Movie Recommendation

Model

Done by: Zachary Kizell, Brian Cabingan, Marcus Dipalma Abdelaziz Ahmed

**Project Summary**

A movie rental place has one copy of each movie available for customers to rent. Given the customer’s preferences, this model aims to assess whether it is possible for every customer to rent a movie they will enjoy.

* It takes the preferences of the customer. This is either generated by the model or can be inputted by the user.
* It asks on how many people are using the module.
* It recommends a movie based on the customer’s preferences.

# Propositions

* Customers (Ci,j)  
  How many customers? All other questions are filled out based off how many customers you choose, only one person can rent a movie at a time. C represents a customer preference, where i is the customer number and j is the customer preference
* Genre (Gi):  
  Customer can choose their preferred genre of movie. The genres included in this project are: Action, Adventure, Animation, Biography, Comedy, Crime, Drama, Fantasy, History, Horror, Mystery, Romance, Sci-Fi, Thriller, Western. G is proposition, i represents the genre. True if movie genre matches customer preferences  
  Each customer can only choose one genre of movie they prefer.
* Quality (Ij):  
  Choose a minimum level of quality or no preference. Users can choose an IMDB score from 7-9 and any movie with a score equal to or higher than that will have their quality proposition set to true. Q represents qualtity and j is the minimum IMDB score. True if score is higher than customer’s preferred score.
  + The regular model only creates user props for IMDB scores 7-9 since no movie in the database we are using has a score of less than 7, as well as to lessen the number of props created so the model runs faster. This can easily be changed if this model were to be used with a different dataset
* Runtime (Rk):  
  Choose a length for your movie choice.
  + Customer can choose for their movie to be either short or long, where short means the movie is less than 2 hours and long means the movie is 2 hours or more.
  + R is runtime, k can be “short” or “long”
* Popularity (Pm):
  + Customer can choose how popular they want the movie to be. Their 3 options are niche, average, or popular. Movies are put into these categories based off of their box office earnings.
  + P is popularity, m is popularity of movie (N for niche, A for average, or P for popular)
  + True if movie popularity matches customers preferences for movie popularity
* Age (Ay):
  + Customer can choose which decade they want their movie to be from
  + A is age, y is decade movie was made in
  + True if proposition matches customer preferences

# Constraints

-Each customer (C) can only choose 1 movie(M) to rent (R).

* $x.C(x), "y. M(y), "x. (C (x) ∧ M(y) → R (x))

-no two customers can have the same movie recommended (even if all their preferences are the same)

- $x. "y.(C(x)∧ R(x) → ¬(R (x) ∧C(y)))

- For a movie selection for each person to be true, the customer preferences for genre, quality, runtime, popularity, and age to be true must align with the movie’s genre, quality, runtime, popularity, and age.

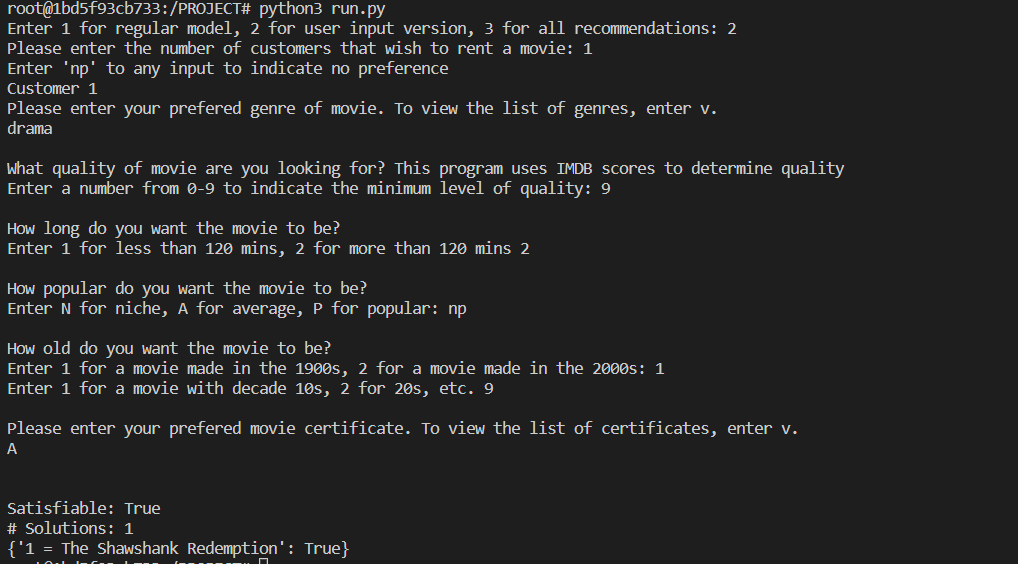
- For the model to be complete every person needs to have a movie that satisfied their constraints.

- $x."y.(C(y) → R(x))

# Model Exploration

* During an early version of this model, we encountered an issue where the model would fail or show near infinite solutions. This is because we did not realise that if a constraint failed then the model would be thrown away. There was a constraint that would recommend the movie if all the propositions associated with it were also true. Clearly this would not work in all situations. This version of the model can be seen in the draft version and was fixed when we removed the implication and implemented a constraint that made it so exactly one movie would be recommended.
* One thing we added was the option for the user (or users) to define their own preferences. This was helpful for exploring our solutions since we could define preferences that should cause the model to return a specific movie we had decided on beforehand. It also allowed us to check if the model would allow two users to rent the same movie. We did this by entering the same set of preferences for a movie whose properties (genre, rating, etc.) were unique to that movie for two or more users, and when there were no solutions, we knew the model did this correctly. It also helped to test how the model would react when multiple movies had preferences that fit the user’s preferences. Through this, we were able to troubleshoot so the model produced the correct number of solutions.

See below for an example of the user preference model:



* A function was created to explore the solution that returns all of the movies that fit the users’ criteria instead of just one. Through this, we were able to visually check to determine if the model was only recommending movies that fit the user criteria.
* Additionally, we added the option for users to test the various constraints by choosing a type of customer preference or aspect of the movie (genre, age, runtime, etc.) and negate all of the props of that type. Given that the movie needs to match all the customer preferences, this should always result in 0 solutions. Using this function we were able to test the constraints and there were always 0 solutions, except for when the customer IMDB rating preferences were all negated. This is because the model does not account for a situation where these props are all negated. Fortunately, this is not an issue as there will always be at least one customer IMDB rating prop that is true.

# First-Order Extension

To describe how our setting can be modeled using predicate logic:

* "x. $y. (C (x) ∧ M(y) → R (x, y)):

For all customers x, there exists a movie y that meets their preferences. If there is customer x and the movie y, then customer x will rent movie y.

* "x. $z(C (x) ∧ G(z) ∧ I(z) ∧ R(z) ∧ P(z) ∧ A(z)) → M (x, z)):

For all customers x, there exists a set of preferences z that they picked. If there is customer x and the preferences z, then the program will recommend a specific movie based on z to x.

* "x. $r. (C (x) ∧ R(r)) → M (x, r)):

For all customers x, there exists a rating r that they picked. If there is customer x and the rating r, then the program will recommend a specific movie based on r. The customer needs to select a popularity rating, the popularity ratings are 1-3 (3 being the most) and no preference.

* "x. $a. (C (x) ∧ A(a)) → M (x, a)):

The customer needs to select an age for the movie, the age options are in decades, the customer will select what decade they want the movie to be from

* *the customer(s) has selected a given option for every category and needs a movie back that has the same as what they picked, there exists a movie for every case scenario, therefore the customer will receive back a movie that meets their given criteria.*

# Jape

This jape proof proves that for every customer there exists a movie that meets their preferences that the customer can rent. C is the customer, A represents all of the aspects of the movie (genre, rating, etc.), R represents the customer x renting a movie.

Graphical user interface, application

Description automatically generated

This Jape proof explains how there can only be one rented movie per customer

C is customer, R is renting a movie.

Graphical user interface

Description automatically generated

This Jape proof is to show that each customer must have a rented movie in order for the model to be satisfied.

Table

Description automatically generated

# Additional Notes

* The model currently includes the first 100 movies in the spreadsheet. This can be changed by by modifying if statement on line 141 to x == whatever number you want.
* When run normally, there should be between 1 and 3 solutions for each movie. This is because the model will succeed for IMDB scores 7,8 and 9 if the movie has IMDB score 9. While it may seem as though you are getting more solutions than you would think, this is the model behaving as intended.