Application Deployment on Flask

Movie Recommendations

Apr 3rd,2023

Virginia Mullins
LISUM 19

Application Overview

- Application Function
- Data Sets
- Modeling
- App.py
- HTML Templates
- Deployment

Application Function

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	D	E	F
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 De	Action			
11	10	GoldenEye	Action|Adventure|Thriller			
12	11	American P	Comedy|Drama|Romance			
13	12	Dracula: De	Comedy|Horror			
14	13	Balto (199	Adventure|Animation|Children			
15	14	Nixon (199	Drama			

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.

```
14
15  # Define the SVD model
16  reader = Reader()
17  data = Dataset.load_from_df(ratings_df[['userId', 'movieId', 'rating']], reader)
18  svd = SVD()
19  cross_validate(svd, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
20
21  # Define the home page
```

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/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', 'F

# 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)

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

# 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
```

```python

    # 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 th

    # Filter the data by user ID and genre
    user_ratings = ratings_df.loc[ratings_df['userId'] == int(user_id)]
    genre_movies = movies_df.loc[movies_df['genres'].str.split('|').apply(lambda x: genre in x)]

    # Filter the genre movies by unrated movies by the user
    unrated_movies = genre_movies[~genre_movies['movieId'].isin(user_ratings['movieId'])]

    # Use the SVD model to predict the ratings for the unrated movies
    predictions = []
    for movie_id in unrated_movies['movieId'].unique():
        predictions.append((movie_id, svd.predict(int(user_id), movie_id).est))
    predictions.sort(key=lambda x: x[1], reverse=True)

    # Get the top 10 predicted movies
    recommended_movies = []
    for i, (movie_id, _) in enumerate(predictions):
        recommended_movies.append(genre_movies.loc[genre_movies['movieId'] == movie_id]['title'].v
        if i == 9:
            break

    return render_template('recommendations.html', user_id=user_id, genre=genre, recommended_movie

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

```

HTML Templates

Recommendations.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

```
<!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

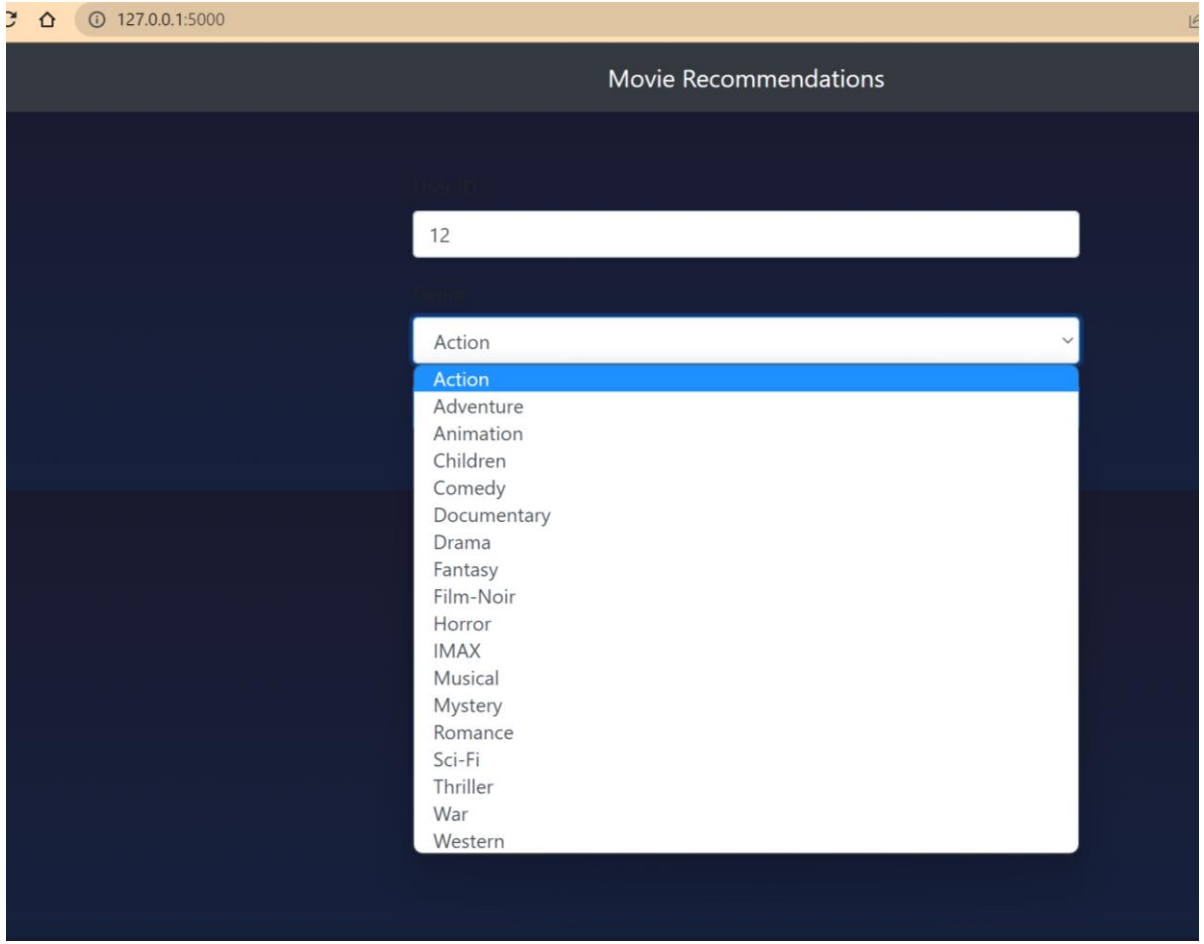
```
<!doctype html>
<html>
  <head>
    <title>Movie Recommendations</title>
  </head>
  <body>
    <h1>Movie Recommendations</h1>
    <p>{{ message }}</p>
    <a href="/">Go back to home page</a>
  </body>
</html>

```

Deployment on local server

```
C:\Users\mohha\repos\DG_WK4\Flask app 3>python app.py
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.8792  0.8763  0.8754  0.8715  0.8676  0.8740  0.0040
MAE (testset)     0.6731  0.6735  0.6749  0.6678  0.6684  0.6715  0.0029
Fit time          1.28    2.23    2.78    2.80    2.86    2.39    0.60
Test time         0.19    0.38    0.40    0.37    0.40    0.35    0.08
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a p
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```



Deployment on local server

```
WARNING: This is a development server. Do not use it in a production deployment. Use a produ
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.8717  0.8742  0.8715  0.8806  0.8740  0.8744  0.0033
MAE (testset)     0.6667  0.6741  0.6700  0.6773  0.6709  0.6718  0.0036
Fit time          1.56    1.80    2.87    2.99    2.90    2.42    0.62
Test time         0.24    0.40    0.41    0.42    0.39    0.37    0.07
 * Debugger is active!
 * Debugger PIN: 196-845-991
127.0.0.1 - - [03/Apr/2023 23:36:39] "POST /recommendations HTTP/1.1" 200 -
```

Top 10 recommended Action movies for user 12:

- Star Wars: Episode IV - A New Hope (1977)
- Léon: The Professional (a.k.a. The Professional) (Léon) (1994)
- Star Wars: Episode V - The Empire Strikes Back (1980)
- Princess Bride, The (1987)
- Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)
- Good, the Bad and the Ugly, The (Buono, il brutto, il cattivo, Il) (1966)
- Apocalypse Now (1979)
- Star Wars: Episode VI - Return of the Jedi (1983)
- Boot, Das (Boat, The) (1981)
- Great Escape, The (1963)

Thank You