Data Glacier

Your Deep Learning Partner

Web Application Deployment

Movie Recommendations

Apr 3rd,2023
Virginia Mullins
LISUM 19

Application Overview

Application Function

Data Sets

Modeling

App.py

HTML Templates

Deployment

Data Glacier

Your Deep Learning Partner

Application Function

I chose to use the same model and data as the flask application from Week 4

Movie Recommendation App

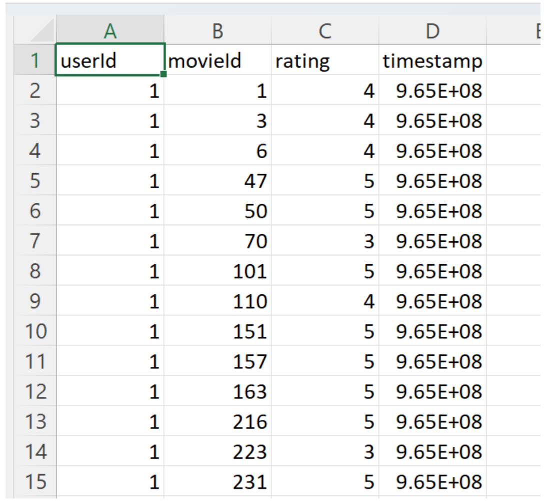
The application takes the users’ ID number and a selected genre and passes through the recommendation model to return a list of unrated movies for the user.

The recommendations are made using predicted rating scores based off of the users ratings of other movies.

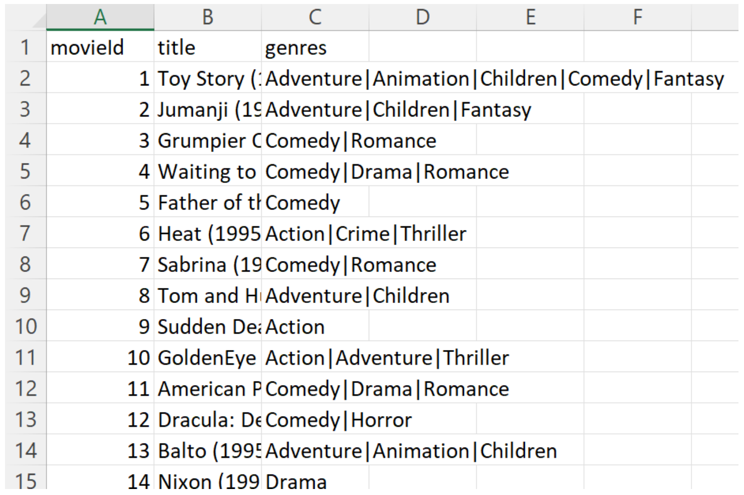
Data Sets

The data sets were obtained from the Movielens database. For this app, we used two of the provided files related to a set of 610 individual users and 9742 films.

In ‘ratings.csv’ we have each 4 features and 100837 occurrences; userId, movieId, rating, timestamp
In ‘movies.csv’ we 3 features and 9742 occurrences; movieId, title, genre
the genre column contains a string that includes every genre the movie classifies for.



	A	B	C	D
1	userId	movieId	rating	timestamp
2	1	1	4	9.65E+08
3	1	3	4	9.65E+08
4	1	6	4	9.65E+08
5	1	47	5	9.65E+08
6	1	50	5	9.65E+08
7	1	70	3	9.65E+08
8	1	101	5	9.65E+08
9	1	110	4	9.65E+08
10	1	151	5	9.65E+08
11	1	157	5	9.65E+08
12	1	163	5	9.65E+08
13	1	216	5	9.65E+08
14	1	223	3	9.65E+08
15	1	231	5	9.65E+08



	A	B	C
1	movieId	title	genres
2	1	Toy Story (	Adventure|Animation|Children|Comedy|Fantasy
3	2	Jumanji (19	Adventure|Children|Fantasy
4	3	Grumpier C	Comedy|Romance
5	4	Waiting to	Comedy|Drama|Romance
6	5	Father of th	Comedy
7	6	Heat (1995	Action|Crime|Thriller
8	7	Sabrina (19	Comedy|Romance
9	8	Tom and H	Adventure|Children
10	9	Sudden Dea	Action
11	10	GoldenEye	Action|Adventure|Thriller
12	11	American P	Comedy|Drama|Romance
13	12	Dracula: De	Comedy|Horror
14	13	Balto (1995	Adventure|Animation|Children
15	14	Nixon (199	Drama

SVD modeling over the cloud

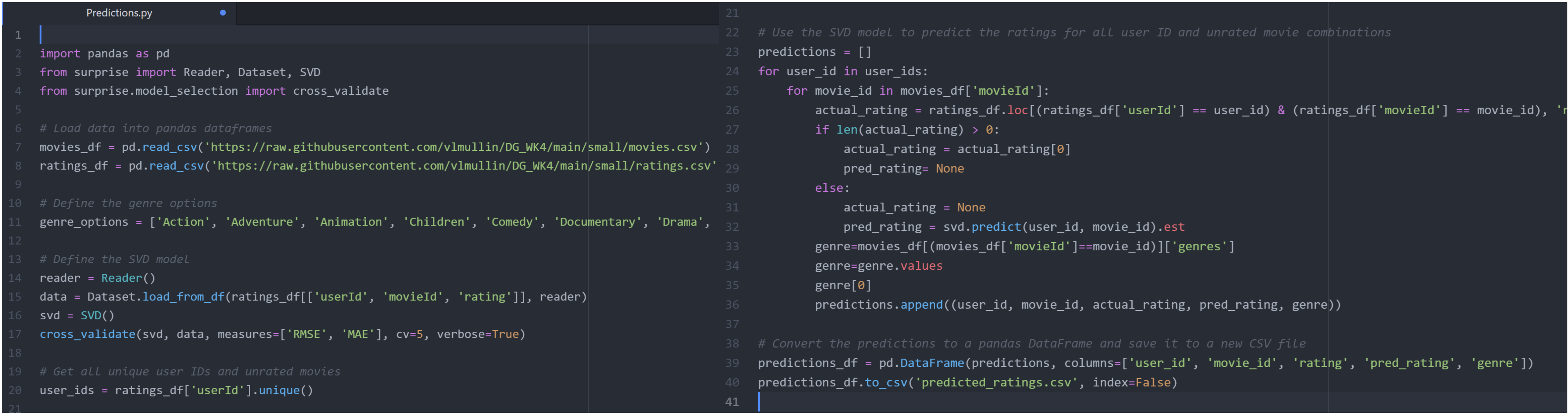
I chose to deploy my application through pythonanywhere.com and I intended to use the same flask code as I used on my local machine. However, there is a restriction on the amount of RAM that one can use on the server provided by pythonanywhere.com. This restriction made it impossible to first gather the user data through the home page, filter through the data for the specific user ID and genre selections, run it through the model, and then retrieve the recommendations back from the model; it was too computationally demanding. I tried multiple method of streaming lining the model (such as using sparse matrices, PCA, and gzip), but I was unsuccessful in implementing them into the flask application.

As an alternative to running the model after filtering the data, I decided to run the SVD model on the entire dataset at once and save the predicted ratings in a separate .csv file for the program to reference. I ran the SVD model on my local machine (it took ~ 15 hours). My intention was to push the ‘predicted_ratings.csv’ file onto the same github repository as the original movielens data, but I was unable to push the file due to its size, even using the large file storage (lfs) extension or the github desktop app. Consequently, I never got the app running properly over the web.

The following pages will show the apps function as it is ran locally. I would love feedback and recommendations to fix these issues so that I can learn to do this process properly.

Modeling

I chose to use a Singular Value Decomposition (SVD) model from the scikit-surprise module. The SVD model is a type of collaborative filtering model that uses a factorization technique to predict user ratings for items based on their historical ratings and the ratings of other similar users. I cross-validated the model by splitting the data into multiple subsets, training the model on some of the subsets, and testing it on the remaining subset. For this application, I chose 5 folds, which means the data was split into 5 subsets, and the model was trained and tested 5 times.

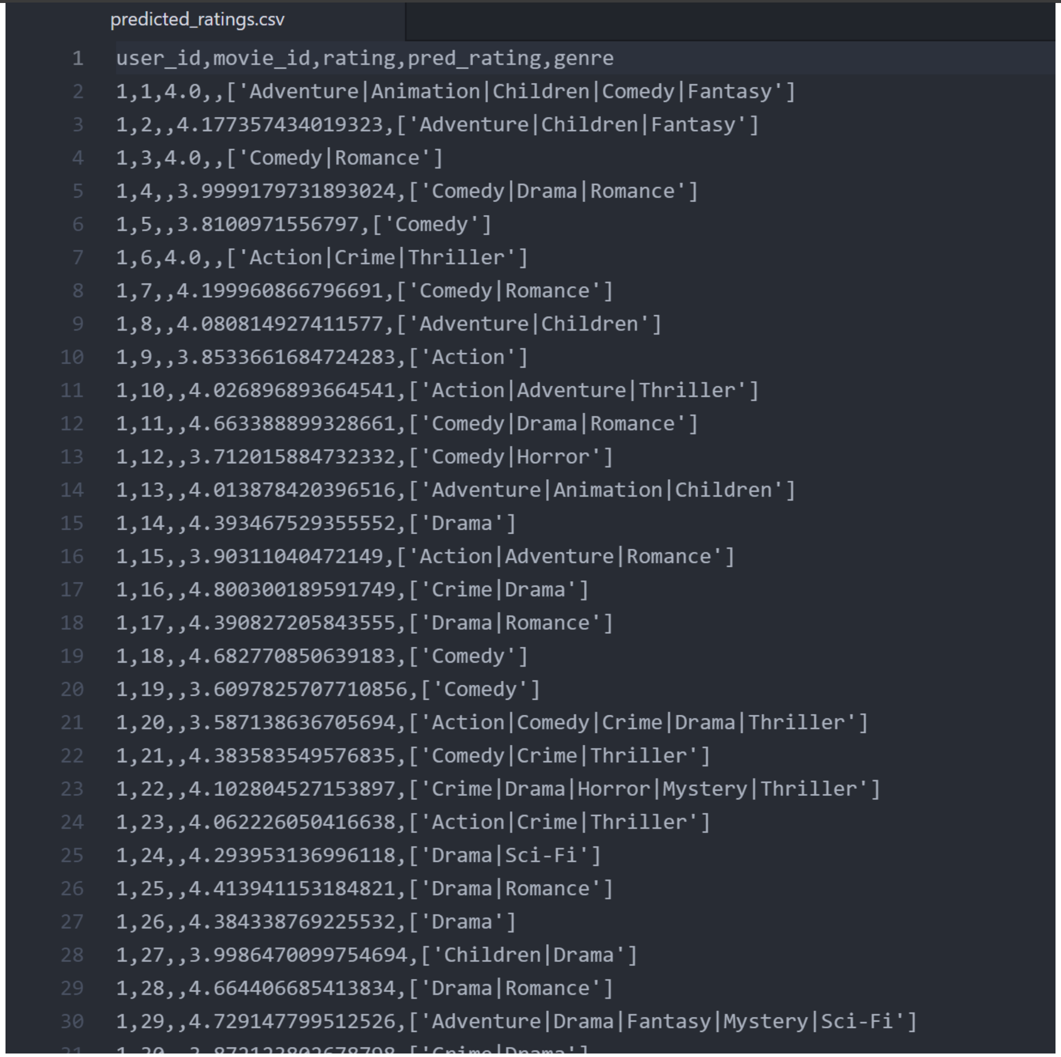


```python

import pandas as pd
from surprise import Reader, Dataset, SVD
from surprise.model_selection import cross_validate

# Load data into pandas dataframes
movies_df = pd.read_csv('https://raw.githubusercontent.com/vlmullin/DG_WK4/main/small/movies.csv')
ratings_df = pd.read_csv('https://raw.githubusercontent.com/vlmullin/DG_WK4/main/small/ratings.csv'

# Define the genre options
genre_options = ['Action', 'Adventure', 'Animation', 'Children', 'Comedy', 'Documentary', 'Drama',

# Define the SVD model
reader = Reader()
data = Dataset.load_from_df(ratings_df[['userId', 'movieId', 'rating']], reader)
svd = SVD()
cross_validate(svd, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)

# Get all unique user IDs and unrated movies
user_ids = ratings_df['userId'].unique()


# Use the SVD model to predict the ratings for all user ID and unrated movie combinations
predictions = []
for user_id in user_ids:
    for movie_id in movies_df['movieId']:
        actual_rating = ratings_df.loc[(ratings_df['userId'] == user_id) & (ratings_df['movieId'] == movie_id), 'r
        if len(actual_rating) > 0:
            actual_rating = actual_rating[0]
            pred_rating= None
        else:
            actual_rating = None
            pred_rating = svd.predict(user_id, movie_id).est
        genre=movies_df[(movies_df['movieId']==movie_id)]['genres']
        genre=genre.values
        genre[0]
        predictions.append((user_id, movie_id, actual_rating, pred_rating, genre))

# Convert the predictions to a pandas DataFrame and save it to a new CSV file
predictions_df = pd.DataFrame(predictions, columns=['user_id', 'movie_id', 'rating', 'pred_rating', 'genre'])
predictions_df.to_csv('predicted_ratings.csv', index=False)

```

Predicted_Ratings.csv

4 features with 5942620 occurrences

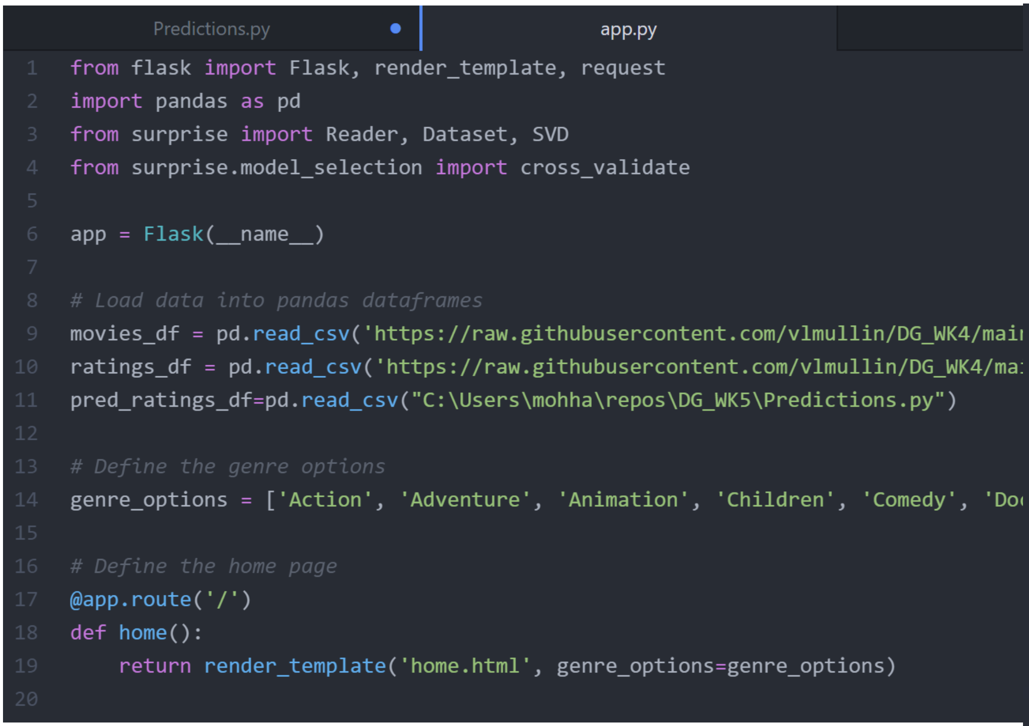
As you can see, for each userId & movieId combo, there is either a rating OR a pred_rating



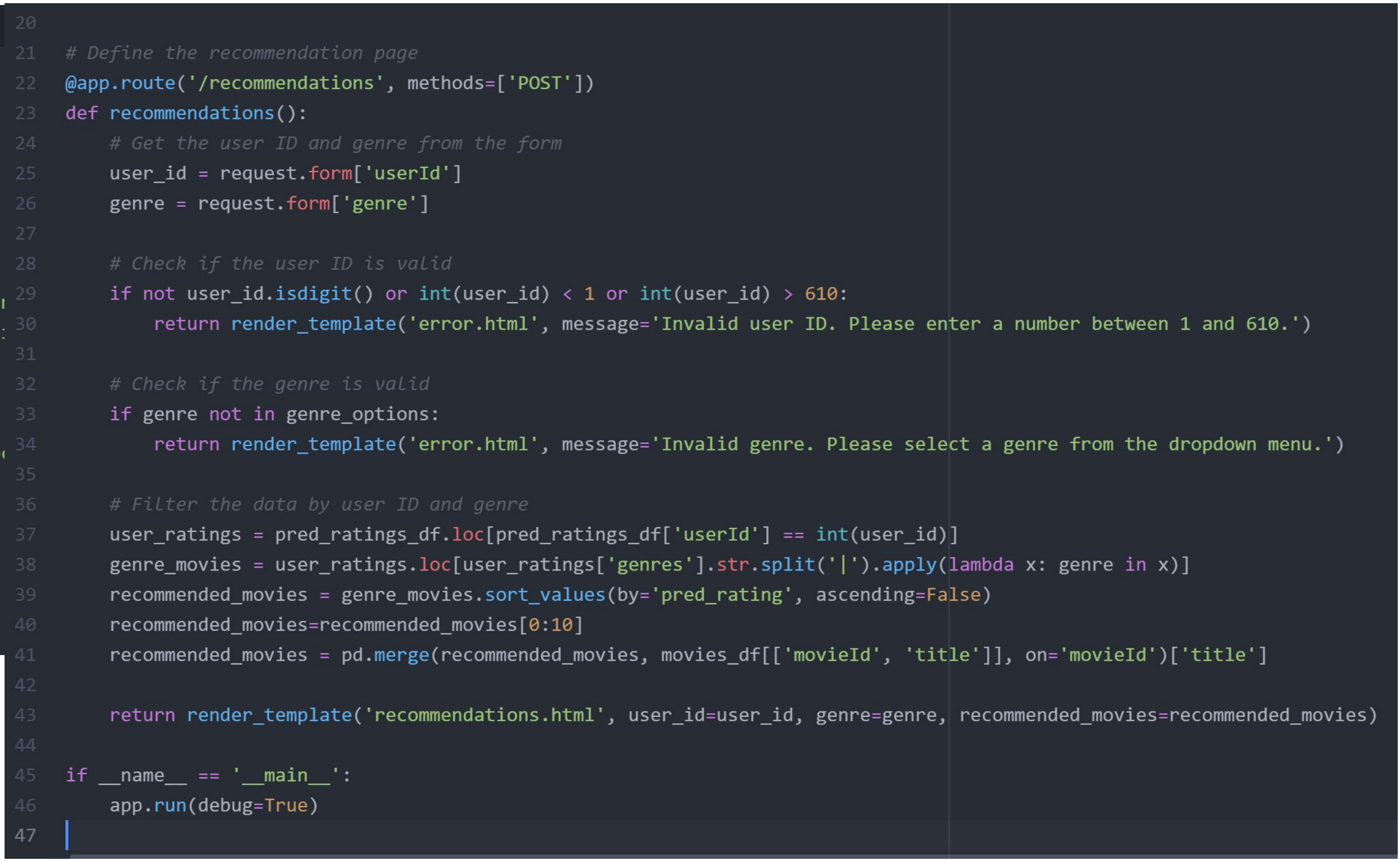
predicted_ratings.csv

```
user_id,movie_id,rating,pred_rating,genre
1,1,4.0,,['Adventure|Animation|Children|Comedy|Fantasy']
1,2,,4.177357434019323,['Adventure|Children|Fantasy']
1,3,4.0,,['Comedy|Romance']
1,4,,3.9999179731893024,['Comedy|Drama|Romance']
1,5,,3.8100971556797,['Comedy']
1,6,4.0,,['Action|Crime|Thriller']
1,7,,4.199960866796691,['Comedy|Romance']
1,8,,4.080814927411577,['Adventure|Children']
1,9,,3.8533661684724283,['Action']
1,10,,4.026896893664541,['Action|Adventure|Thriller']
1,11,,4.663388899328661,['Comedy|Drama|Romance']
1,12,,3.712015884732332,['Comedy|Horror']
1,13,,4.013878420396516,['Adventure|Animation|Children']
1,14,,4.393467529355552,['Drama']
1,15,,3.90311040472149,['Action|Adventure|Romance']
1,16,,4.800300189591749,['Crime|Drama']
1,17,,4.390827205843555,['Drama|Romance']
1,18,,4.682770850639183,['Comedy']
1,19,,3.6097825707710856,['Comedy']
1,20,,3.587138636705694,['Action|Comedy|Crime|Drama|Thriller']
1,21,,4.383583549576835,['Comedy|Crime|Thriller']
1,22,,4.102804527153897,['Crime|Drama|Horror|Mystery|Thriller']
1,23,,4.062226050416638,['Action|Crime|Thriller']
1,24,,4.293953136996118,['Drama|Sci-Fi']
1,25,,4.413941153184821,['Drama|Romance']
1,26,,4.384338769225532,['Drama']
1,27,,3.9986470099754694,['Children|Drama']
1,28,,4.664406685413834,['Drama|Romance']
1,29,,4.729147799512526,['Adventure|Drama|Fantasy|Mystery|Sci-Fi']
1,30,,3.872122802678708,['Crime|Drama']
```

App.py



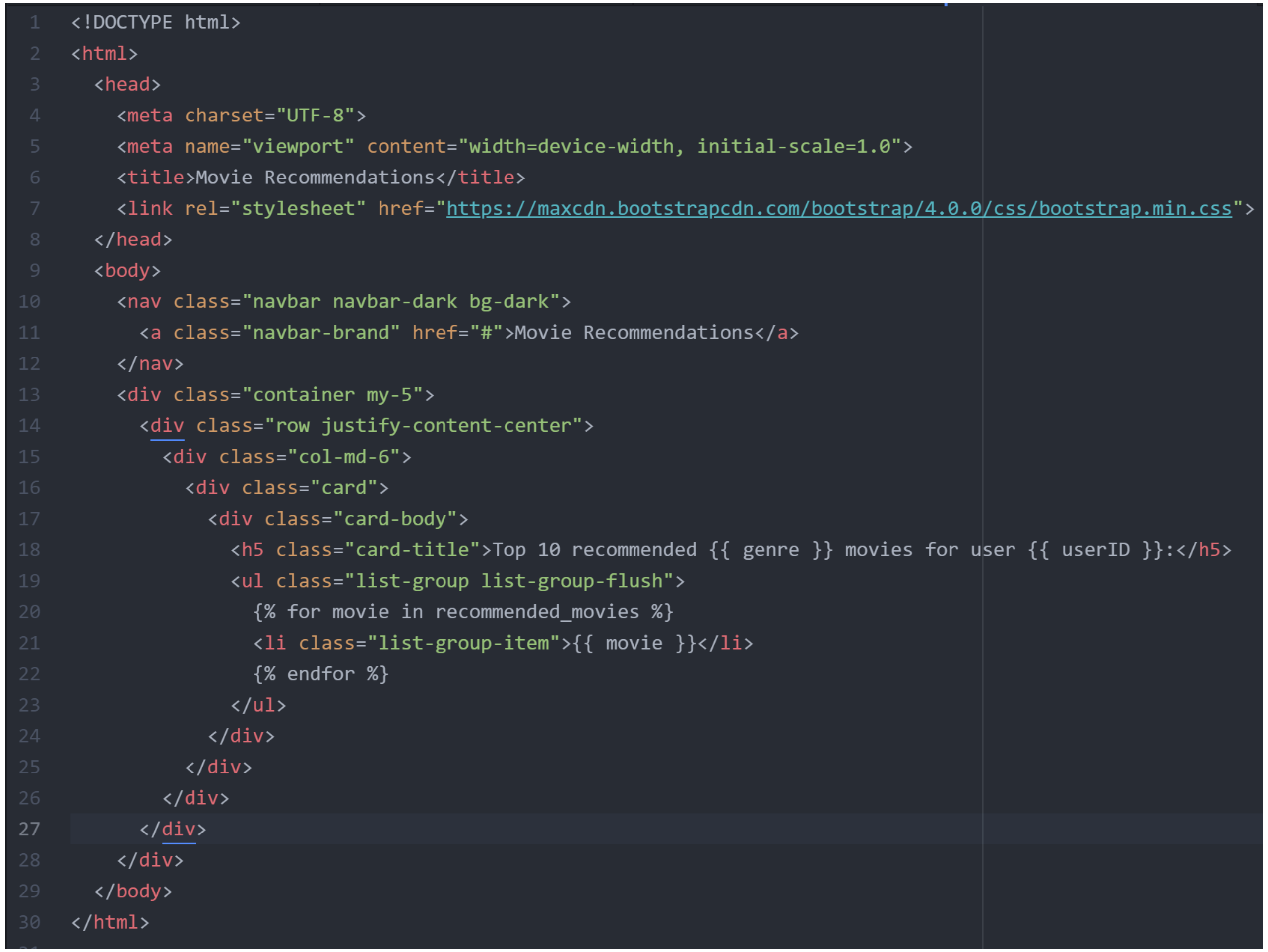
```python
from flask import Flask, render_template, request
import pandas as pd
from surprise import Reader, Dataset, SVD
from surprise.model_selection import cross_validate

app = Flask(__name__)

# Load data into pandas dataframes
movies_df = pd.read_csv('https://raw.githubusercontent.com/vlmullin/DG_WK4/mai
ratings_df = pd.read_csv('https://raw.githubusercontent.com/vlmullin/DG_WK4/ma
pred_ratings_df=pd.read_csv("C:\Users\mohha\repos\DG_WK5\Predictions.py")

# Define the genre options
genre_options = ['Action', 'Adventure', 'Animation', 'Children', 'Comedy', 'Do

# Define the home page
@app.route('/')
def home():
    return render_template('home.html', genre_options=genre_options)

```



```python

# Define the recommendation page
@app.route('/recommendations', methods=['POST'])
def recommendations():
    # Get the user ID and genre from the form
    user_id = request.form['userId']
    genre = request.form['genre']

    # Check if the user ID is valid
    if not user_id.isdigit() or int(user_id) < 1 or int(user_id) > 610:
        return render_template('error.html', message='Invalid user ID. Please enter a number between 1 and 610.')

    # Check if the genre is valid
    if genre not in genre_options:
        return render_template('error.html', message='Invalid genre. Please select a genre from the dropdown menu.')

    # Filter the data by user ID and genre
    user_ratings = pred_ratings_df.loc[pred_ratings_df['userId'] == int(user_id)]
    genre_movies = user_ratings.loc[user_ratings['genres'].str.split('|').apply(lambda x: genre in x)]
    recommended_movies = genre_movies.sort_values(by='pred_rating', ascending=False)
    recommended_movies=recommended_movies[0:10]
    recommended_movies = pd.merge(recommended_movies, movies_df[['movieId', 'title']], on='movieId')['title']

    return render_template('recommendations.html', user_id=user_id, genre=genre, recommended_movies=recommended_movies)

if __name__ == '__main__':
    app.run(debug=True)

```

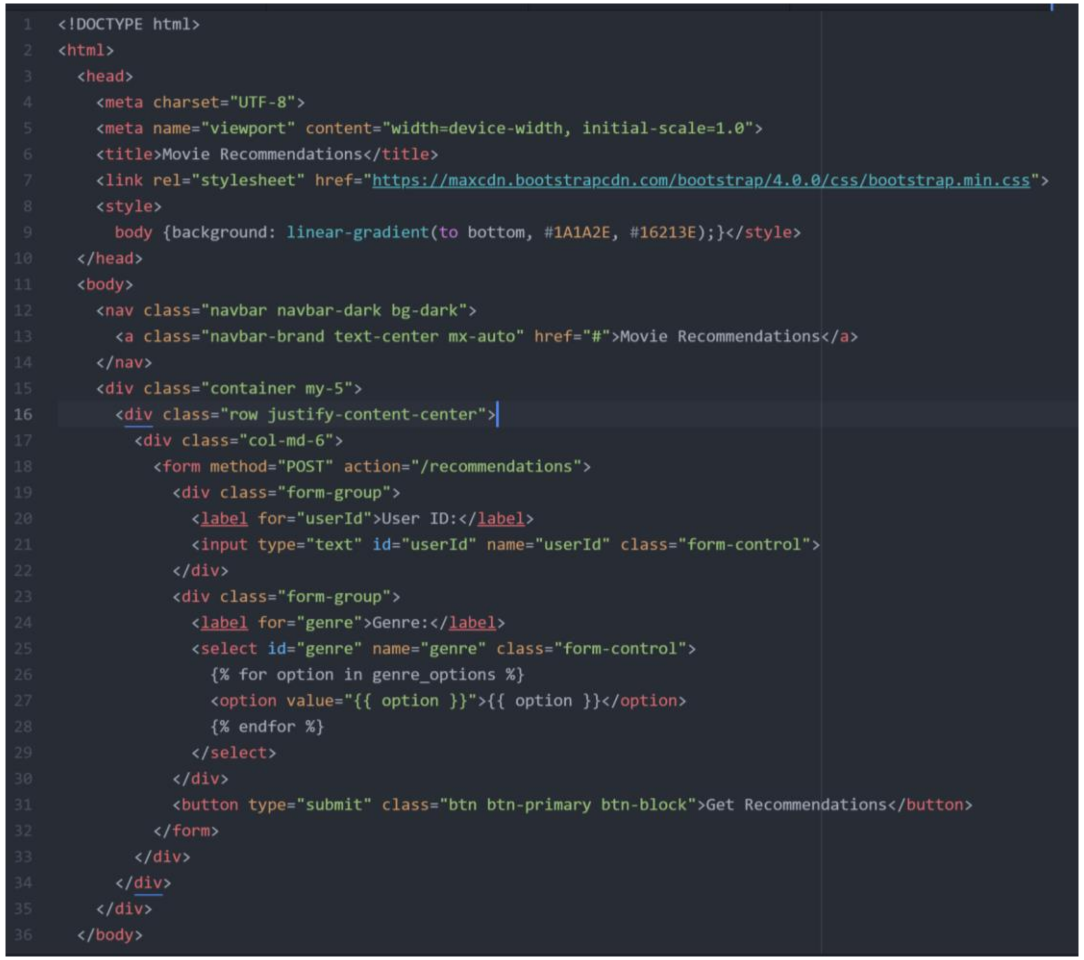
HTML Templates

Recommendations.html



```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Movie Recommendations</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
  </head>
  <body>
    <nav class="navbar navbar-dark bg-dark">
      <a class="navbar-brand" href="#">Movie Recommendations</a>
    </nav>
    <div class="container my-5">
      <div class="row justify-content-center">
        <div class="col-md-6">
          <div class="card">
            <div class="card-body">
              <h5 class="card-title">Top 10 recommended {{ genre }} movies for user {{ userID }}:</h5>
              <ul class="list-group list-group-flush">
                {% for movie in recommended_movies %}
                <li class="list-group-item">{{ movie }}</li>
                {% endfor %}
              </ul>
            </div>
          </div>
        </div>
      </div>
    </div>
  </body>
</html>
```

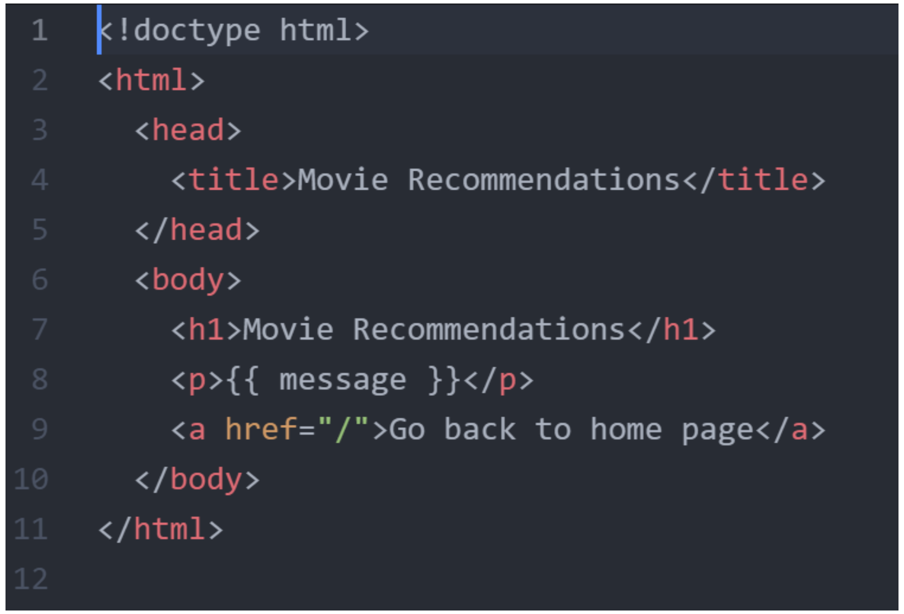
home.html



```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Movie Recommendations</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
    <style>
      body {background: linear-gradient(to bottom, #1A1A2E, #16213E);}</style>
  </head>
  <body>
    <nav class="navbar navbar-dark bg-dark">
      <a class="navbar-brand text-center mx-auto" href="#">Movie Recommendations</a>
    </nav>
    <div class="container my-5">
      <div class="row justify-content-center">
        <div class="col-md-6">
          <form method="POST" action="/recommendations">
            <div class="form-group">
              <label for="userId">User ID:</label>
              <input type="text" id="userId" name="userId" class="form-control">
            </div>
            <div class="form-group">
              <label for="genre">Genre:</label>
              <select id="genre" name="genre" class="form-control">
                {% for option in genre_options %}
                <option value="{{ option }}">{{ option }}</option>
                {% endfor %}
              </select>
            </div>
            <button type="submit" class="btn btn-primary btn-block">Get Recommendations</button>
          </form>
        </div>
      </div>
    </div>
  </body>
```

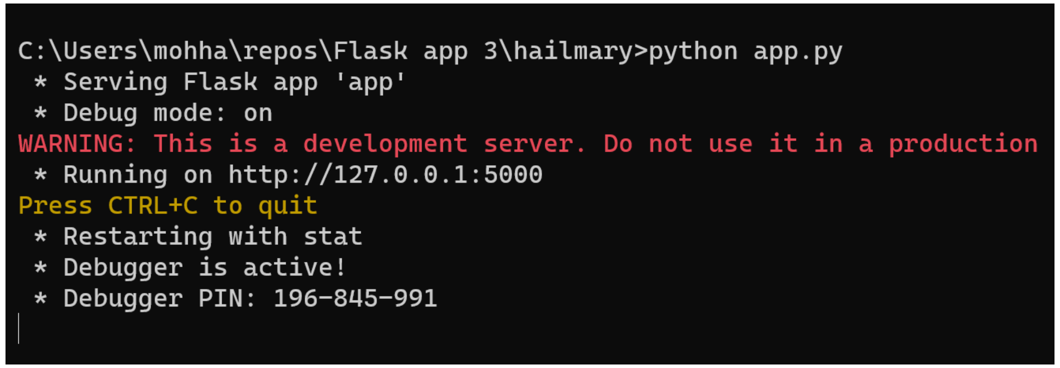
HTML Templates

error.html

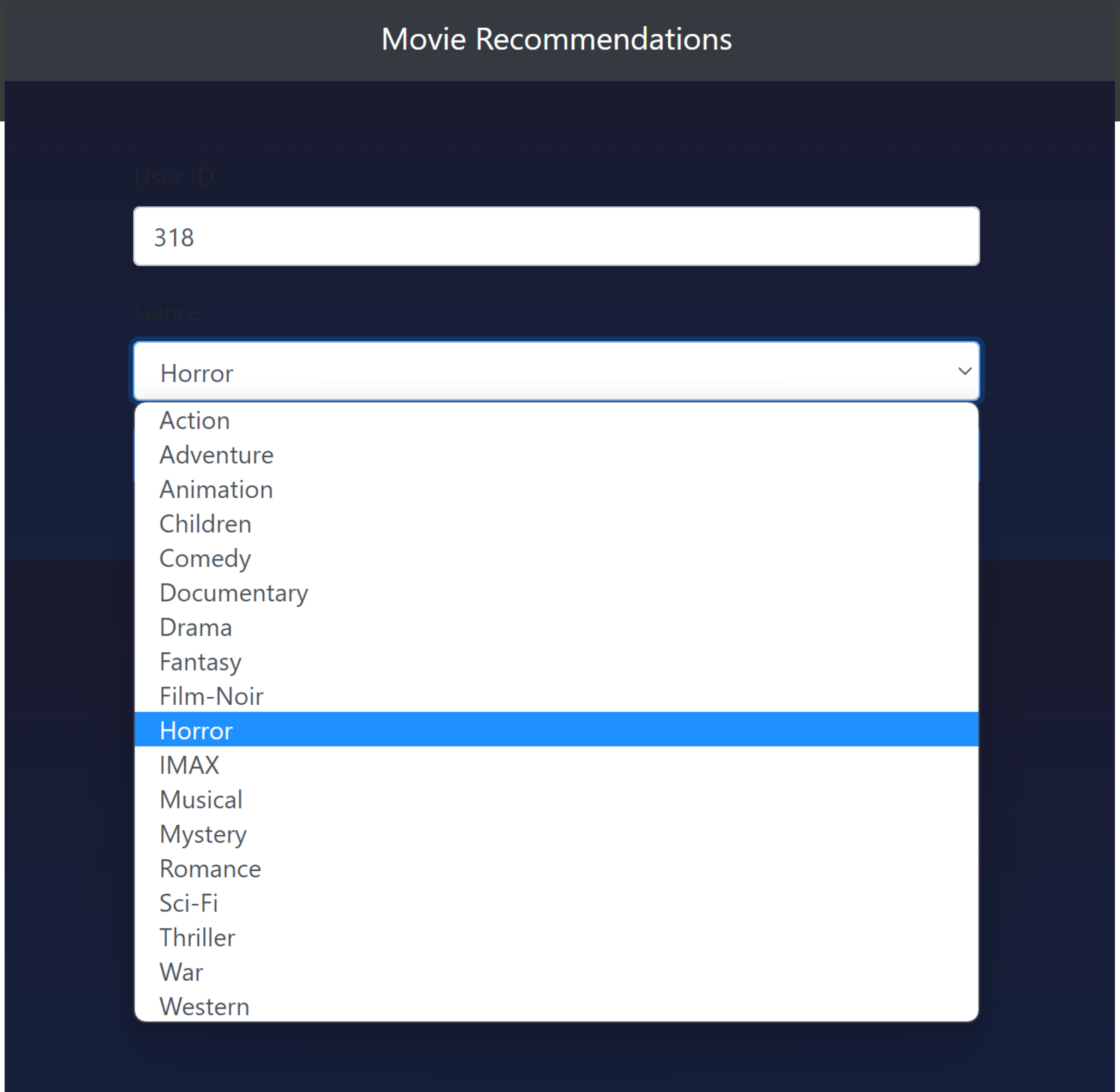


```html
<!doctype html>
<html>
  <head>
    <title>Movie Recommendations</title>
  </head>
  <body>
    <h1>Movie Recommendations</h1>
    <p>{{ message }}</p>
    <a href="/">Go back to home page</a>
  </body>
</html>
```

Deployment on local server



```
C:\Users\mohha\repos\Flask app 3\hailmary>python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 196-845-991
```

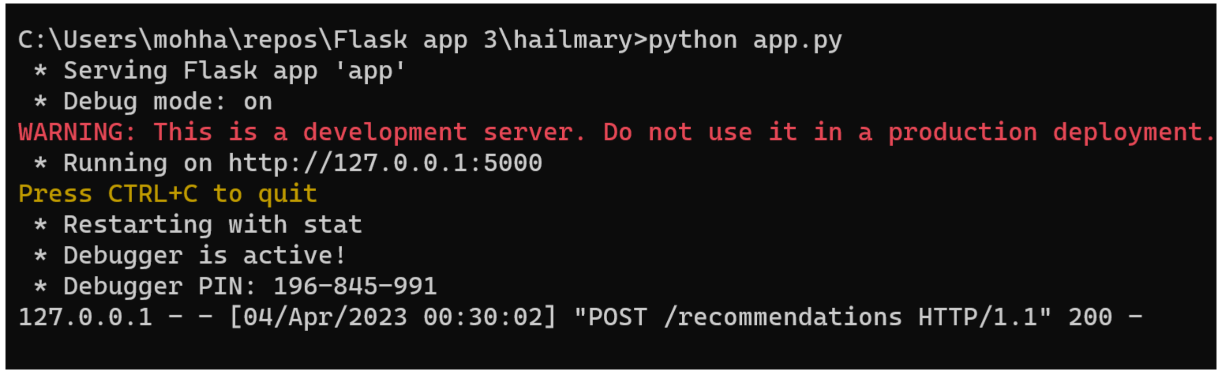


Movie Recommendations

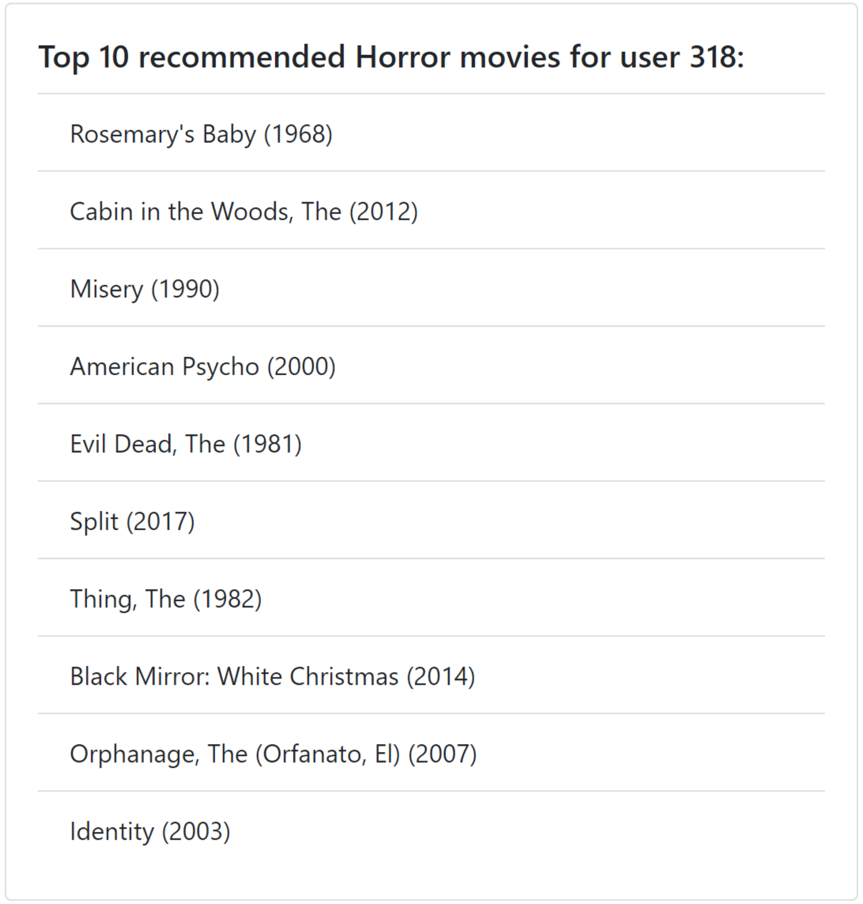
User ID: 318

Genre: Horror

Deployment on local server



```
C:\Users\mohha\repos\Flask app 3\hailmary>python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 196-845-991
127.0.0.1 - - [04/Apr/2023 00:30:02] "POST /recommendations HTTP/1.1" 200 -
```



Top 10 recommended Horror movies for user 318:

- Rosemary's Baby (1968)
- Cabin in the Woods, The (2012)
- Misery (1990)
- American Psycho (2000)
- Evil Dead, The (1981)
- Split (2017)
- Thing, The (1982)
- Black Mirror: White Christmas (2014)
- Orphanage, The (Orfanato, El) (2007)
- Identity (2003)

Thank You

Data Glacier

Your Deep Learning Partner