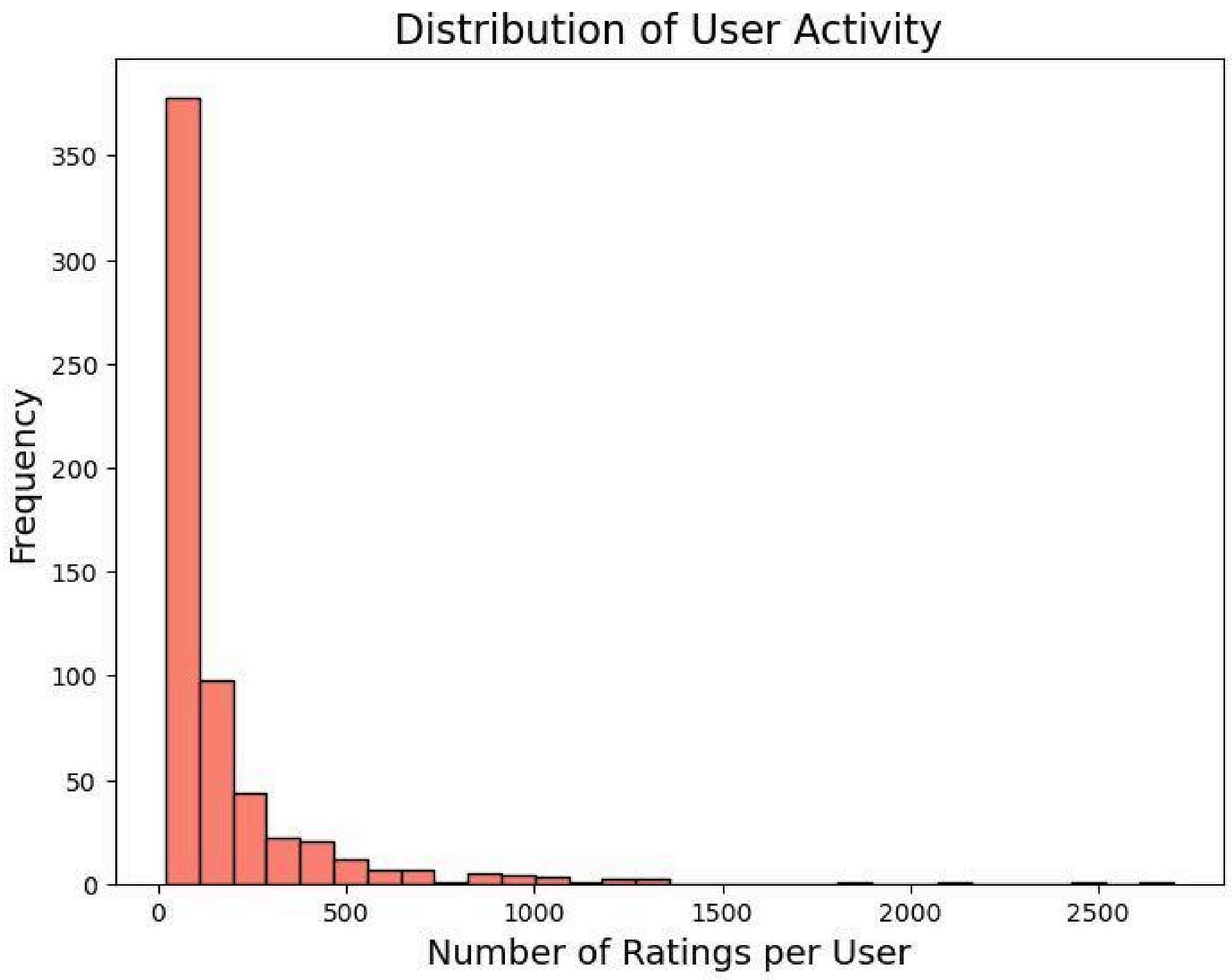


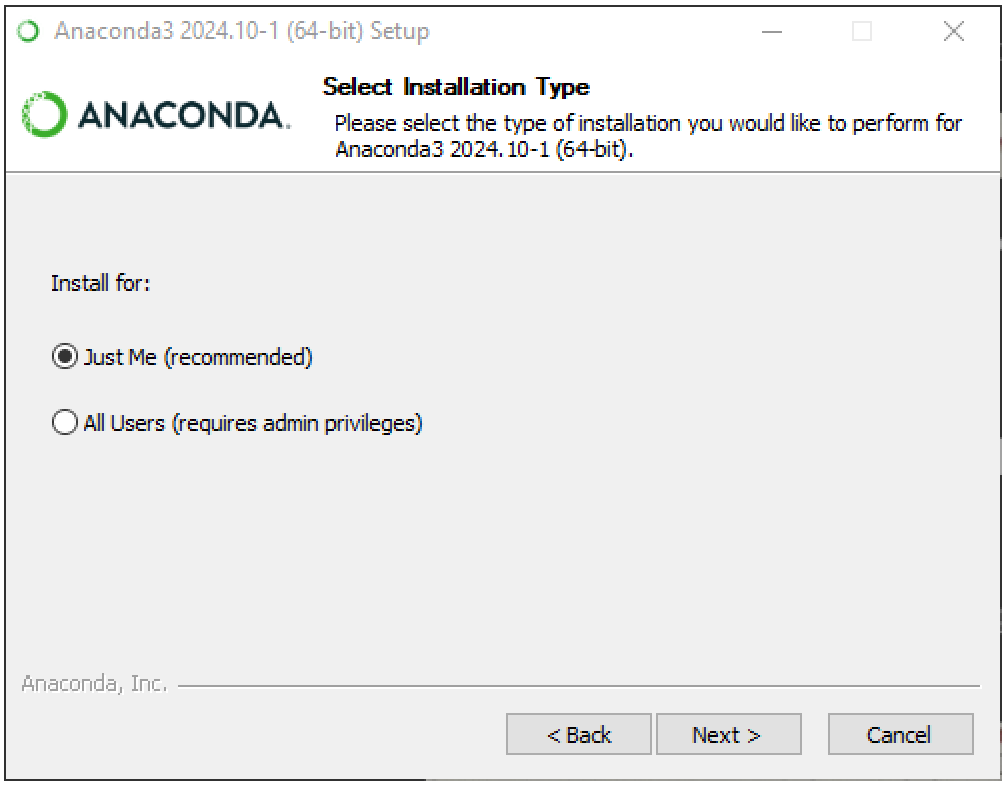
Figure 8: Histogram showcasing the distribution of user activity based on their number of ratings

## User Guide

1. Download the folder named C964 Capstone Submission.
    a. Inside are the datasets for the movies and the ratings as well as the Jupyter Notebook file for the standalone tool.

2. Navigate to [https://www.anaconda.com/download/success](https://www.anaconda.com/download/success) to download Anaconda if you don't have it yet.

3. Run the installer and when prompted to select your installation type, select "Just Me (recommended)"



Anaconda3 2024.10-1 (64-bit) Setup

**Select Installation Type**

Please select the type of installation you would like to perform for Anaconda3 2024.10-1 (64-bit).

Install for:

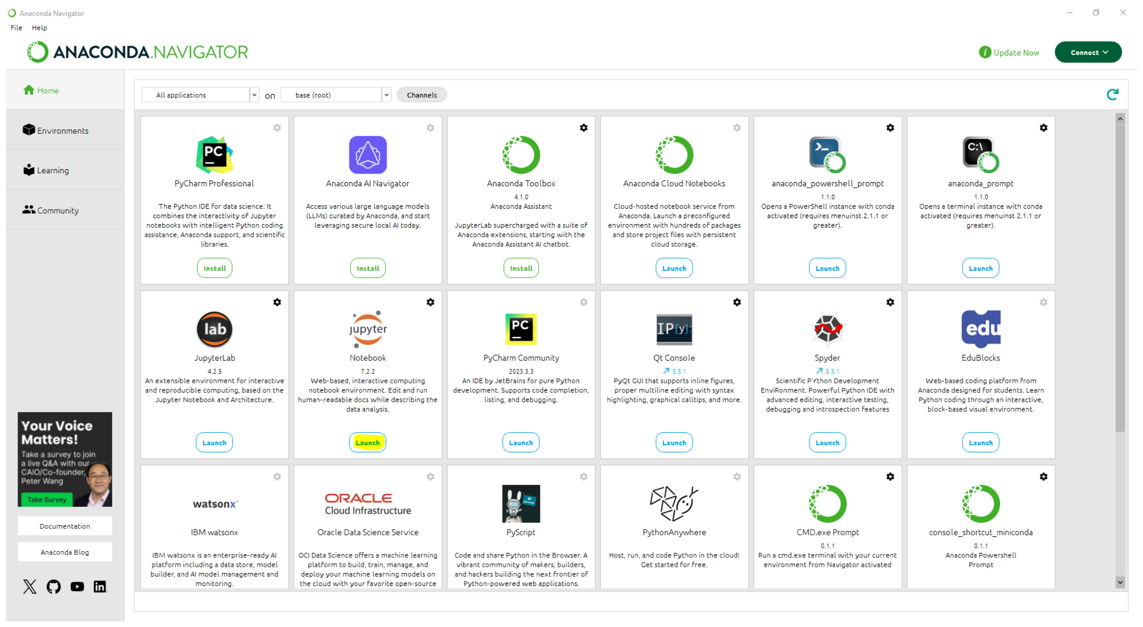
[✓ Just Me (recommended)]

All Users (requires admin privileges)

Anaconda, Inc.

4. Click through the prompts and let the installer run.

5. Once Anaconda Navigator is installed, make sure that Notebook is installed from the home screen. Once installed, click launch.



PyCharm Professional
The Python IDE for data science. It combines the interactivity of Jupyter notebooks with intelligent Python coding assistance, Anaconda support, and scientific libraries.
Install

Anaconda AI Navigator
Access various large language models (LLMs) curated by Anaconda, and start leveraging secure local AI today.
Install

Anaconda Toolbox
4.1.0
Anaconda Assistant
JupyterLab supercharged with a suite of Anaconda extensions, starting with the Anaconda Assistant AI chatbot.
Install

Anaconda Cloud Notebooks
Cloud-hosted notebook service from Anaconda. Launch a preconfigured environment with hundreds of packages and store project files with persistent cloud storage.
Launch

anaconda_powershell_prompt
1.1.0
Opens a PowerShell instance with conda activated (requires menuinst 2.1.1 or greater).
Launch

anaconda_prompt
1.1.0
Opens a terminal instance with conda activated (requires menuinst 2.1.1 or greater).
Launch

JupyterLab
4.2.5
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch

Notebook
7.2.2
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch

PyCharm Community
2023.3.3
An IDE by JetBrains for pure Python development. Supports code completion, listing, and debugging.
Launch

Qt Console
5.5.1
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.
Launch

Spyder
5.5.1
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
Launch

EduBlocks
Web-based coding platform from Anaconda designed for students. Learn Python coding through an interactive, block-based visual environment.
Launch

IBM watsonx
IBM watsonx is an enterprise-ready AI platform including a data store, model builder, and AI model management and monitoring.

Oracle Data Science Service
OCI Data Science offers a machine learning platform to build, train, manage, and deploy your machine learning models on the cloud with your favorite open-source

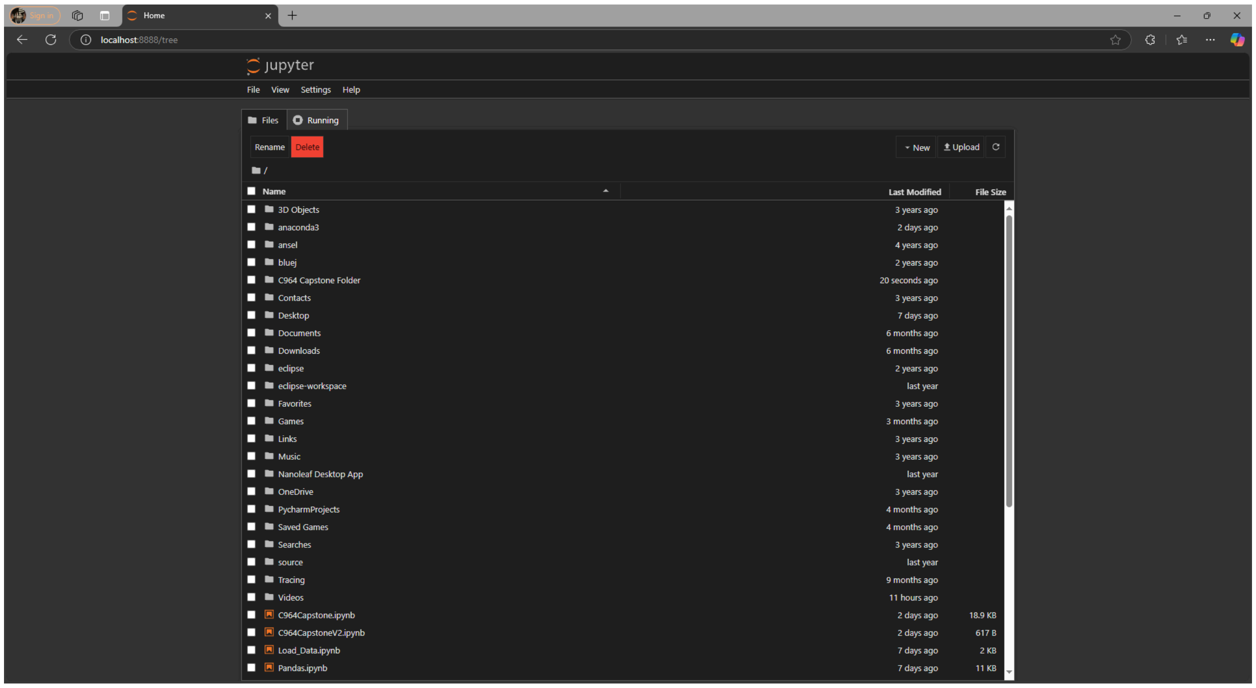
PyScript
Code and share Python in the Browser. A vibrant community of makers, builders, and hackers building the next frontier of Python-powered web applications.

PythonAnywhere
Host, run, and code Python in the cloud! Get started for free.

CMD.exe Prompt
0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated

console_shortcut_miniconda
0.1.1
Anaconda Powershell Prompt

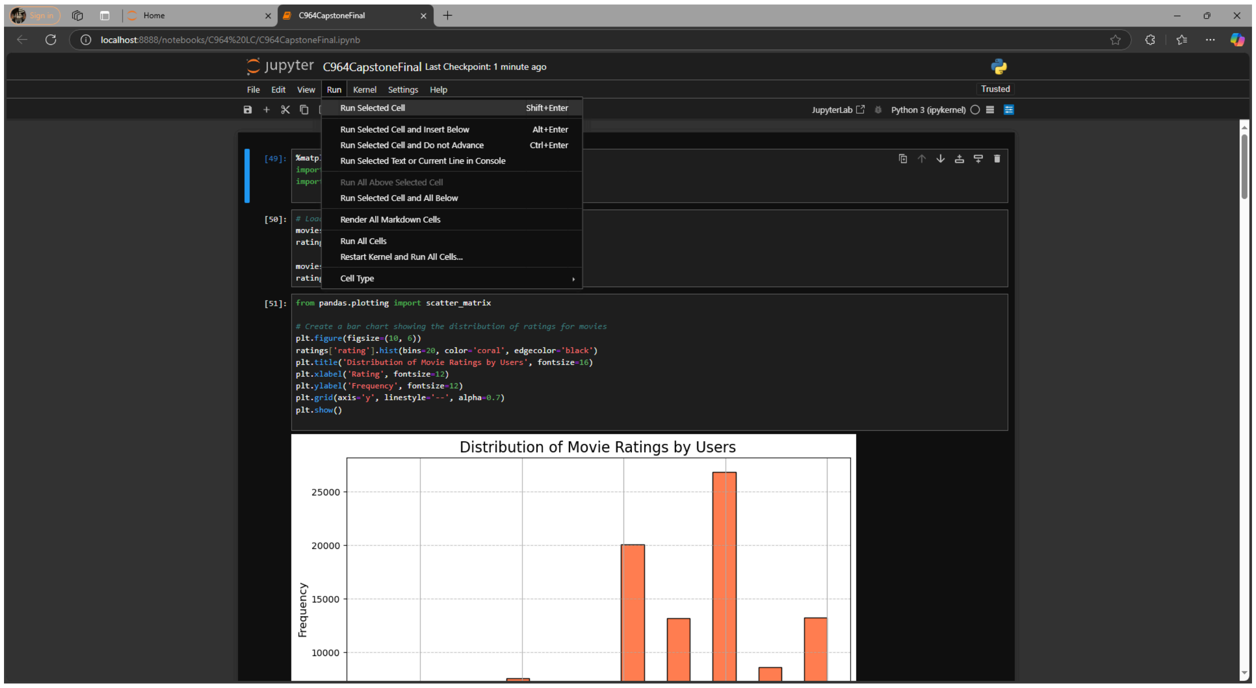
6. Your internet browser will open Jupyter Notebook and your screen will appear like this.



| Name | Last Modified | File Size |
|---|---|---|
| 3D Objects | 3 years ago | |
| anaconda3 | 2 days ago | |
| ansel | 4 years ago | |
| bluej | 2 years ago | |
| C964 Capstone Folder | 20 seconds ago | |
| Contacts | 3 years ago | |
| Desktop | 7 days ago | |
| Documents | 6 months ago | |
| Downloads | 6 months ago | |
| eclipse | 2 years ago | |
| eclipse-workspace | last year | |
| Favorites | 3 years ago | |
| Games | 3 months ago | |
| Links | 3 years ago | |
| Music | 3 years ago | |
| Nanoleaf Desktop App | last year | |
| OneDrive | 3 years ago | |
| PycharmProjects | 4 months ago | |
| Saved Games | 4 months ago | |
| Searches | 3 years ago | |
| source | last year | |
| Tracing | 9 months ago | |
| Videos | 11 hours ago | |
| C964Capstone.ipynb | 2 days ago | 18.9 KB |
| C964CapstoneV2.ipynb | 2 days ago | 617 B |
| Load_Data.ipynb | 7 days ago | 2 KB |
| Pandas.ipynb | 7 days ago | 11 KB |

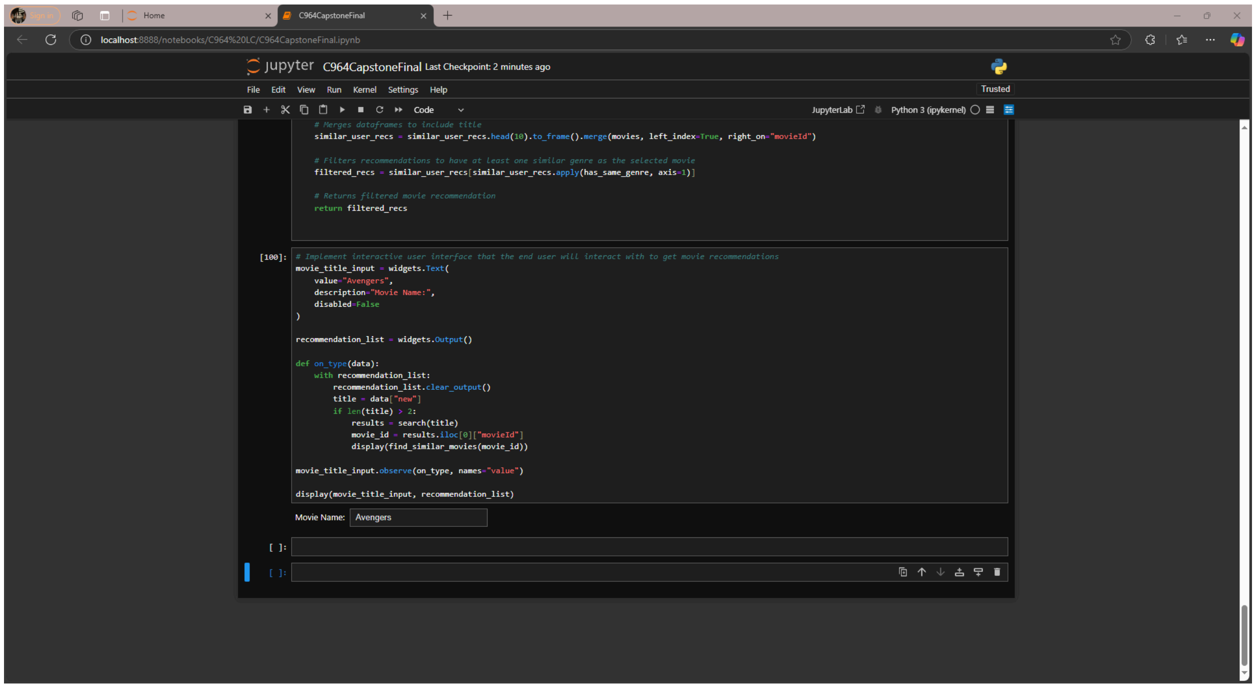
7. Once here, navigate through your file directory to where the folder was downloaded to and open the folder. Double click the C964CapstoneFinal.ipynb file.

8. At the top, click Run -> Run All Cells. Because Anaconda downloads packages for you, no additional libraries need to be manually downloaded.


Home
C964CapstoneFinal
localhost:8888/notebooks/C964%20LC/C964CapstoneFinal.ipynb
jupyter C964CapstoneFinal Last Checkpoint: 1 minute ago
File Edit View Run Kernel Settings Help
Trusted
JupyterLab
Python 3 (ipykernel)
Run Selected Cell Shift+Enter
Run Selected Cell and Insert Below Alt+Enter
Run Selected Cell and Do not Advance Ctrl+Enter
Run Selected Text or Current Line in Console
Run All Above Selected Cell
Run Selected Cell and All Below
Render All Markdown Cells
Run All Cells
Restart Kernel and Run All Cells...
Cell Type
Distribution of Movie Ratings by Users
Frequency
25000
20000
15000
10000

```
[49]: %matpl
      impor
      impor

[50]: # Loa
      movie
      ratin

      movie
      ratin

[51]: from pandas.plotting import scatter_matrix

      # Create a bar chart showing the distribution of ratings for movies
      plt.figure(figsize=(10, 6))
      ratings['rating'].hist(bins=20, color='coral', edgecolor='black')
      plt.title('Distribution of Movie Ratings by Users', fontsize=16)
      plt.xlabel('Rating', fontsize=12)
      plt.ylabel('Frequency', fontsize=12)
      plt.grid(axis='y', linestyle='--', alpha=0.7)
      plt.show()
```

9. The Notebook will redirect you to the bottom where the user will interact with.

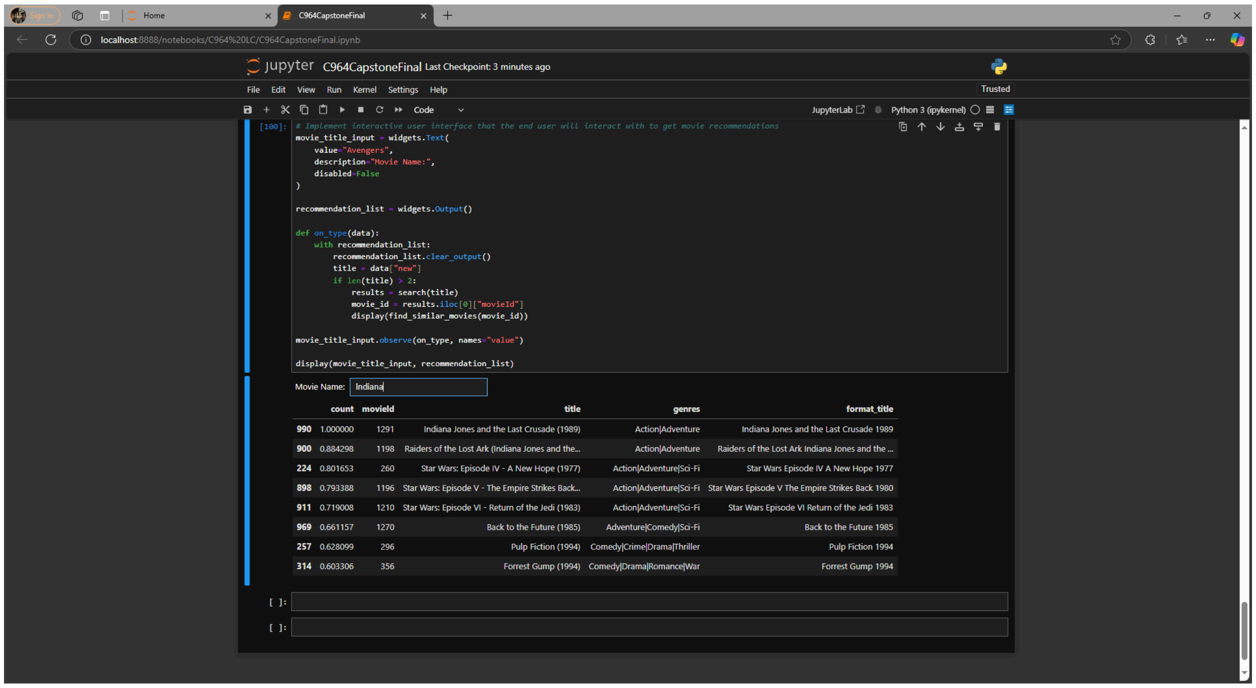

localhost:8888/notebooks/C964%20LC/C964CapstoneFinal.ipynb
jupyter C964CapstoneFinal Last Checkpoint: 2 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
    # Merges dataframes to include title
    similar_user_recs = similar_user_recs.head(10).to_frame().merge(movies, left_index=True, right_on="movieId")

    # Filters recommendations to have at least one similar genre as the selected movie
    filtered_recs = similar_user_recs[similar_user_recs.apply(has_same_genre, axis=1)]

    # Returns filtered movie recommendation
    return filtered_recs
```

```
[100]: # Implement interactive user interface that the end user will interact with to get movie recommendations
movie_title_input = widgets.Text(
    value="Avengers",
    description="Movie Name:",
    disabled=False
)

recommendation_list = widgets.Output()

def on_type(data):
    with recommendation_list:
        recommendation_list.clear_output()
        title = data["new"]
        if len(title) > 2:
            results = search(title)
            movie_id = results.iloc[0]["movieId"]
            display(find_similar_movies(movie_id))

movie_title_input.observe(on_type, names="value")

display(movie_title_input, recommendation_list)
```

Movie Name: Avengers

10. This bottom search bar labeled “Movie Name:” is where the user will input the name of a movie and the tool will output a list of movie recommendations.


Home
C964CapstoneFinal
localhost:8888/notebooks/C964%20LC/C964CapstoneFinal.ipynb
jupyter C964CapstoneFinal Last Checkpoint: 3 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
[100]: # Implement interactive user interface that the end user will interact with to get movie recommendations
movie_title_input = widgets.Text(
    value="Avengers",
    description="Movie Name:",
    disabled=False
)

recommendation_list = widgets.Output()

def on_type(data):
    with recommendation_list:
        recommendation_list.clear_output()
        title = data["new"]
        if len(title) > 2:
            results = search(title)
            movie_id = results.iloc[0]["movieId"]
            display(find_similar_movies(movie_id))

movie_title_input.observe(on_type, names="value")

display(movie_title_input, recommendation_list)
```

```
Movie Name: Indiana

        count  movieId                                          title                    genres                                        format_title
990  1.000000     1291      Indiana Jones and the Last Crusade (1989)          Action|Adventure        Indiana Jones and the Last Crusade 1989
900  0.884298     1198   Raiders of the Lost Ark (Indiana Jones and the...          Action|Adventure   Raiders of the Lost Ark Indiana Jones and the ...
224  0.801653      260       Star Wars: Episode IV - A New Hope (1977)   Action|Adventure|Sci-Fi           Star Wars Episode IV A New Hope 1977
898  0.793388     1196  Star Wars: Episode V - The Empire Strikes Back...   Action|Adventure|Sci-Fi Star Wars Episode V The Empire Strikes Back 1980
911  0.719008     1210  Star Wars: Episode VI - Return of the Jedi (1983)   Action|Adventure|Sci-Fi     Star Wars Episode VI Return of the Jedi 1983
969  0.661157     1270                     Back to the Future (1985)   Adventure|Comedy|Sci-Fi                         Back to the Future 1985
257  0.628099      296                           Pulp Fiction (1994)  Comedy|Crime|Drama|Thriller                             Pulp Fiction 1994
314  0.603306      356                           Forrest Gump (1994)  Comedy|Drama|Romance|War                             Forrest Gump 1994
```

```
[ ]:
[ ]:
```