**Christopher Straka**

**Student ID:** 011551615

**Business Vision:**

**Objective:** Enhance user engagement and satisfaction on the streaming platform by creating a personalized movie recommendation system.

**Vision Statement:** To become a leading streaming service known for intuitive, user-centric content discovery and recommendation capabilities. To drive increased user retention and growth.

**Quick Start:** Please see the README.md file at the root of the project for a setup guide to get this project running locally.

**Datasets:** The data was pulled from Kaggle. The creator already cleaned the data, so there were no missing values or anomalies. The ratings are already normalized between 0 and 5. I changed the variables to snake case to fit python naming conventions, but nothing else was done to alter the dataset. All Kaggle data is in the model/data folder. The datasets were collected from this link <https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset>.

**Data Analysis And Data Product Code:**

To start things off, I wanted to get a brief look at the data to see if it looked like what I was expecting. So, I printed the head of both tables to see if it resembled what I saw on Kaggle.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

This checked out. I then wanted to verify if there were any missing ratings from the dataset, and that the ratings were between 0.5 and 5, just to make sure that there wouldn’t be any issues.

A screen shot of a computer

Description automatically generated**A screen shot of a computer program

Description automatically generated**

After verifying that there wasn’t anything wrong with the dataset, I then moved on to visualizing the data which I’ll talk about later, and splitting up my data so that I could train and measure the effectiveness of my model. I split up the data so 25% of the data would not be trained with (my test set). It’s important, because this is the data I’m using after my model is trained to see how close my model could predict the user’s expected rating, vs what the user’s rating ended up as.

A screenshot of a computer program

Description automatically generated

My algorithm to make my predictive model work is the SVD algorithm. Once trained, this SVD algorithm becomes my model. It needs data so that I can train it and get the right parameters for it. The SVD algorithm is a collaborative filtering algorithm, this means it makes recommendations to people, based on what similar people have said. This contrasts with content-based filtering, which uses explicit feedback from the user and explicit movie metadata that we create (genre, year, etc.) My algorithm is unsupervised because it discovers features within the data. Maybe there’s different sub genres that we didn’t know existed, or maybe there’s something connecting the movies that the users can see but isn’t captured in metadata. The metrics that I use to evaluate the model’s accuracy will be covered in its own dedicated section later.

**Hypothesis Evaluation:**

**Hypothesis:** If we implement a recommender system using collaborative filtering, then we will see an increase in user engagement as it provides more relevant movie suggestions based on user ratings and the ratings of similar users.

I accept the hypothesis because having recommendations make the service much more engaging. When inspecting user 5’s ratings and recommendations, the result is a very natural fit. When they rated Grumpier Old Men (1995) and Clueless (1995), both romantic comedies, the model recommended them It Happened One Night (1935) an Indiana Jones and the Last Crusade (1989). The former is a romantic comedy, and the latter has elements of romance and comedy despite being an action-adventure movie (something a content-based algorithm might not have picked up on). Without these suggestions, the user might’ve been shifting through movies like The Grudge (2004), an obvious bad pick. So, the hypothesis is confirmed.

A screenshot of a computer

Description automatically generated

**Visualizations:**

After I downloaded the dataset and verified its properties, I started by looking at what the top 10 most rated movies were using a bar chart. It would be helpful to know if some movies have almost all the reviews and others are left sparse. Visualizing every movie’s rating count in a bar chart is not feasible because there are too many, so I limited the visual to just ten films.

VISUAL NEXT PAGE

A screen shot of a computer screen

Description automatically generated

Now that I know what movies are popular and that the ratings count is reasonable, I'd like to know what number rating (0.5 – 5) was the most popular. This is important to know, because the more varied the user ratings are, the richer the dataset is for my algorithm and the easier it is to tell what people like. If everyone gave every movie 5 stars, then my algorithm would just recommend everything to everybody, which doesn’t make for a good algorithm. So, I created a histogram that had rating counts on the y axis and ratings given on the x axis.

A screen shot of a graph

Description automatically generated

This distribution is sort of what I expected. There’s bias for users to like a movie than not, but there’s still a decent distribution that I can use to make predictions. To look over what I looked at so far in more detail, I wanted to know what the average rating is for a movie with respect to its rating count. Do movies with more ratings have better reviews?

VISUAL NEXT PAGE

**A screen shot of a computer screen

Description automatically generated**

There are several conclusions to be made from this scatterplot. The bottom purple group of movies that tend to have lower rating counts also have most of the low ratings when compared to any other group. There’s also the widest spread in average rating out of any other group. So, there’s a lot of movies people haven’t seen that are good. As you move up, the rating counts start to spread more as you go up each group. The rating also tends to get better.

These conclusions help set the stage for my predictive algorithm by giving us insights into what my data looks like and what movies the model is more likely to predict. There are potential gems that don’t have many ratings yet and might be missed, and there are popular well received movies that are likely to be overrepresented in the predictions because we have lots of positive data for them.

After I trained my model, I wanted to get a good idea of what my predictions looked like for each user, so I created a heatmap to visualize the model’s recommendations for 15 users with respect to 15 movies.

A screen shot of a computer screen

Description automatically generated

The grid cells with the brighter colors means that the model predicts the user will like that movie and should be recommended to the user. The dark colors means that those movies should not be recommended for those users.

**Product Accuracy Assessment:** The product will primarily be assessed using its RMSE and MAE scores. An RMSE (Root Mean Squares Error) and an MAE (Mean Absolute Error) score of >= 1 will be considered a bad score and hence, a failed recommendation system. The final score was 0.8888 for RMSE and 0.6859 for MAE. This falls within the acceptable parameters set out in the executive summary.   
  
RMSE (Root Mean Square Error) measures the square root of average squared errors between predicted and actual values. MAE (Mean Absolute Error) measures the average of absolute differences between predicted and actual values. RMSE penalizes larger errors more than MAE. Both were used to measure the model’s accuracy.

**Testing, Revisions, Optimization:**

Testing: For product testing, I manually checked what the results were for user 5 to see if they seemed reasonable from my own perspective. The results were better than I expected and explained in the hypothesis evaluation section.

A screenshot of a computer

Description automatically generated

I also split the data set into a test set and a train set and measured the model’s accuracy using RMSE and MAE scores.   
  
A screenshot of a computer program

Description automatically generated

Finally, I included a few unit tests to check if a few functions were behaving correctly in the ratings.py file. I want the movie ids to convert to imdb ids so that the user can click on movies in the UI and be brought to the imdb page. I also want to make sure that I’m grabbing all the movies a user rated and not just some of them.

A screen shot of a computer program

Description automatically generated

A screen shot of a computer

Description automatically generated

Revisions:I dropped Rust, which I originally planned to use with Tauri, and used a python GUI instead. This made it easier to deploy and reason with as everything was in a single language and much easier to package. I also dropped Tkinter because that wasn’t working for the M1 Mac on lower versions of python, so I used Kivy instead.

A screen shot of a computer program

Description automatically generated

Optimizations: Earlier when developing this application, I originally trained the model every time a user asked for their personal recommendations. This would be a computationally expensive way of doing it. So, I instead optimized this process by training the model a single time and then saving it to a pickle file. The model will have to be updated in the future by calling the train() method that I made available manually, but this is a simple process that can be done nightly with a chron job, not every time the user wants a recommendation.

**Source Code:** Please see the Github for this repository.

<https://github.com/chris56974/C964>

**Quick Start:** Please see the README.md file at the root of the project for a setup guide to getting this project running locally.

<https://github.com/chris56974/C964/blob/main/README.md>