GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Recommendation System

A Report for the Evaluation 3 of Project - 2

Submitted by

RAJIV PRAJAPATI

(1613101550 / 16SCSE101067)

In partial fulfillment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE ENGINEERING

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Under Supervision Of

MR. P. Rajakumar

Assistant Professor

APRIL / MAY – 2020

GALGOTIAS
UNIVERSITY

SCHOOL OF COMPUTING AND SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report "**Recommendation System"** is the bonafide work of "RAJIV PRAJAPATI (1613101550)" who carried out the project work under my supervision.

SIGNATURE OF HEAD	**SIGNATURE OF SUPERVISOR**
Dr. MUNISH SHABARWAL,	Mr. P.RAJAKUMAR
PhD (Management), PhD (CS)	
Professor & Dean,	**Asst. Prof.**
School of Computing Science & Engineering.	**School of Computing Science & Engineering.**

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. ABSTRACT

On the Internet, where the number of choices is overwhelming, there is need to filter, prioritize and efficiently deliver relevant information in order to alleviate the problem of information overload, which has created a potential problem to many Internet users. Recommender systems solve this problem by searching through large volume of dynamically generated information to provide users with personalized content and services. This paper explores the different characteristics and potentials of different prediction techniques in recommendation systems in order to serve as a compass for research and practice in the field of recommendation systems.Recommendation systems in e-commerce have become essential tools to help businesses increase their sales. Project, we detail the design of a product recommendation system for small online retailers. Our system is specifically designed to address the needs of retailers with small data pools and limited processing power, and is tested for accuracy, efficiency, and scalability on real life data from a small online retailer.

Keywords: Machine Learning, Cosine Similarity, Flask, Data Science, Python, Imdb Api, Recommender System.

2. INTRODUCTION

A product recommendation is basically a filtering system that seeks to predict and show the items that a user would like to purchase. It may not be entirely accurate, but if it shows you what you like then it is doing its job right.

A product recommendation is basically a filtering system that seeks to predict and show the items that a user would like to purchase. It may not be entirely accurate, but if it shows you what you like then it is doing its job right.

Recommender_systems have become increasingly popular in recent years, and are utilized in a variety of areas including movies, music, news, books, research articles, search queries, social tags, and products in general. Mostly used in the digital domain, majority of today's E-Commerce sites like eBay, Amazon, Alibaba etc. make use of their proprietary recommendation algorithms in order to better serve the customers with the products they are bound to like. There many other benefits too.

And if set up and configured properly, it can significantly boost revenues, CTRs, conversions, and other important metrics. Moreover, they can have positive effects on the user experience as well, which translates into metrics that are harder to measure but are nonetheless of much importance to online businesses, such as customer satisfaction and retention.

All this is only possible with a recommendations engine. Recommendation engines basically are data filtering tools that make use of algorithms and data to recommend the most relevant items to a particular user. Or in simple terms, they are nothing but an automated form of a "shop counter guy". You ask him for a product. Not only he shows that product, but also the related ones which you could buy. They are well trained in cross-selling and upselling.

With the growing amount of information on the internet and with a significant rise in the number of users, it is becoming important for companies to search, map and provide them with the relevant chunk of information according to their preferences and tastes. Chatbots_also work on the same working, but they are little smarter and learn from each product the use views or buys.

Let’s consider an example to better understand the concept of a recommendation engine. If I am not wrong, almost all of you must have used Amazon for shopping. And just so you know, 35% of Amazon.com’s revenue is generated by its recommendation engine.

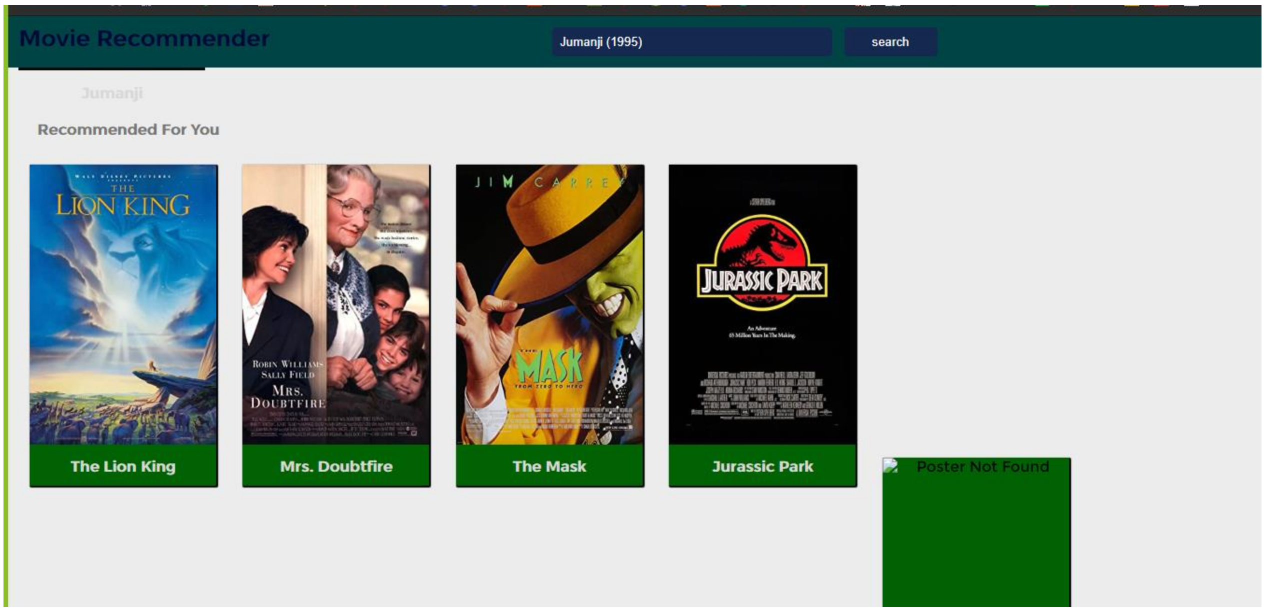


Figure 1: Recommendation System Image

A recommendation system is a type of information filtering system which attempts to predict the preferences of a user, and make suggests based on these preferences. There are a wide variety of applications for recommendation systems. These have become increasingly popular over the last few years and are now utilized in most online platforms that we use. The content of such platforms varies from movies, music, books and videos, to friends and stories on social media platforms, to products on e-commerce websites, to people on professional and dating websites, to search results returned on Google. Often, these systems are able to collect information about a users choices, and can use this information to improve their suggestions in the future. For example, Facebook can monitor your interaction with various stories on your feed in order to learn what types of stories appeal to you. Sometimes, the recommender systems can make improvements based on the activities of a large number of people. For example, if Amazon observes that a large number of customers who buy the latest Apple Macbook also buy a USB-C-toUSB Adapter, they can recommend the Adapter to a new user who has just added a Macbook to his cart.

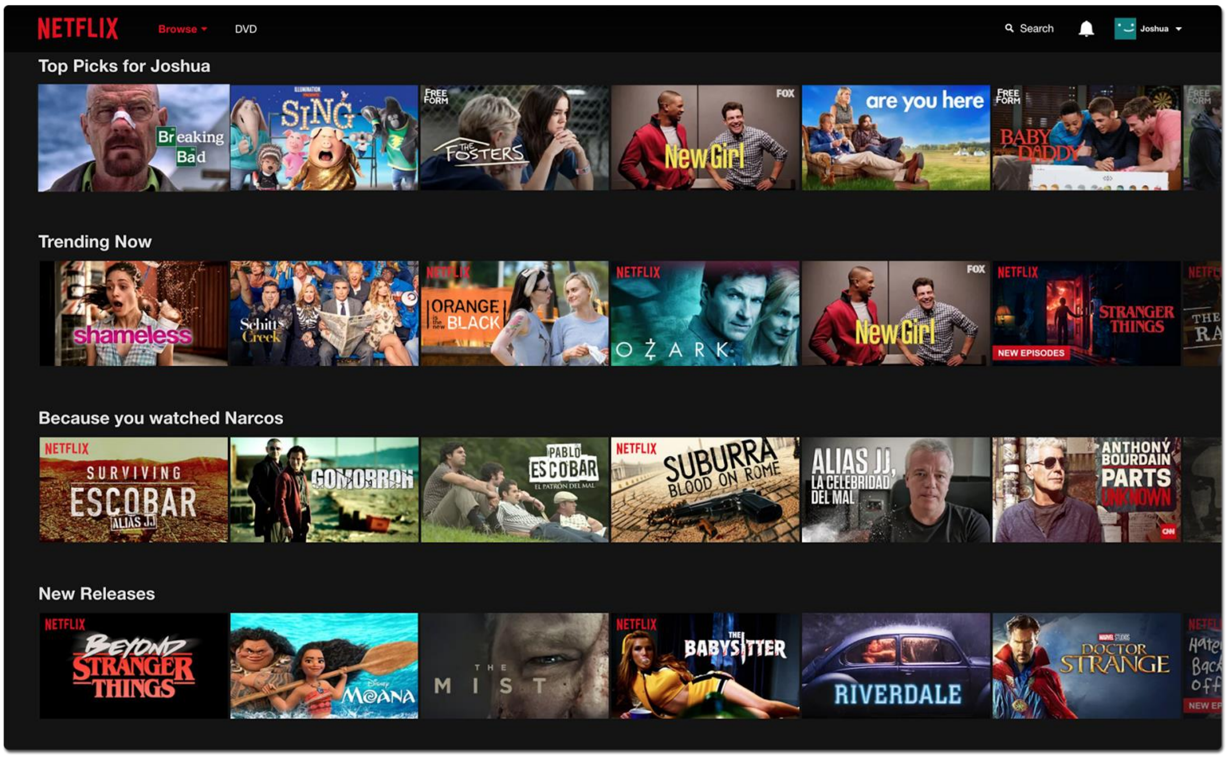


Figure -2 Netflix Movie Recommendation

3. LITERATURE SURVEY

Recommender systems have become an important research field since the emergence of the first paper on collaborative filtering in the mid-1990s. Although academic research on recommender systems has increased significantly over the past 10 years, there are deficiencies in the comprehensive literature review and classification of that research. For that reason, we reviewed 210 articles on recommender systems from 46 journals published between 2001 and 2010, and then classified those by the year of publication, the journals in which they appeared, their application fields, and their data mining techniques. The 210 articles are categorized into eight application fields (books, documents, images, movie, music, shopping, TV programs, and others) and eight data mining techniques (association rule, clustering, decision tree, k-nearest neighbor, link analysis, neural network, regression, and other heuristic methods). Our research provides information about trends in recommender systems research by examining the publication years of the articles, and provides practitioners and researchers with insight and future direction on recommender systems. We hope that this paper helps anyone who is interested in recommender systems research with insight for future research direction.

4. PROPOSED METHOD

In this Project (Recommender System) I am using content based filtering. There are different categories of recommender system . But Recommender System is mainly divided into two Categories .

Content Based Filtering

It is a technique where individual user profiles are taken into account. It enhances the user's interest and predicts whether the user would be interested in eating at any particular restaurant or interested in seeing any particular movie [22]. It is also known as adaptive Filtering as it provides suggestions according to user's field of interest and adapts user's likes and dislikes. It represents the comparison between the content contained in the item with the content of items of user's interest. By using Bayesian hierarchical model, better user profiles for upcoming users is made by collecting feedbacks from the old users [20]. Content based collaborative filtering is more widely used to compare pure CF and pure Content-base. In CF the problem of sparsity is overcome (converting sparse user filled matrix into full user rating matrix) by using content-based prediction [21]. Fig.2 displays the flow of information in a content based recommendation system. Relevant entities of an item and relations are kept together as input. Main features of items are extracted from item ontology. Features of items, user's ratings and user modeling data Content based [9] systems focus on the features of the products and aim at creating a user profile depending on the previous reviews and also a profile of the item in accordance with the features it provides and the reviews it has received [8], [5].It is observed that reviews usually contain product feature and user opinion in pairs [5], [9], [10]. It is observed that users' reviews contain a feature of the product followed by his/her opinion about the product. Content based recommendation systems help overcome sparsity problem that is faced in collaborative filtering based recommendation system.

In This project we are basically creating a website where a user searches a movie and it get that movie with some Recommended Movie (in case of our project it displays 6 movies out of which one is searched movie and remaining 5 are Recommended Movies). In this Project we are using flask for backend purpose and HTML, CSS, Js for Designing and making dynamic page.

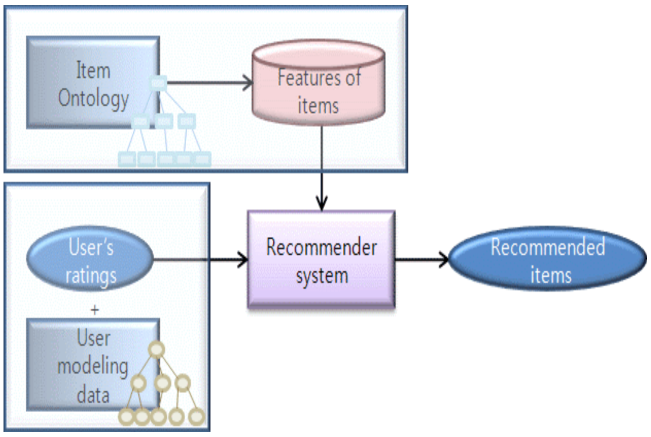


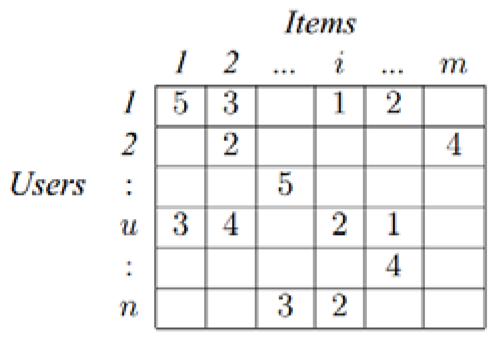
Figure 3 Content Based Filtering

Table 1. Content-Based Recommender System

Method	**Description**	**References**
Content-Boosted Collaborative Filtering	It gives an approach to combine content and collaboration to enhance existing user data and to give better performance than a pure content based predictor.	Prem Melville, Raymond J. Mooney, RamadassNagarajan[21]
FAB Technique	An adaptive recommendation service for collection and selection of web pages. It makes the system more personalized and combines the benefits of content analysis with the shared user interests.	Marko Balabanovic [24]
Bayesian hierarchical model(BHM)	Proposes a faster technique to gather a huge number of individual user profiles even if feedbacks available are less. It uses various parameters of BHM for optimization of joint data likelihood.	Yi Zhang , Jonathan Koren [20]
Keyword Map Algorithm	It captures the relations between conditions that the learner has been exposed to, which are used to represent the knowledge of the learner. The relevance and complement of learning resources recommendation are increased by keyword map. It is better suited for e-learning settings and achieves higher accuracy than common recommender methods.	Xin Wan, Neil Rubens,Toshio Okamoto and Yan Feng [23]

Table 1 shows different content based methods using sentimental analysis for recommendation system. These methods are based on a variety of different models. Different researchers have studied the relevance of various techniques that can be implemented in content based recommendation systems. Accuracy and relevance of the recommendation systems have become better by extensive research. Content-Boosted Collaborative filtering technique was implemented by Prem Melville et. Al. Recommendation system using content based technique and Bayesian Hierarchical Model (BHM) was built by Marko Balabanovic.

In Case of our model (Content Based Filtering) We are using the data set from movie lens in this data there are user and movie (item) and we created user item matrix and on the basis of that we are going to find cosine similarity of each vector and on the basis of cosine similarity we are going to decide whether the given data is matching (like to recommend) to given user or not.



		Items					
		1	*2*	...	*i*	...	*m*
	1	5	3		1	2	
	2		2				4
Users	:			5			
	u	3	4		2	1	
	:					4	
	n			3	2		

Figure 4 user item interaction matrix

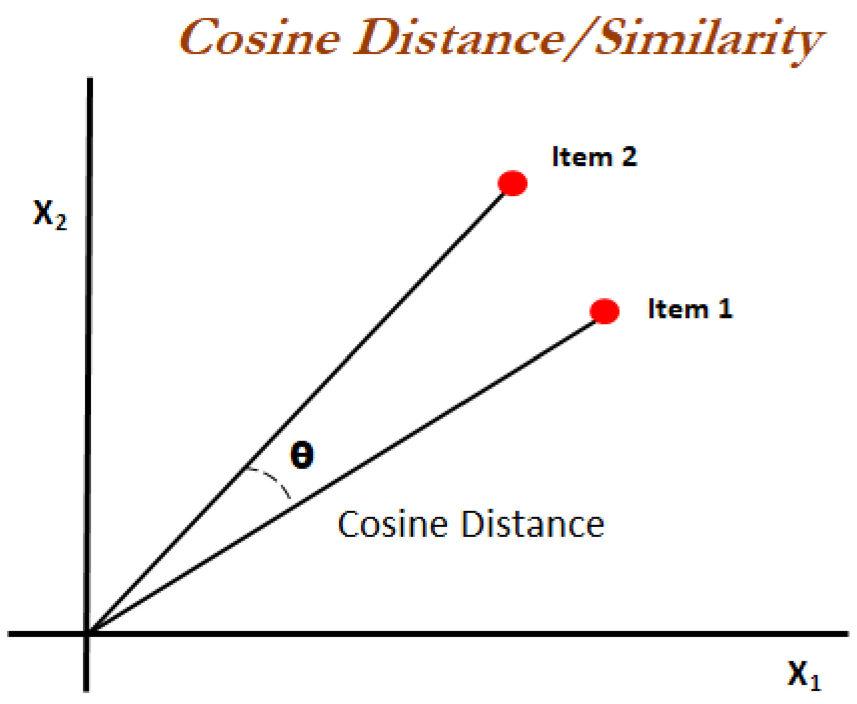


Figure 5 Cosine Similarity Between two items

As we will be working on this concept, it would be nice to reiterate the basics. Cosine similarity is a measure of similarity between two nonzero vectors of an inner product space that measures the cosine of the angle between them. Cosine of *0^0* is *1* and it is less than *1* for any other angle:

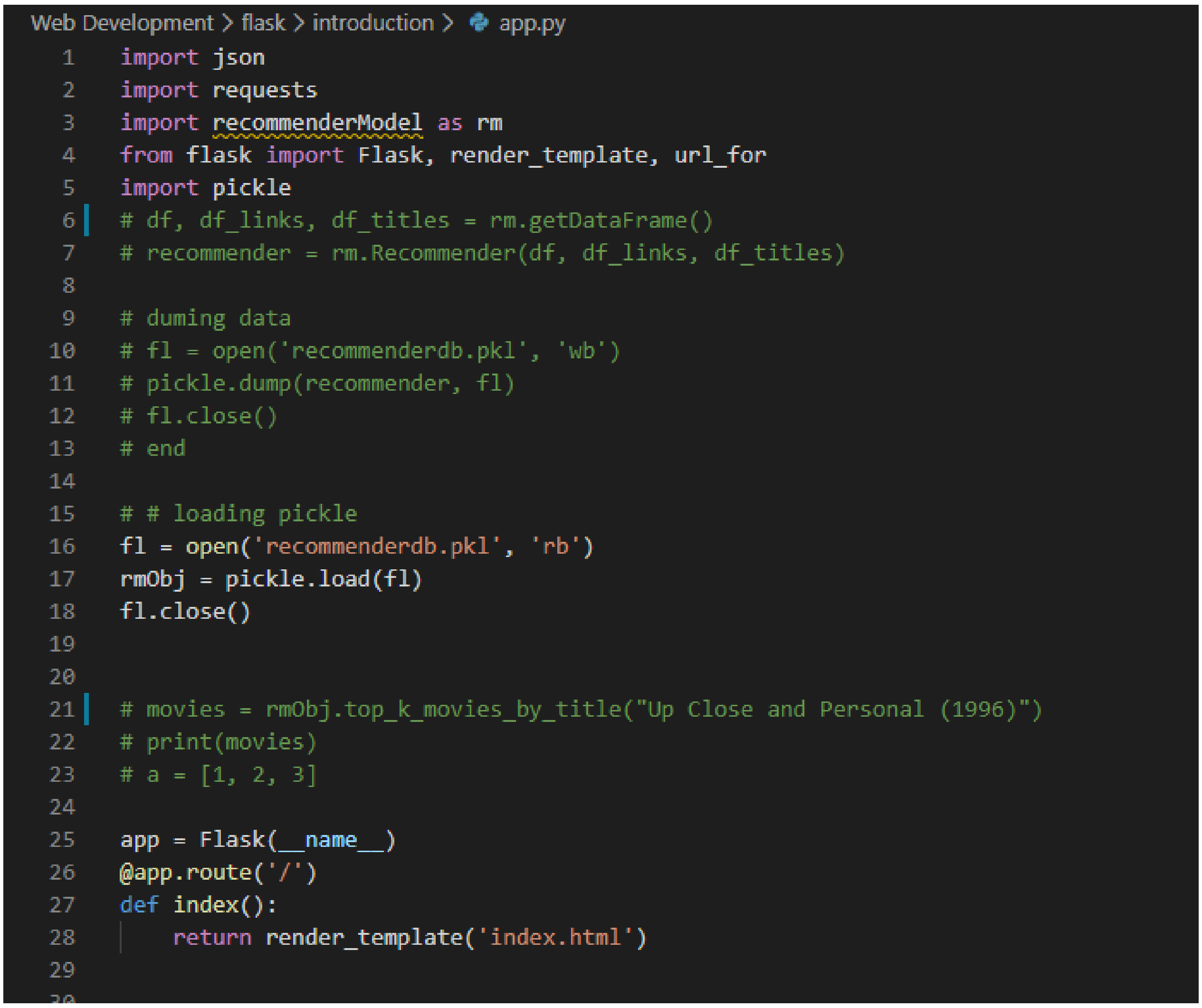
$$\cos(\theta) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Here, A_i and B_i are components of vector *A* and *B* respectively:

4.1 IMPLEMENTATION

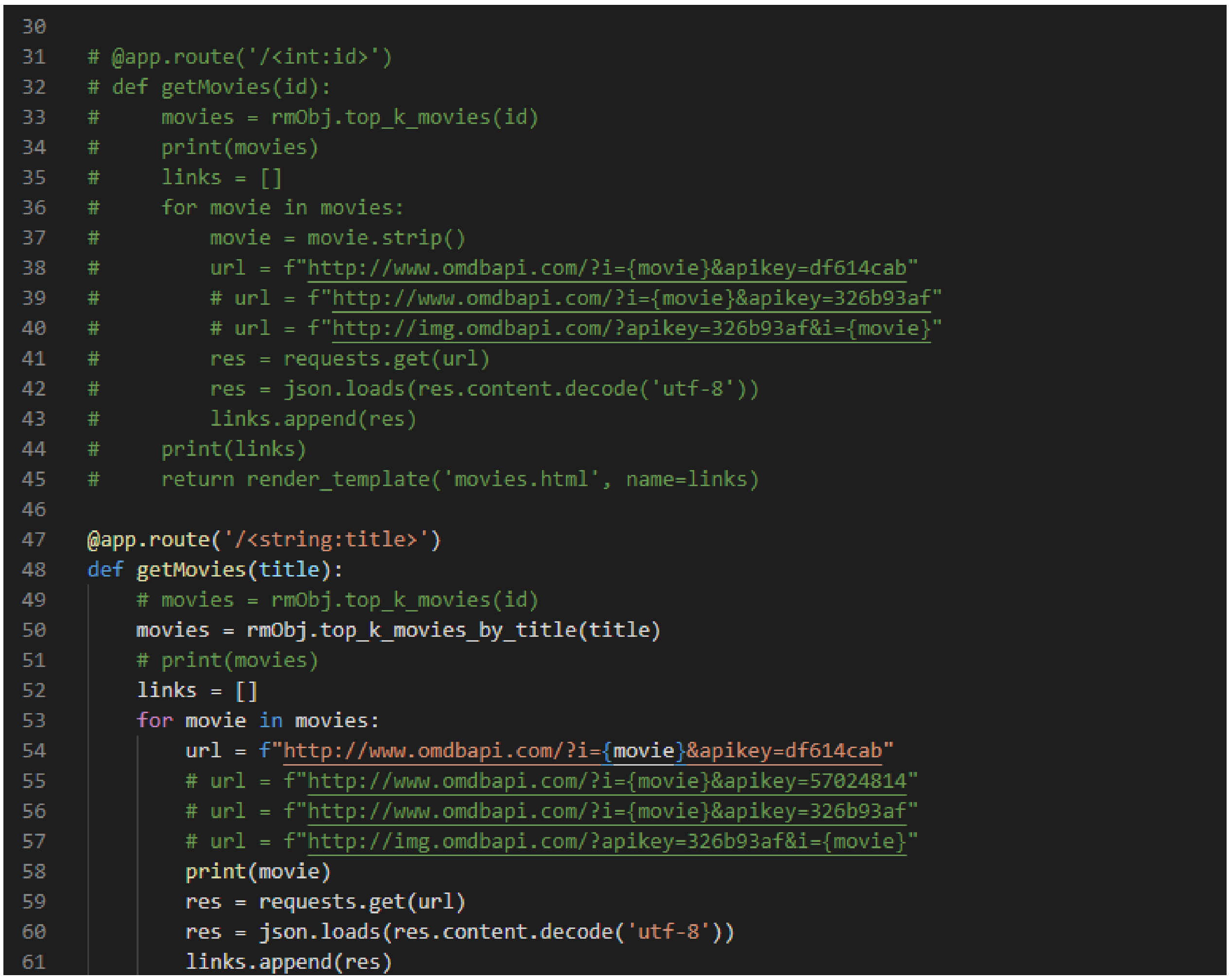
Technology used: Flask, Python, Html, CSS, Java Script, Pickle

App.py

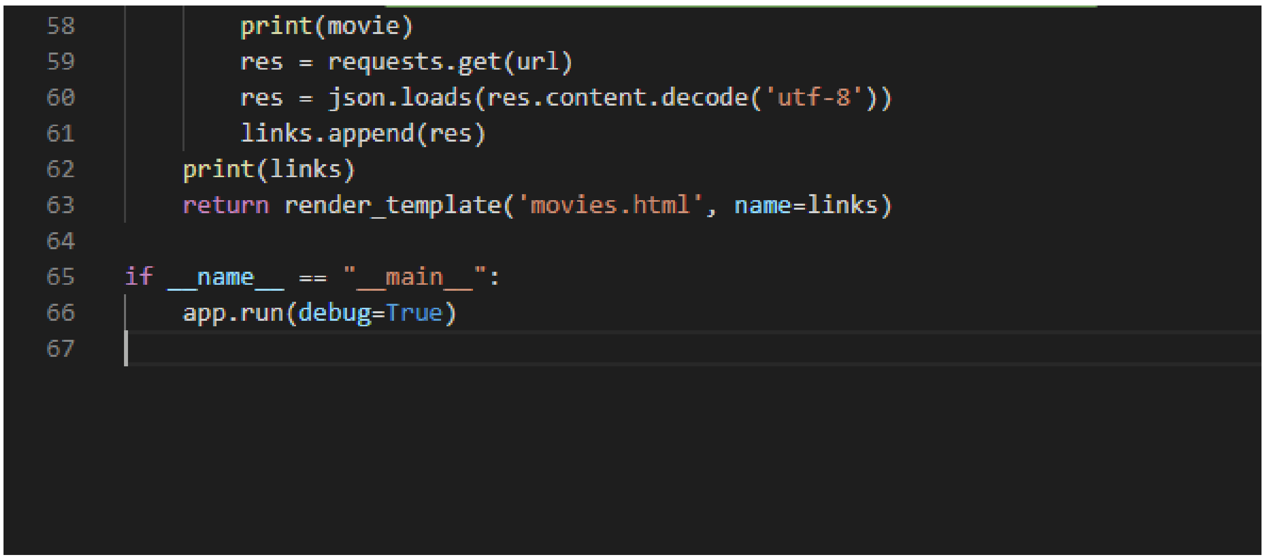


Web Development > flask > introduction > app.py

```python
import json
import requests
import recommenderModel as rm
from flask import Flask, render_template, url_for
import pickle
# df, df_links, df_titles = rm.getDataFrame()
# recommender = rm.Recommender(df, df_links, df_titles)

# duming data
# fl = open('recommenderdb.pkl', 'wb')
# pickle.dump(recommender, fl)
# fl.close()
# end

# # loading pickle
fl = open('recommenderdb.pkl', 'rb')
rmObj = pickle.load(fl)
fl.close()


# movies = rmObj.top_k_movies_by_title("Up Close and Personal (1996)")
# print(movies)
# a = [1, 2, 3]

app = Flask(__name__)
@app.route('/')
def index():
    return render_template('index.html')

```



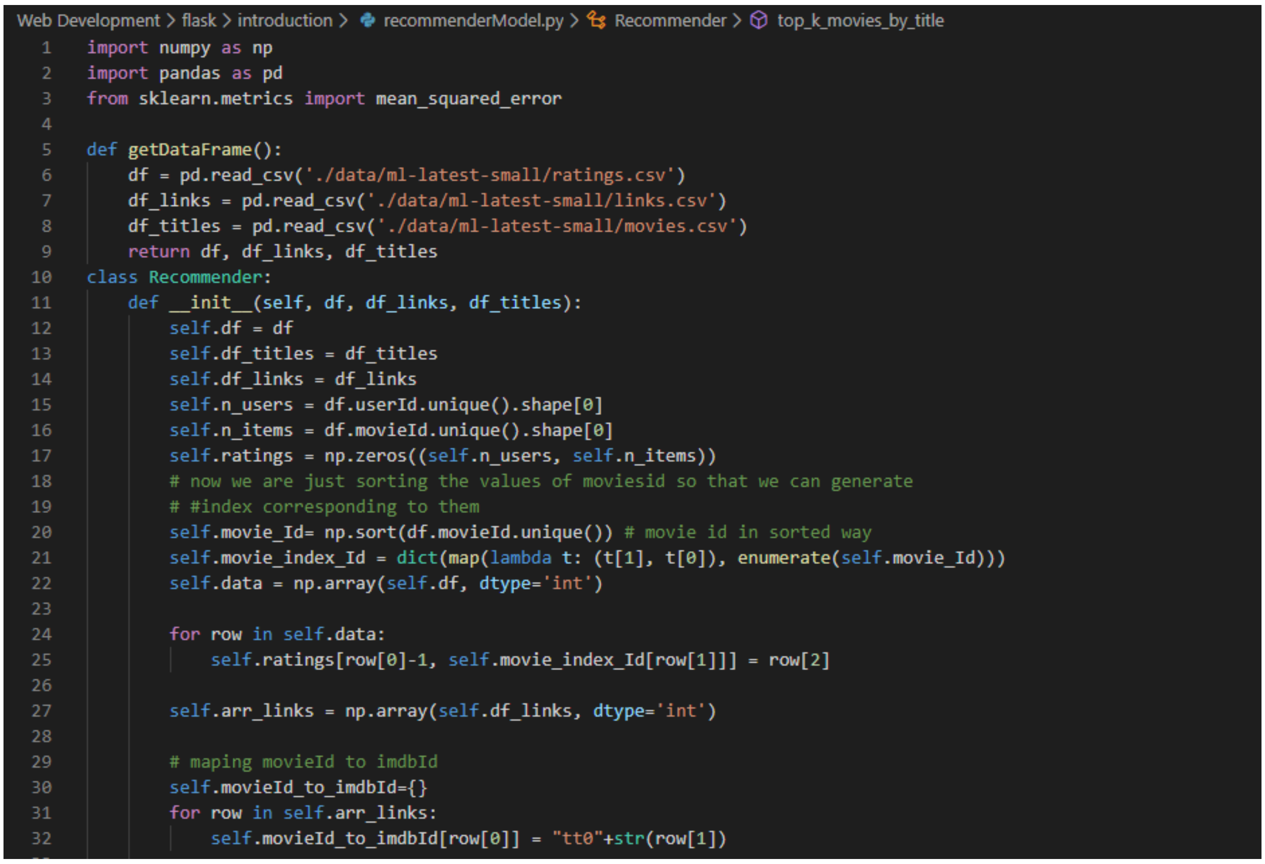
```python

# @app.route('/<int:id>')
# def getMovies(id):
#     movies = rmObj.top_k_movies(id)
#     print(movies)
#     links = []
#     for movie in movies:
#         movie = movie.strip()
#         url = f"http://www.omdbapi.com/?i={movie}&apikey=df614cab"
#         # url = f"http://www.omdbapi.com/?i={movie}&apikey=326b93af"
#         # url = f"http://img.omdbapi.com/?apikey=326b93af&i={movie}"
#         res = requests.get(url)
#         res = json.loads(res.content.decode('utf-8'))
#         links.append(res)
#     print(links)
#     return render_template('movies.html', name=links)

@app.route('/<string:title>')
def getMovies(title):
    # movies = rmObj.top_k_movies(id)
    movies = rmObj.top_k_movies_by_title(title)
    # print(movies)
    links = []
    for movie in movies:
        url = f"http://www.omdbapi.com/?i={movie}&apikey=df614cab"
        # url = f"http://www.omdbapi.com/?i={movie}&apikey=57024814"
        # url = f"http://www.omdbapi.com/?i={movie}&apikey=326b93af"
        # url = f"http://img.omdbapi.com/?apikey=326b93af&i={movie}"
        print(movie)
        res = requests.get(url)
        res = json.loads(res.content.decode('utf-8'))
        links.append(res)
```



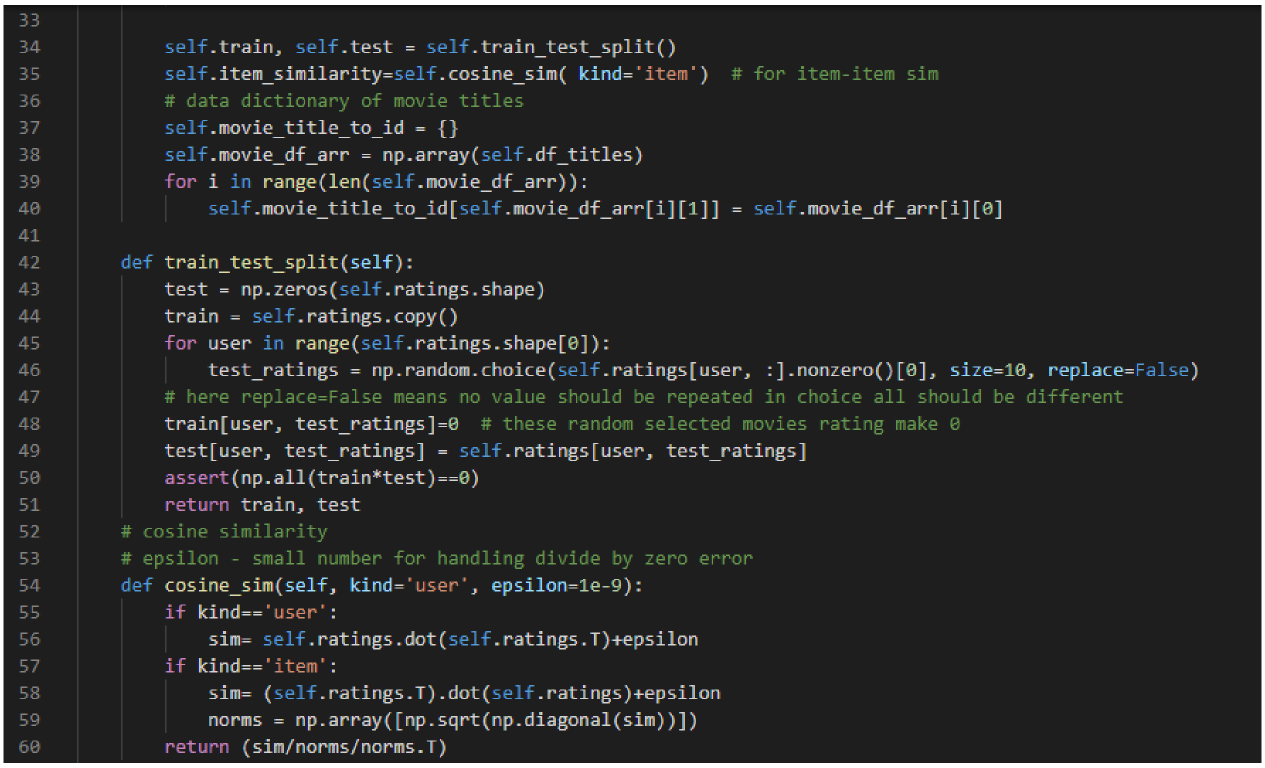
```python
        print(movie)
        res = requests.get(url)
        res = json.loads(res.content.decode('utf-8'))
        links.append(res)
```

```python
    print(links)
    return render_template('movies.html', name=links)

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

```

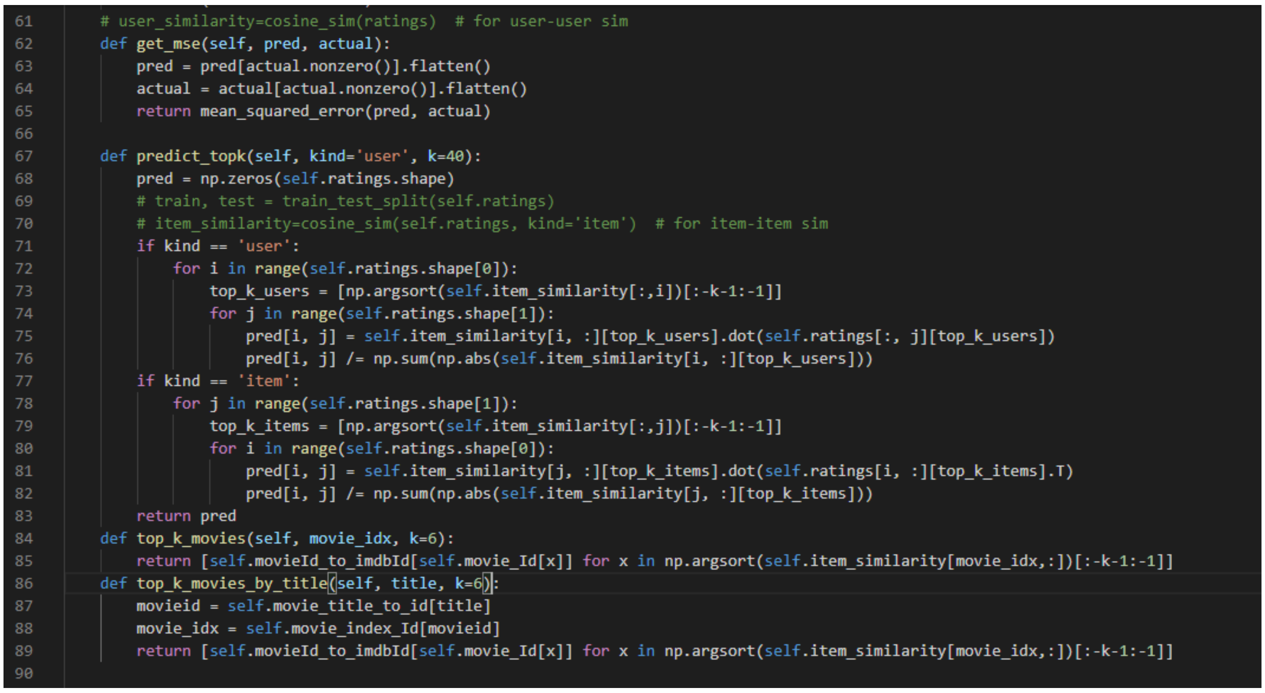
RecommenderModel.py   --file name


Web Development > flask > introduction > recommenderModel.py > Recommender > top_k_movies_by_title

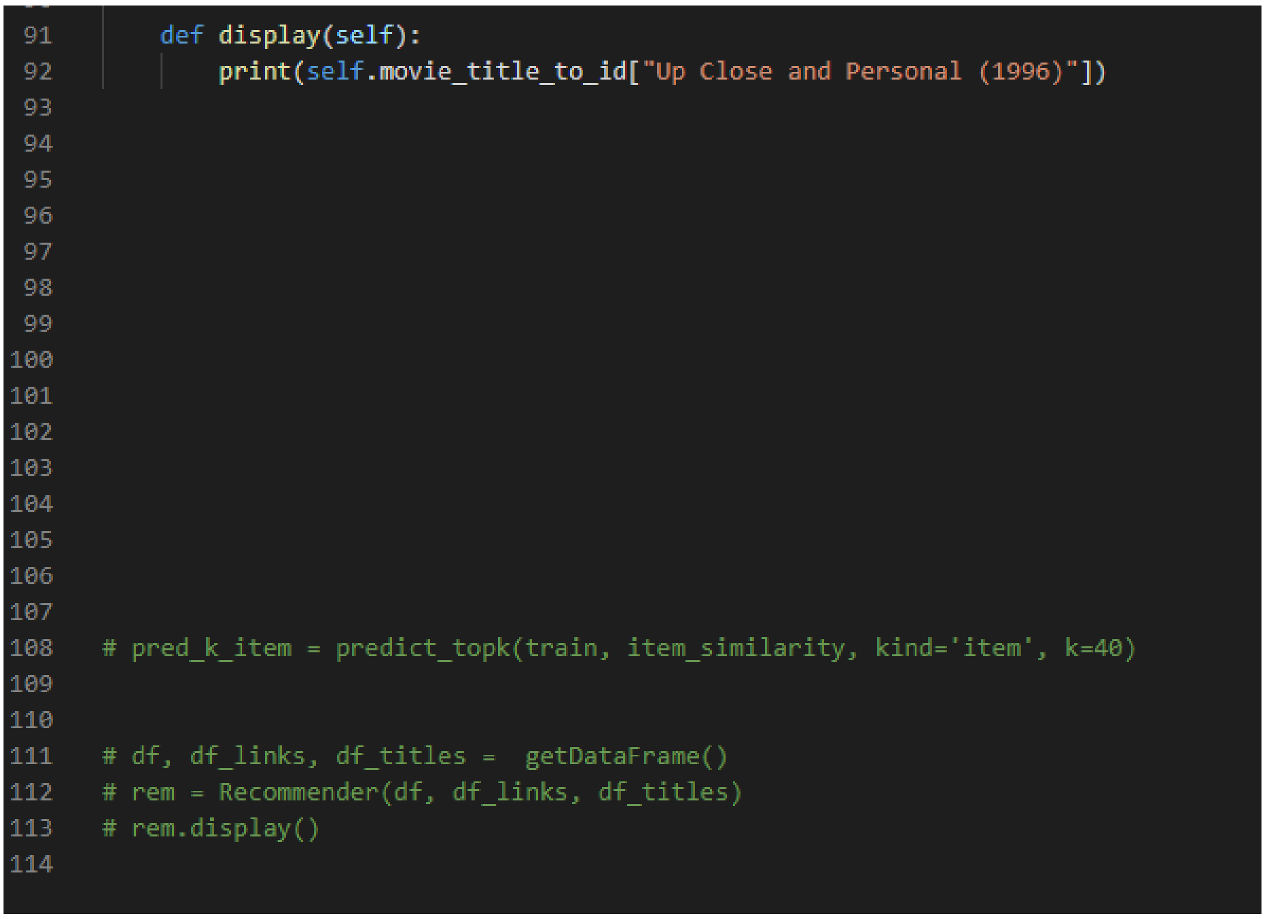
```python
import numpy as np
import pandas as pd
from sklearn.metrics import mean_squared_error

def getDataFrame():
    df = pd.read_csv('./data/ml-latest-small/ratings.csv')
    df_links = pd.read_csv('./data/ml-latest-small/links.csv')
    df_titles = pd.read_csv('./data/ml-latest-small/movies.csv')
    return df, df_links, df_titles
class Recommender:
    def __init__(self, df, df_links, df_titles):
        self.df = df
        self.df_titles = df_titles
        self.df_links = df_links
        self.n_users = df.userId.unique().shape[0]
        self.n_items = df.movieId.unique().shape[0]
        self.ratings = np.zeros((self.n_users, self.n_items))
        # now we are just sorting the values of moviesid so that we can generate
        # #index corresponding to them
        self.movie_Id= np.sort(df.movieId.unique()) # movie id in sorted way
        self.movie_index_Id = dict(map(lambda t: (t[1], t[0]), enumerate(self.movie_Id)))
        self.data = np.array(self.df, dtype='int')

        for row in self.data:
            self.ratings[row[0]-1, self.movie_index_Id[row[1]]] = row[2]

        self.arr_links = np.array(self.df_links, dtype='int')

        # maping movieId to imdbId
        self.movieId_to_imdbId={}
        for row in self.arr_links:
            self.movieId_to_imdbId[row[0]] = "tt0"+str(row[1])
```



```python

        self.train, self.test = self.train_test_split()
        self.item_similarity=self.cosine_sim( kind='item')  # for item-item sim
        # data dictionary of movie titles
        self.movie_title_to_id = {}
        self.movie_df_arr = np.array(self.df_titles)
        for i in range(len(self.movie_df_arr)):
            self.movie_title_to_id[self.movie_df_arr[i][1]] = self.movie_df_arr[i][0]

    def train_test_split(self):
        test = np.zeros(self.ratings.shape)
        train = self.ratings.copy()
        for user in range(self.ratings.shape[0]):
            test_ratings = np.random.choice(self.ratings[user, :].nonzero()[0], size=10, replace=False)
        # here replace=False means no value should be repeated in choice all should be different
        train[user, test_ratings]=0  # these random selected movies rating make 0
        test[user, test_ratings] = self.ratings[user, test_ratings]
        assert(np.all(train*test)==0)
        return train, test
    # cosine similarity
    # epsilon - small number for handling divide by zero error
    def cosine_sim(self, kind='user', epsilon=1e-9):
        if kind=='user':
            sim= self.ratings.dot(self.ratings.T)+epsilon
        if kind=='item':
            sim= (self.ratings.T).dot(self.ratings)+epsilon
            norms = np.array([np.sqrt(np.diagonal(sim))])
        return (sim/norms/norms.T)
```

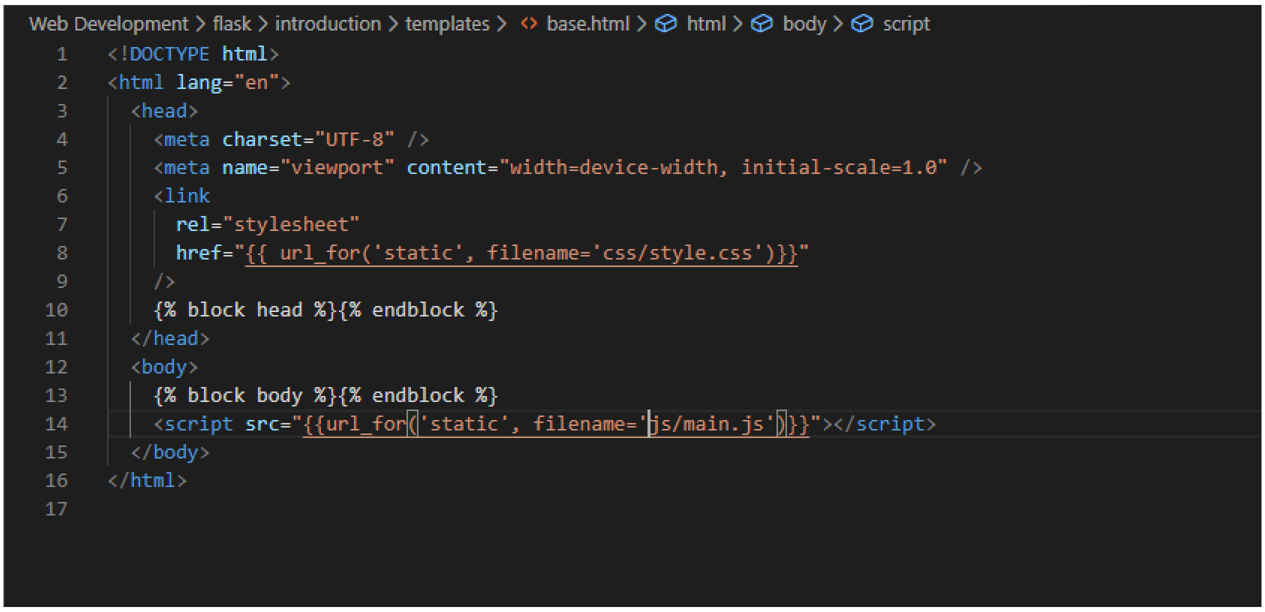


```python
    # user_similarity=cosine_sim(ratings)  # for user-user sim
    def get_mse(self, pred, actual):
        pred = pred[actual.nonzero()].flatten()
        actual = actual[actual.nonzero()].flatten()
        return mean_squared_error(pred, actual)

    def predict_topk(self, kind='user', k=40):
        pred = np.zeros(self.ratings.shape)
        # train, test = train_test_split(self.ratings)
        # item_similarity=cosine_sim(self.ratings, kind='item')  # for item-item sim
        if kind == 'user':
            for i in range(self.ratings.shape[0]):
                top_k_users = [np.argsort(self.item_similarity[:,i])[:-k-1:-1]]
                for j in range(self.ratings.shape[1]):
                    pred[i, j] = self.item_similarity[i, :][top_k_users].dot(self.ratings[:, j][top_k_users])
                    pred[i, j] /= np.sum(np.abs(self.item_similarity[i, :][top_k_users]))
        if kind == 'item':
            for j in range(self.ratings.shape[1]):
                top_k_items = [np.argsort(self.item_similarity[:,j])[:-k-1:-1]]
                for i in range(self.ratings.shape[0]):
                    pred[i, j] = self.item_similarity[j, :][top_k_items].dot(self.ratings[i, :][top_k_items].T)
                    pred[i, j] /= np.sum(np.abs(self.item_similarity[j, :][top_k_items]))
        return pred
    def top_k_movies(self, movie_idx, k=6):
        return [self.movieId_to_imdbId[self.movie_Id[x]] for x in np.argsort(self.item_similarity[movie_idx,:])[:-k-1:-1]]
    def top_k_movies_by_title(self, title, k=6):
        movieid = self.movie_title_to_id[title]
        movie_idx = self.movie_index_Id[movieid]
        return [self.movieId_to_imdbId[self.movie_Id[x]] for x in np.argsort(self.item_similarity[movie_idx,:])[:-k-1:-1]]

```



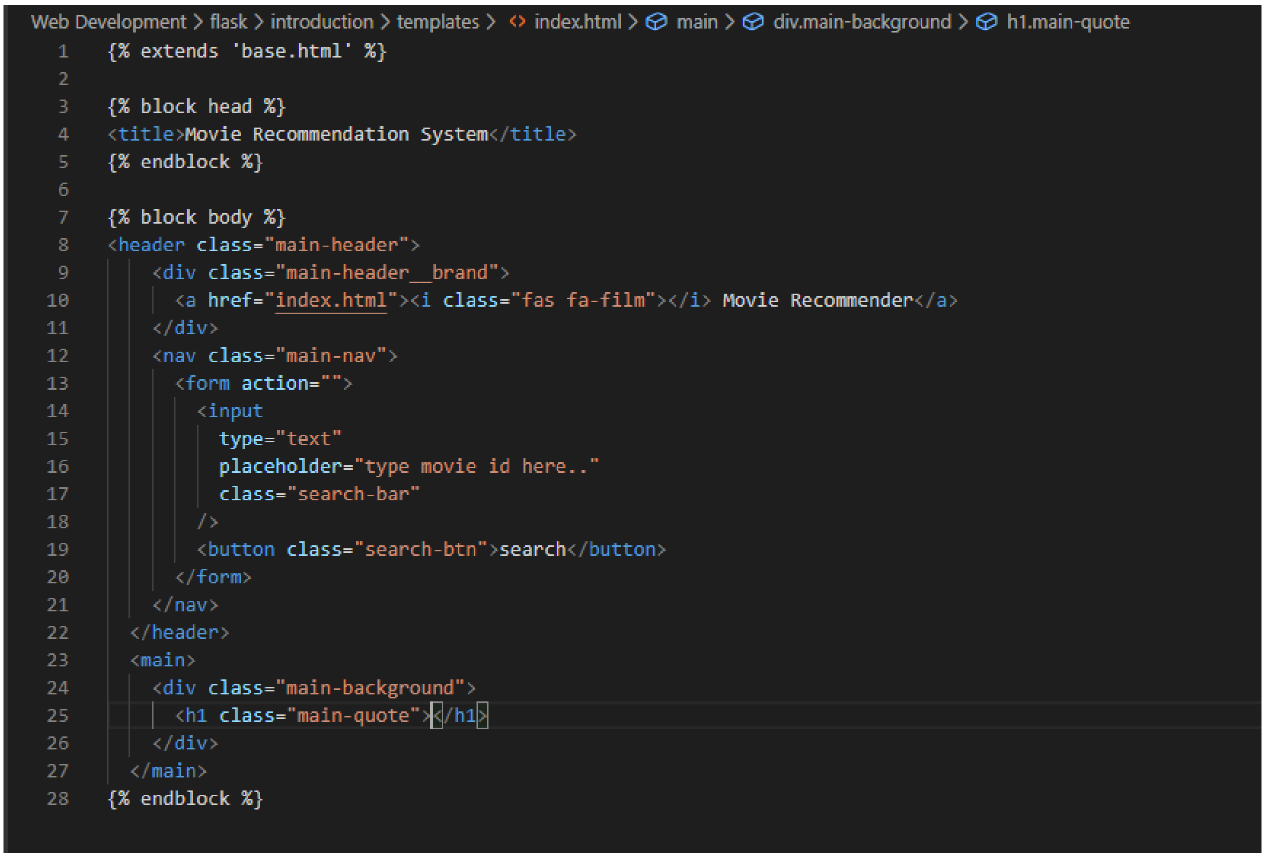
```python
    def display(self):
        print(self.movie_title_to_id["Up Close and Personal (1996)"])














# pred_k_item = predict_topk(train, item_similarity, kind='item', k=40)


# df, df_links, df_titles =  getDataFrame()
# rem = Recommender(df, df_links, df_titles)
# rem.display()

```

Base.html -- file name


Web Development > flask > introduction > templates > base.html > html > body > script

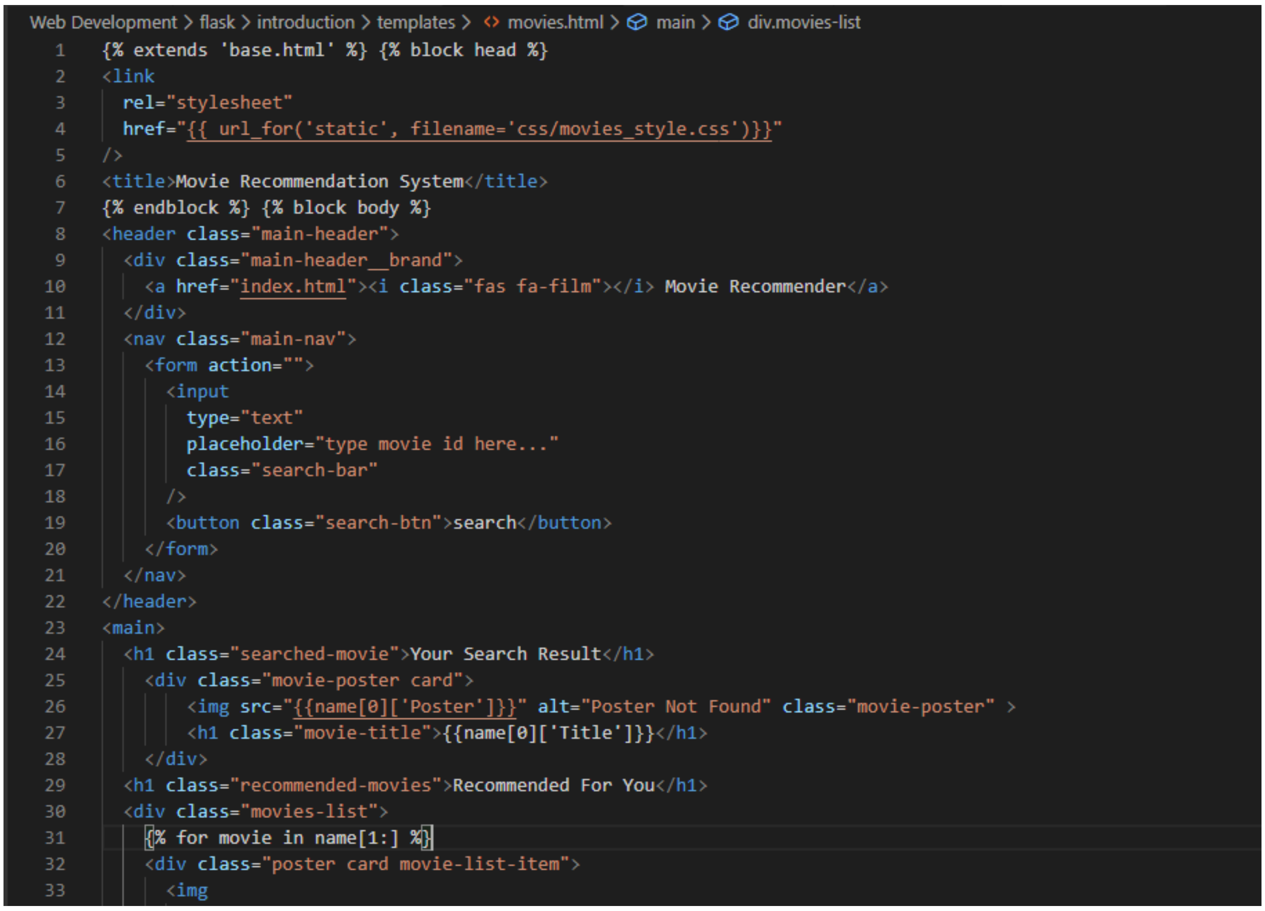
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link
      rel="stylesheet"
      href="{{ url_for('static', filename='css/style.css')}}"
    />
    {% block head %}{% endblock %}
  </head>
  <body>
    {% block body %}{% endblock %}
    <script src="{{url_for('static', filename='js/main.js')}}"></script>
  </body>
</html>
```

Index.html –filename

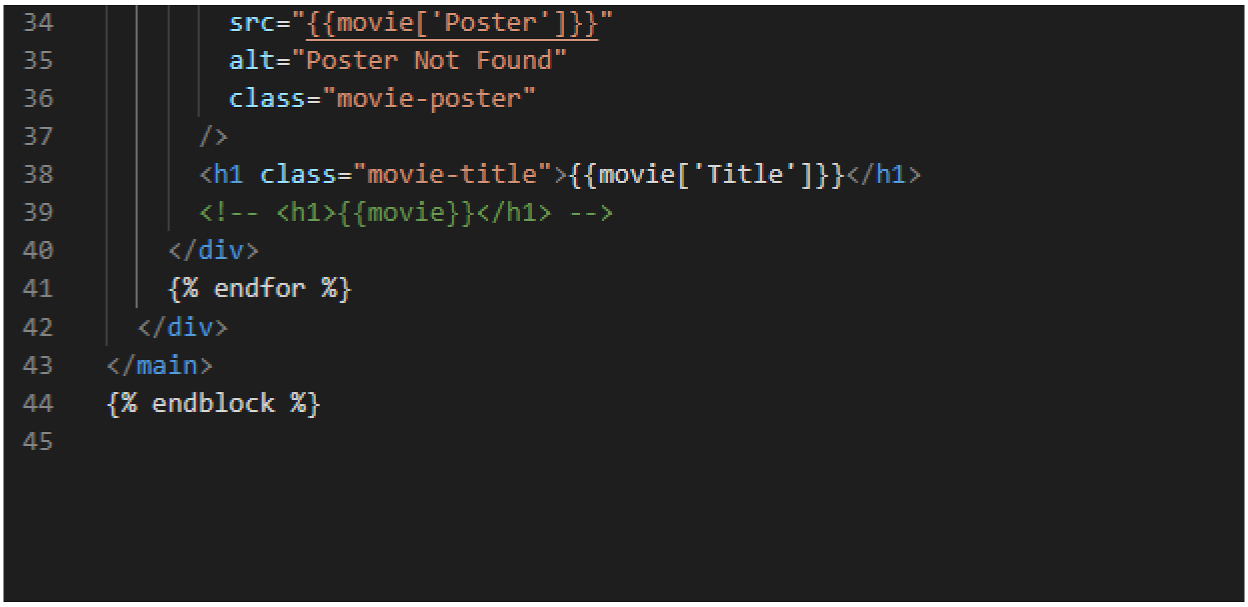

Web Development > flask > introduction > templates > index.html > main > div.main-background > h1.main-quote

```
{% extends 'base.html' %}

{% block head %}
<title>Movie Recommendation System</title>
{% endblock %}

{% block body %}
<header class="main-header">
    <div class="main-header__brand">
      <a href="index.html"><i class="fas fa-film"></i> Movie Recommender</a>
    </div>
    <nav class="main-nav">
      <form action="">
        <input
          type="text"
          placeholder="type movie id here.."
          class="search-bar"
        />
        <button class="search-btn">search</button>
      </form>
    </nav>
  </header>
  <main>
    <div class="main-background">
      <h1 class="main-quote"></h1>
    </div>
  </main>
{% endblock %}
```

Movies.html --filename

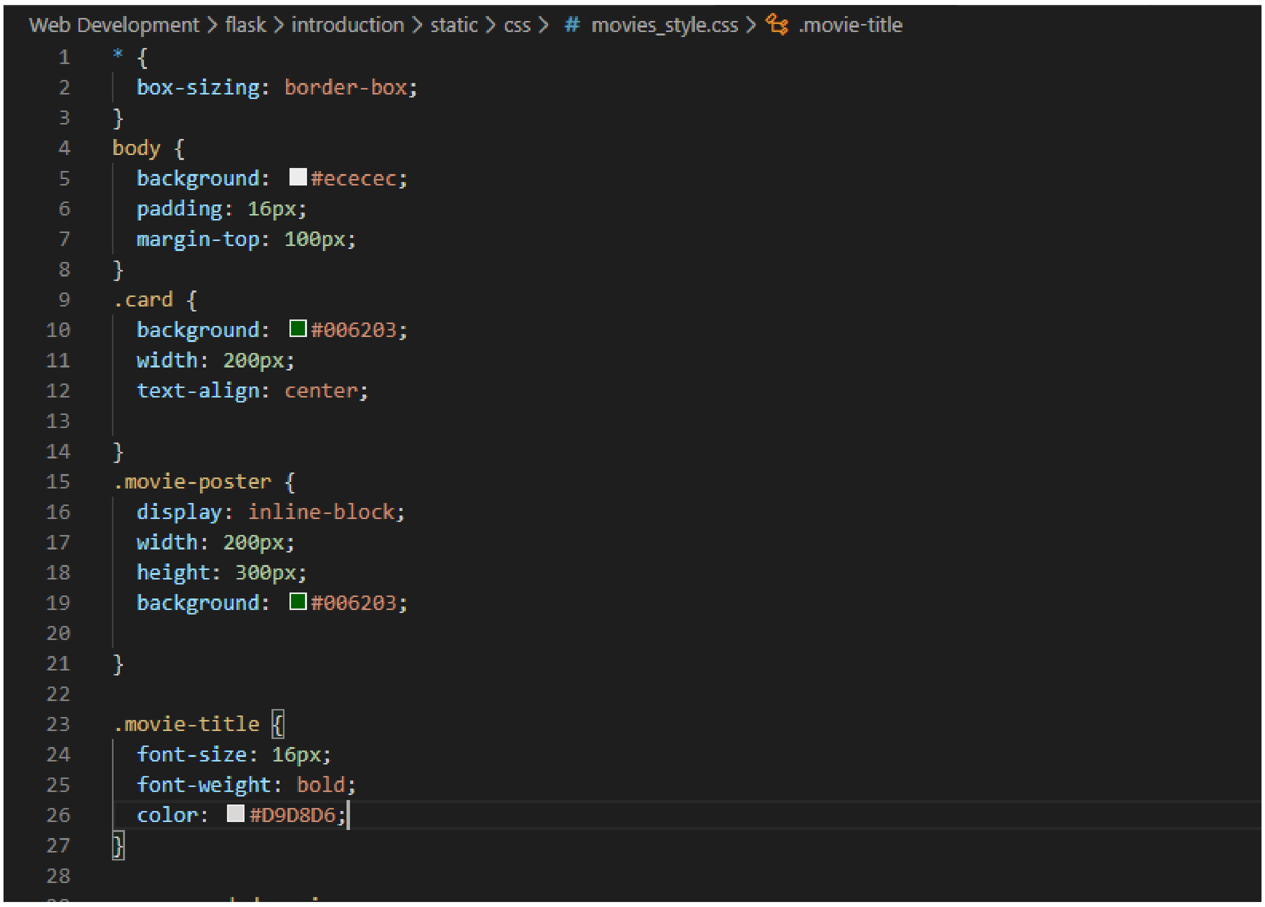

Web Development > flask > introduction > templates > movies.html > main > div.movies-list

```
{% extends 'base.html' %} {% block head %}
<link
  rel="stylesheet"
  href="{{ url_for('static', filename='css/movies_style.css')}}"
/>
<title>Movie Recommendation System</title>
{% endblock %} {% block body %}
<header class="main-header">
  <div class="main-header__brand">
    <a href="index.html"><i class="fas fa-film"></i> Movie Recommender</a>
  </div>
  <nav class="main-nav">
    <form action="">
      <input
        type="text"
        placeholder="type movie id here..."
        class="search-bar"
      />
      <button class="search-btn">search</button>
    </form>
  </nav>
</header>
<main>
  <h1 class="searched-movie">Your Search Result</h1>
    <div class="movie-poster card">
      <img src="{{name[0]['Poster']}}" alt="Poster Not Found" class="movie-poster" >
      <h1 class="movie-title">{{name[0]['Title']}}</h1>
    </div>
  <h1 class="recommended-movies">Recommended For You</h1>
  <div class="movies-list">
    {% for movie in name[1:] %}
    <div class="poster card movie-list-item">
      <img
```



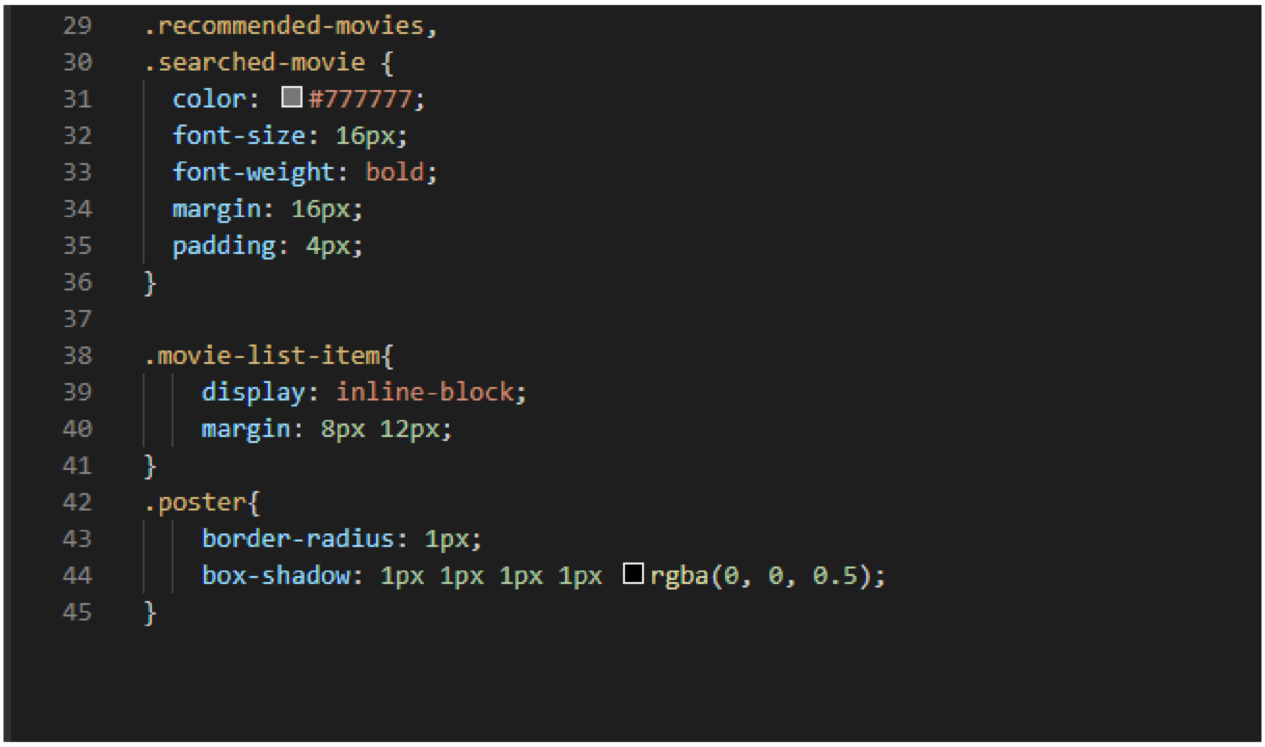
```
        src="{{movie['Poster']}}"
        alt="Poster Not Found"
        class="movie-poster"
      />
      <h1 class="movie-title">{{movie['Title']}}</h1>
      <!-- <h1>{{movie}}</h1> -->
    </div>
    {% endfor %}
  </div>
</main>
{% endblock %}
```

Movie-style.css



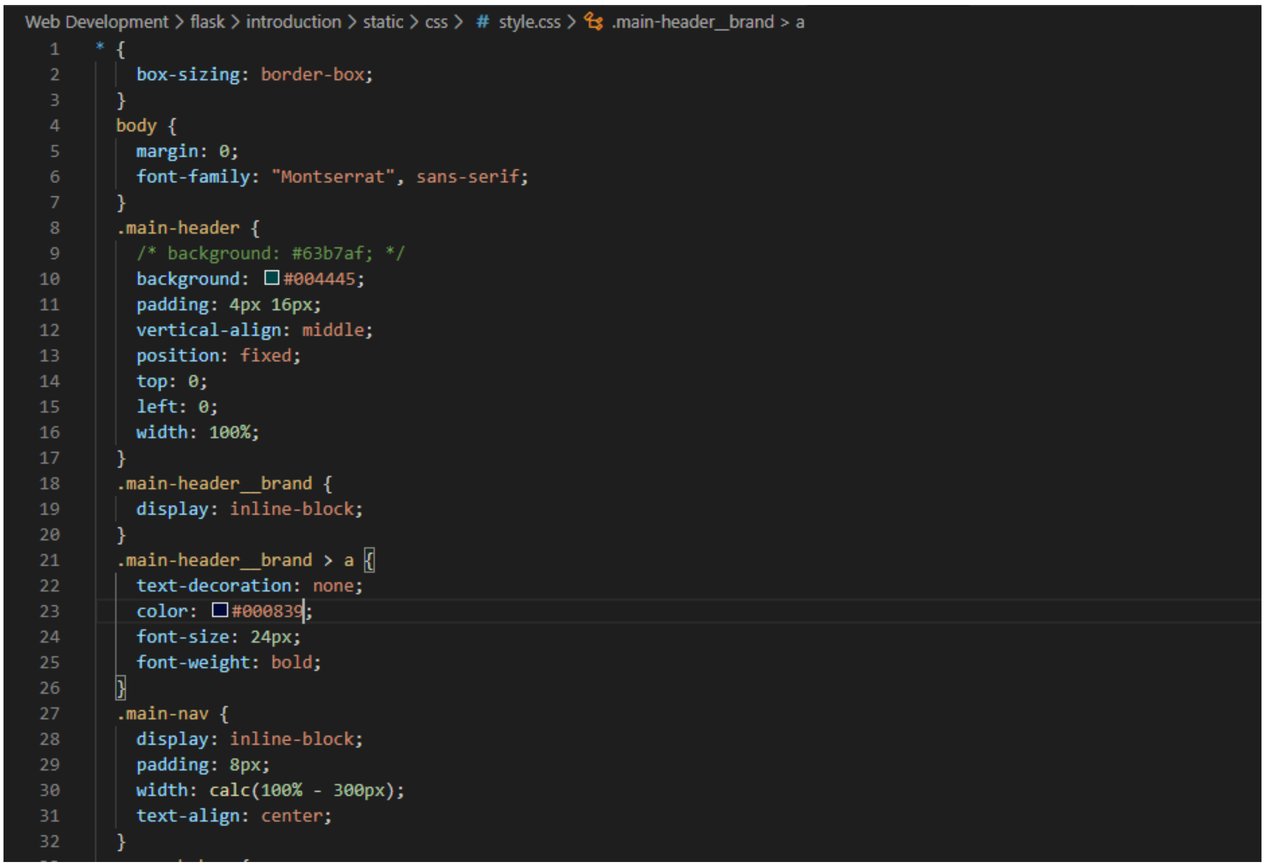
Web Development > flask > introduction > static > css > # movies_style.css > .movie-title

```css
* {
  box-sizing: border-box;
}
body {
  background: #ececec;
  padding: 16px;
  margin-top: 100px;
}
.card {
  background: #006203;
  width: 200px;
  text-align: center;

}
.movie-poster {
  display: inline-block;
  width: 200px;
  height: 300px;
  background: #006203;

}

.movie-title {
  font-size: 16px;
  font-weight: bold;
  color: #D9D8D6;
}

```

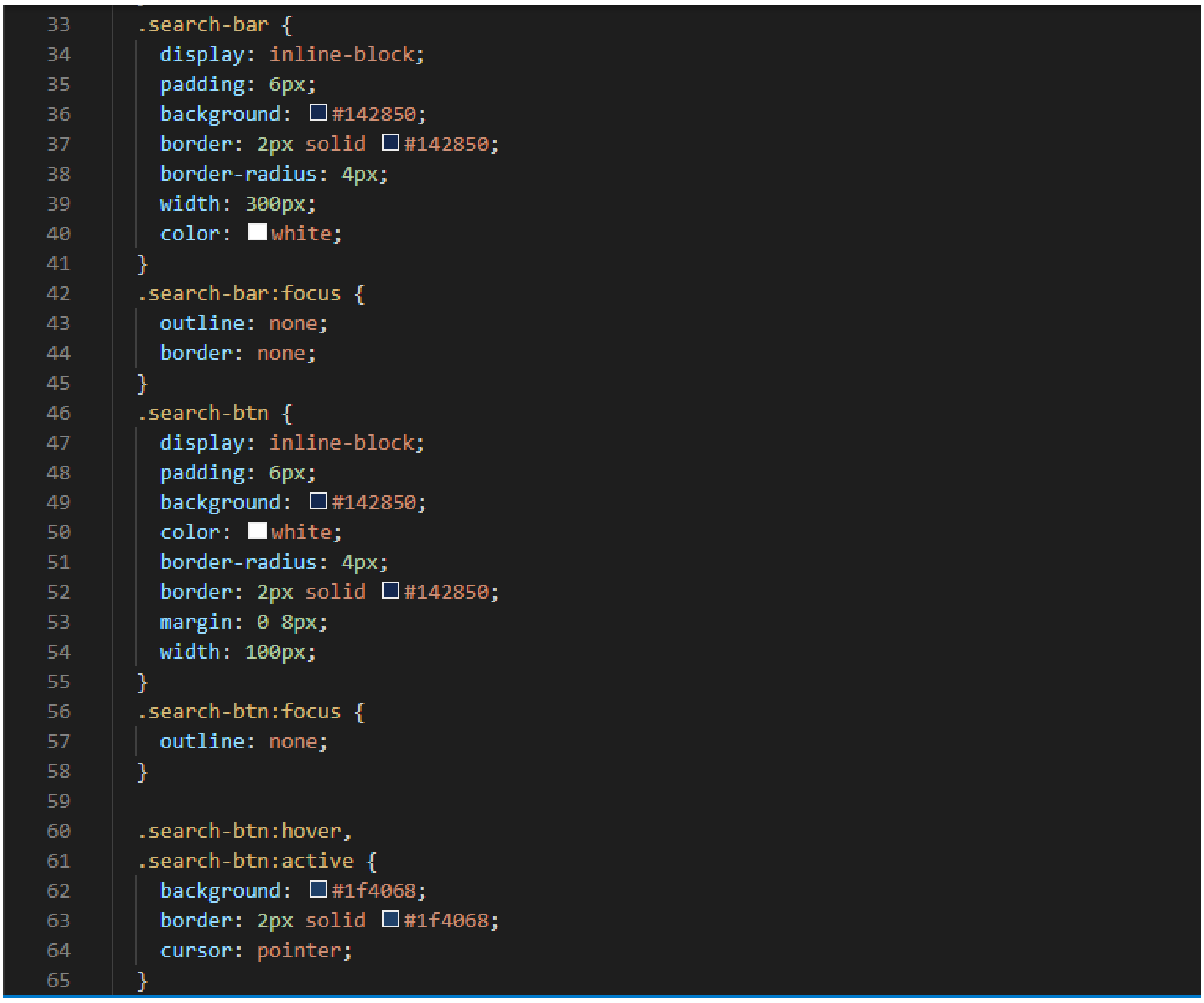


```css
.recommended-movies,
.searched-movie {
  color: #777777;
  font-size: 16px;
  font-weight: bold;
  margin: 16px;
  padding: 4px;
}

.movie-list-item{
    display: inline-block;
    margin: 8px 12px;
}
.poster{
    border-radius: 1px;
    box-shadow: 1px 1px 1px 1px rgba(0, 0, 0.5);
}
```

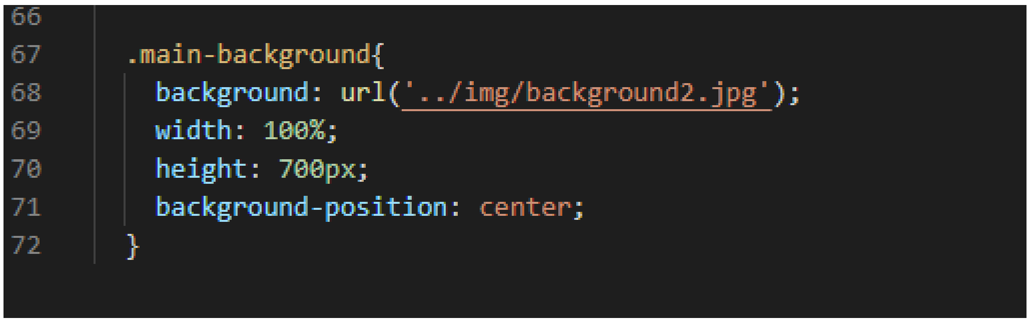
Style.css –filename


Web Development > flask > introduction > static > css > # style.css > .main-header__brand > a

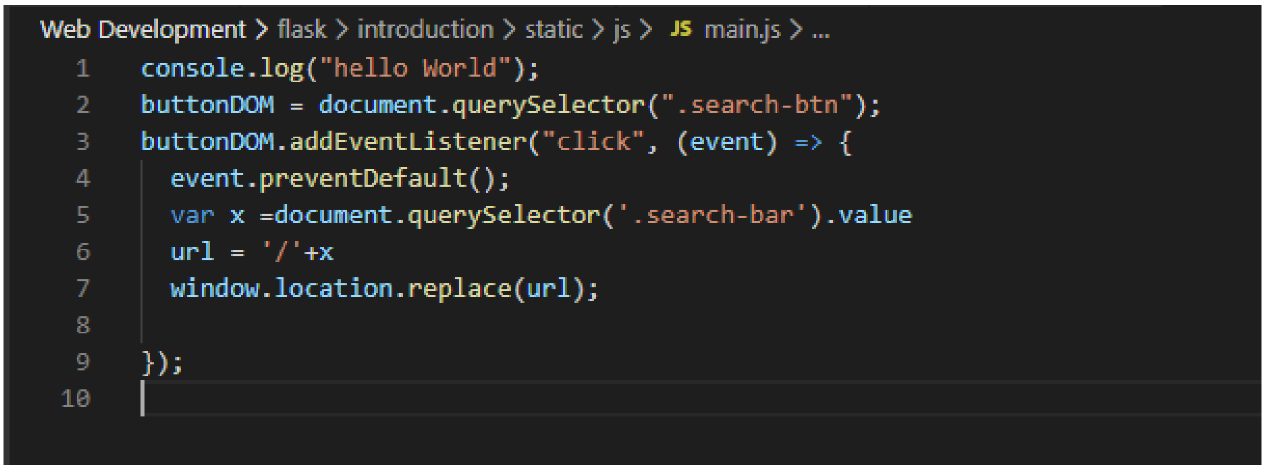
```css
* {
  box-sizing: border-box;
}
body {
  margin: 0;
  font-family: "Montserrat", sans-serif;
}
.main-header {
  /* background: #63b7af; */
  background: #004445;
  padding: 4px 16px;
  vertical-align: middle;
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
}
.main-header__brand {
  display: inline-block;
}
.main-header__brand > a {
  text-decoration: none;
  color: #000839;
  font-size: 24px;
  font-weight: bold;
}
.main-nav {
  display: inline-block;
  padding: 8px;
  width: calc(100% - 300px);
  text-align: center;
}
```



```css
.search-bar {
  display: inline-block;
  padding: 6px;
  background: #142850;
  border: 2px solid #142850;
  border-radius: 4px;
  width: 300px;
  color: white;
}
.search-bar:focus {
  outline: none;
  border: none;
}
.search-btn {
  display: inline-block;
  padding: 6px;
  background: #142850;
  color: white;
  border-radius: 4px;
  border: 2px solid #142850;
  margin: 0 8px;
  width: 100px;
}
.search-btn:focus {
  outline: none;
}

.search-btn:hover,
.search-btn:active {
  background: #1f4068;
  border: 2px solid #1f4068;
  cursor: pointer;
}
```


```css

.main-background{
  background: url('../img/background2.jpg');
  width: 100%;
  height: 700px;
  background-position: center;
}
```

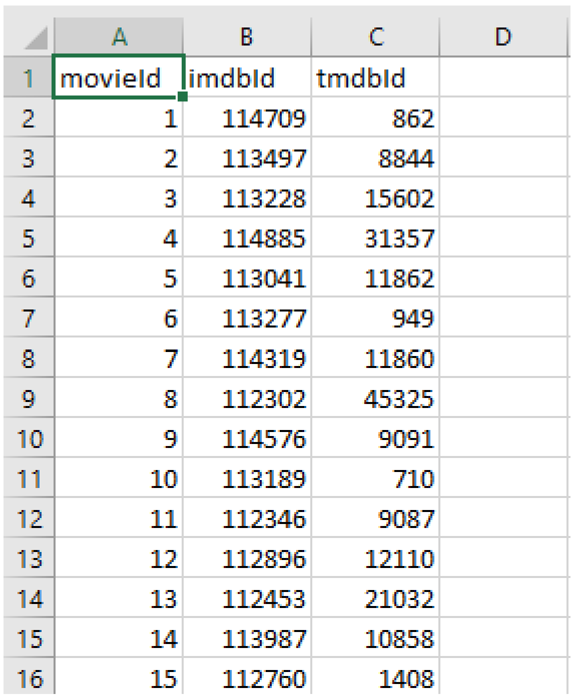
Main.js –filename


Web Development > flask > introduction > static > js > JS main.js > ...

```javascript
console.log("hello World");
buttonDOM = document.querySelector(".search-btn");
buttonDOM.addEventListener("click", (event) => {
  event.preventDefault();
  var x =document.querySelector('.search-bar').value
  url = '/'+x
  window.location.replace(url);

});

```

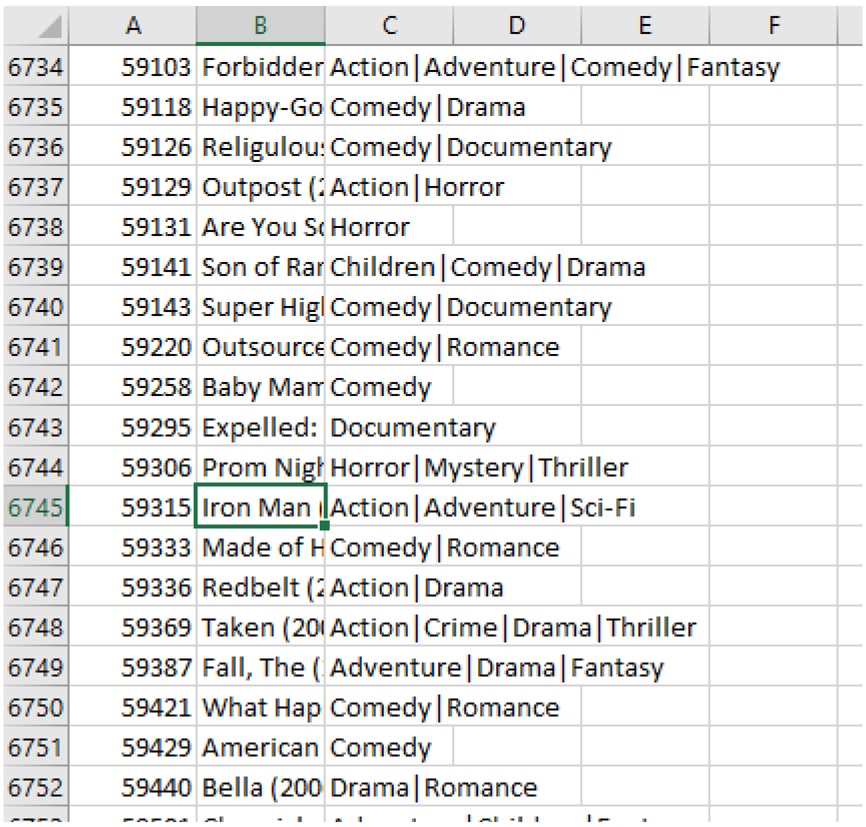
Dataset

Links.csv



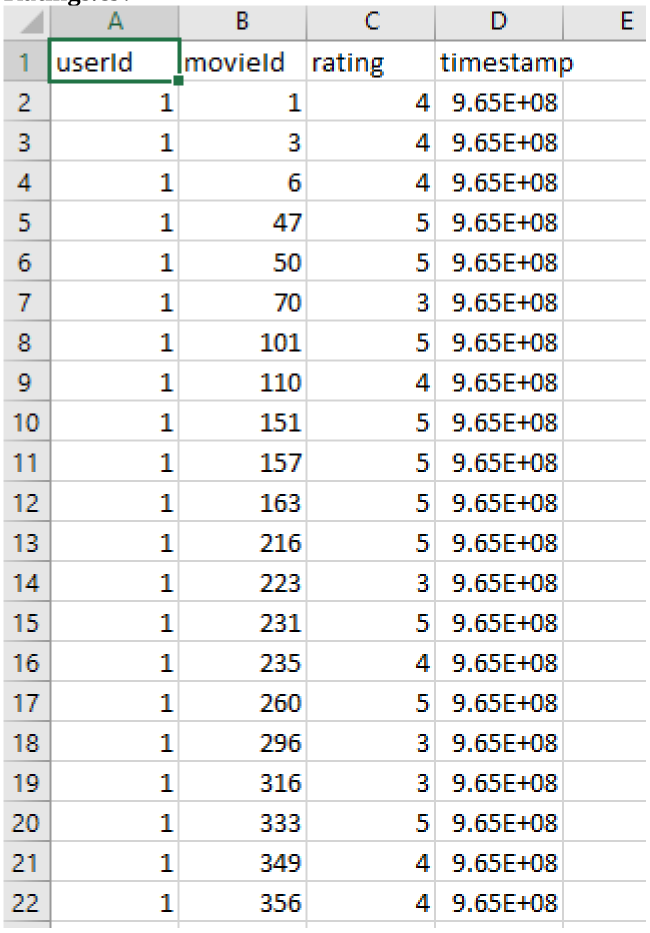
	A	B	C	D
1	movieId	imdbId	tmdbId	
2	1	114709	862	
3	2	113497	8844	
4	3	113228	15602	
5	4	114885	31357	
6	5	113041	11862	
7	6	113277	949	
8	7	114319	11860	
9	8	112302	45325	
10	9	114576	9091	
11	10	113189	710	
12	11	112346	9087	
13	12	112896	12110	
14	13	112453	21032	
15	14	113987	10858	
16	15	112760	1408	

Movies.csv



	A	B	C	D	E	F
6734	59103	Forbidder	Action|Adventure|Comedy|Fantasy			
6735	59118	Happy-Go	Comedy|Drama			
6736	59126	Religulou:	Comedy|Documentary			
6737	59129	Outpost (:	Action|Horror			
6738	59131	Are You Sc	Horror			
6739	59141	Son of Rar	Children|Comedy|Drama			
6740	59143	Super Higl	Comedy|Documentary			
6741	59220	Outsource	Comedy|Romance			
6742	59258	Baby Mam	Comedy			
6743	59295	Expelled:	Documentary			
6744	59306	Prom Nigl	Horror|Mystery|Thriller			
6745	59315	Iron Man (	Action|Adventure|Sci-Fi			
6746	59333	Made of H	Comedy|Romance			
6747	59336	Redbelt (:	Action|Drama			
6748	59369	Taken (20(	Action|Crime|Drama|Thriller			
6749	59387	Fall, The (	Adventure|Drama|Fantasy			
6750	59421	What Hap	Comedy|Romance			
6751	59429	American	Comedy			
6752	59440	Bella (200	Drama|Romance			

Ratings.csv



	A	B	C	D	E
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	
16	1	235	4	9.65E+08	
17	1	260	5	9.65E+08	
18	1	296	3	9.65E+08	
19	1	316	3	9.65E+08	
20	1	333	5	9.65E+08	
21	1	349	4	9.65E+08	
22	1	356	4	9.65E+08	

5. RESULT

We tested our algorithms by removing portions of the dataset then trying to predict what the users would have rated the removed items. This is a simple test which allows us to see how far off our predictions are from the actual rating the user gave.

In order to rate the amount of error from the actual predictions, we used the root mean squared error (RMSE) metric. It can be thought of as a weight average error, emphasizing large errors and promoting small ones. It was useful to us for three reasons - first, it is a simple, one number measurement of success. Second, its emphasis on large errors was helpful in tracking down how far off we were from the actual predictions. Third, the NetflixPrize used it.

In order to determine the amount of success, we used a control algorithm. The control algorithm - that is, the most basic prediction method - is simply using the global average for an item as the predicted value. This is a simple algorithm, so it is important to us to perform better than it.

The Movielens dataset was easy to test on. It was relatively small (with only 100,000 entries) and already had two test sets created, ua and ub. The results below are for the ua dataset.

Ultimately most of our algorithms performed well. The two that performed worse than global average did so based upon difficulty working with such a small, sparse dataset.

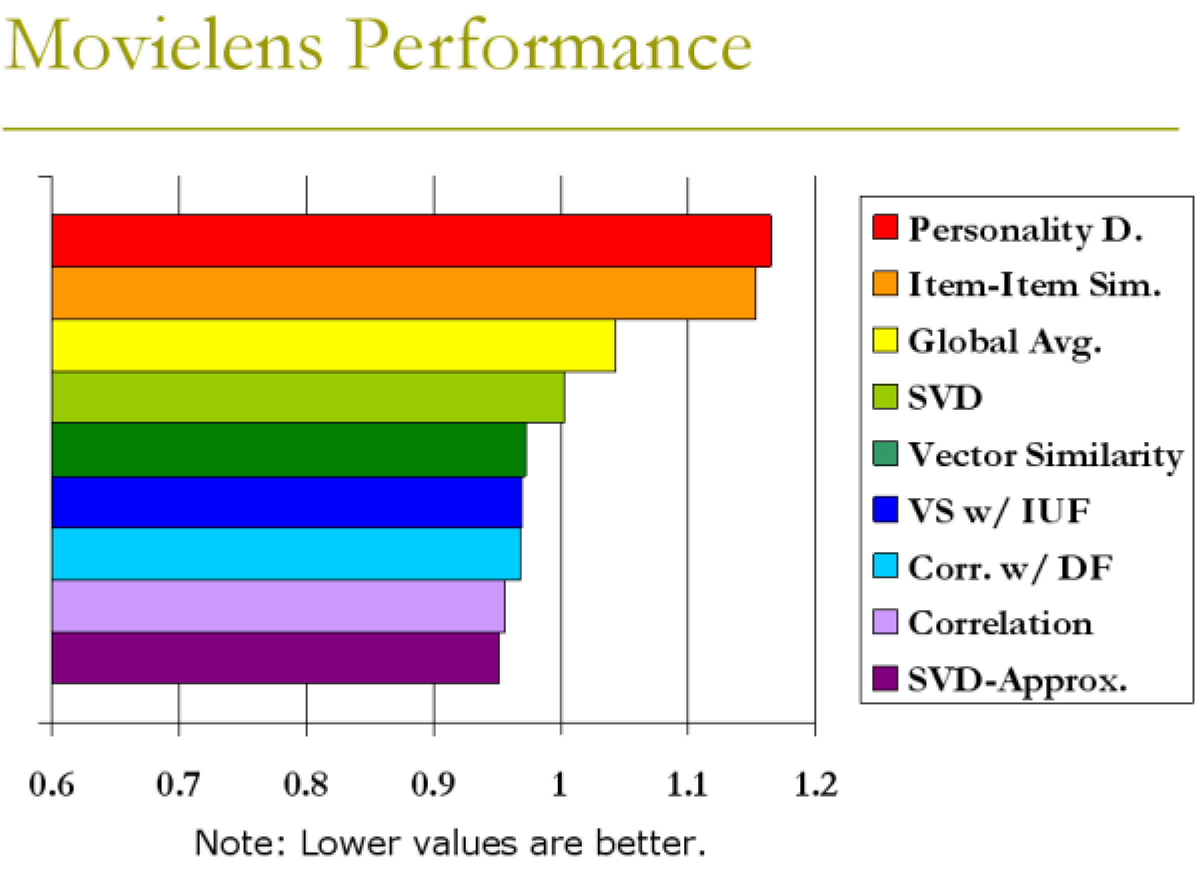


Figure 6 movie lens dataset performance

The NetflixPrize dataset was absolutely gigantic, with 100 million entries. Also, their testing set had 2 million predictions they wanted. Thus, we only had time to try out two algorithms on the data.

Of the two we tried, both performed better than global average. One performed almost as well as the Netflix algorithm, and one performed better.

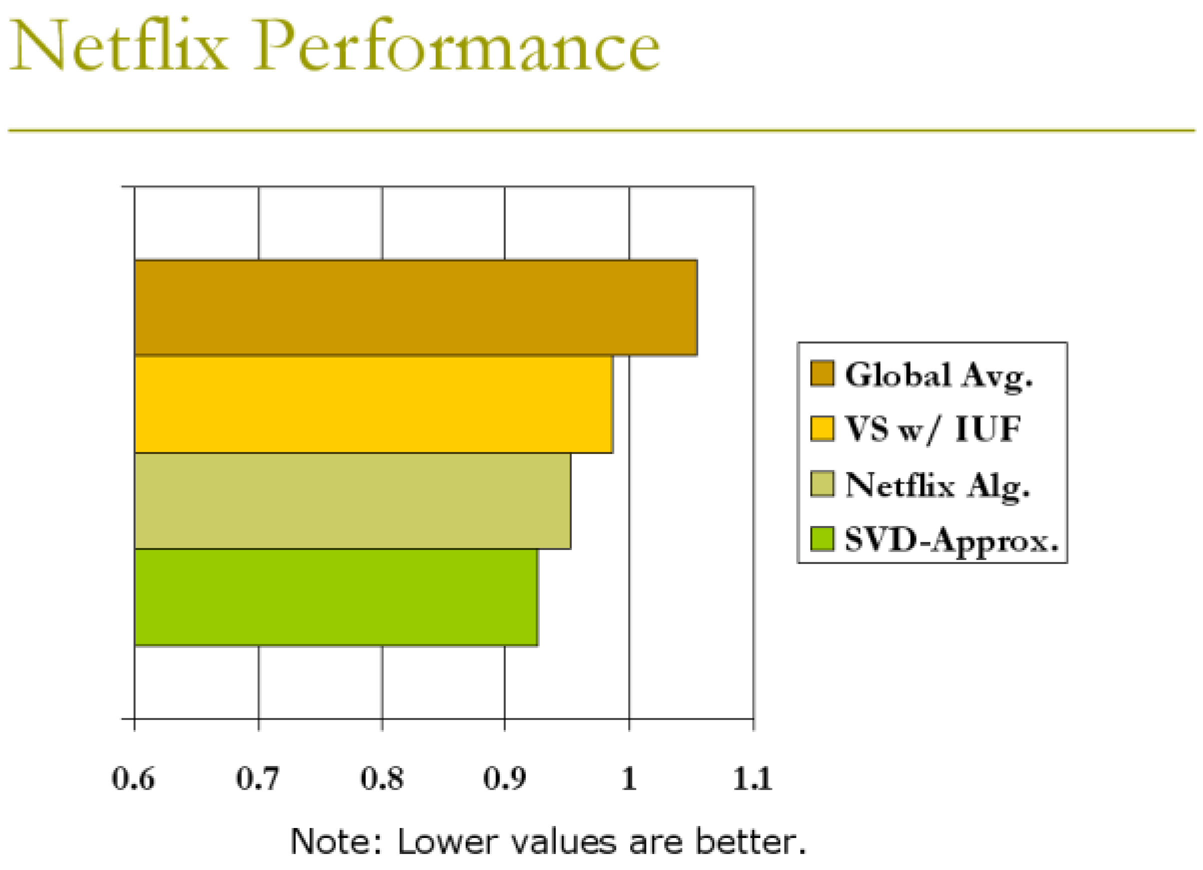
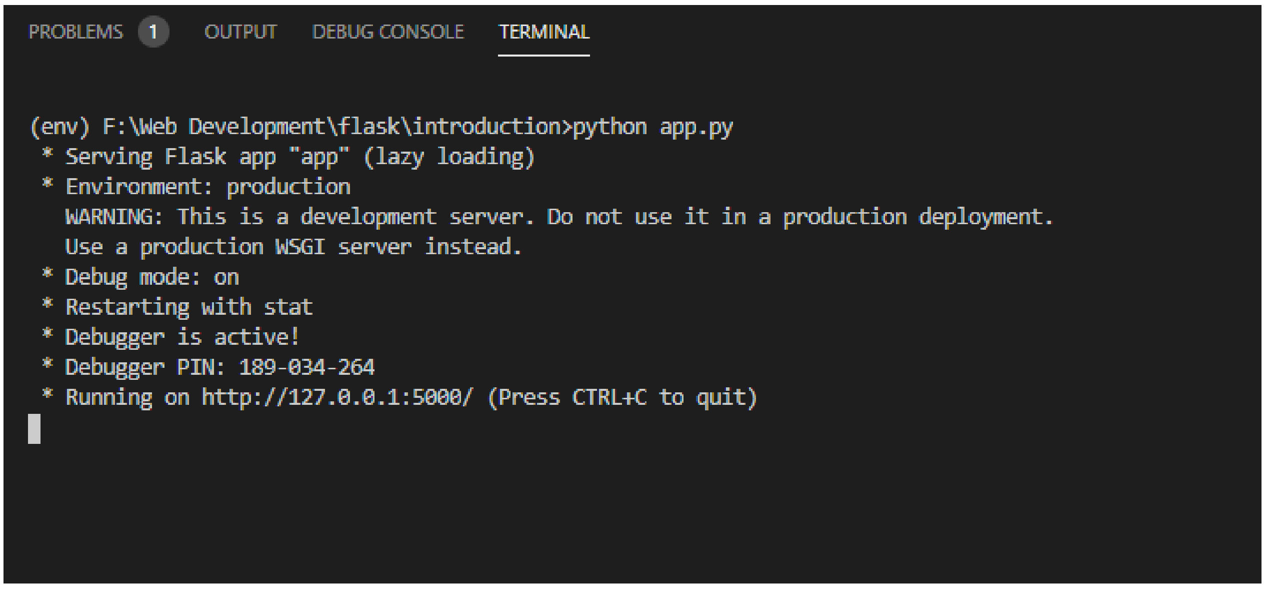


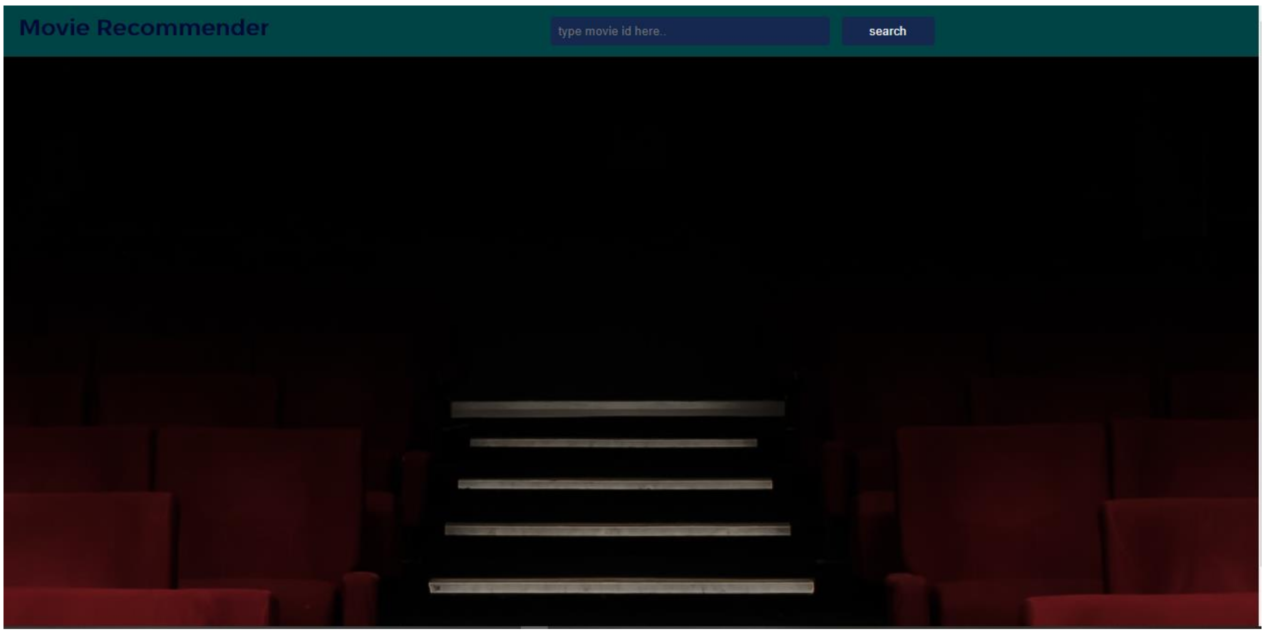
Figure 7 Netflix dataset performance



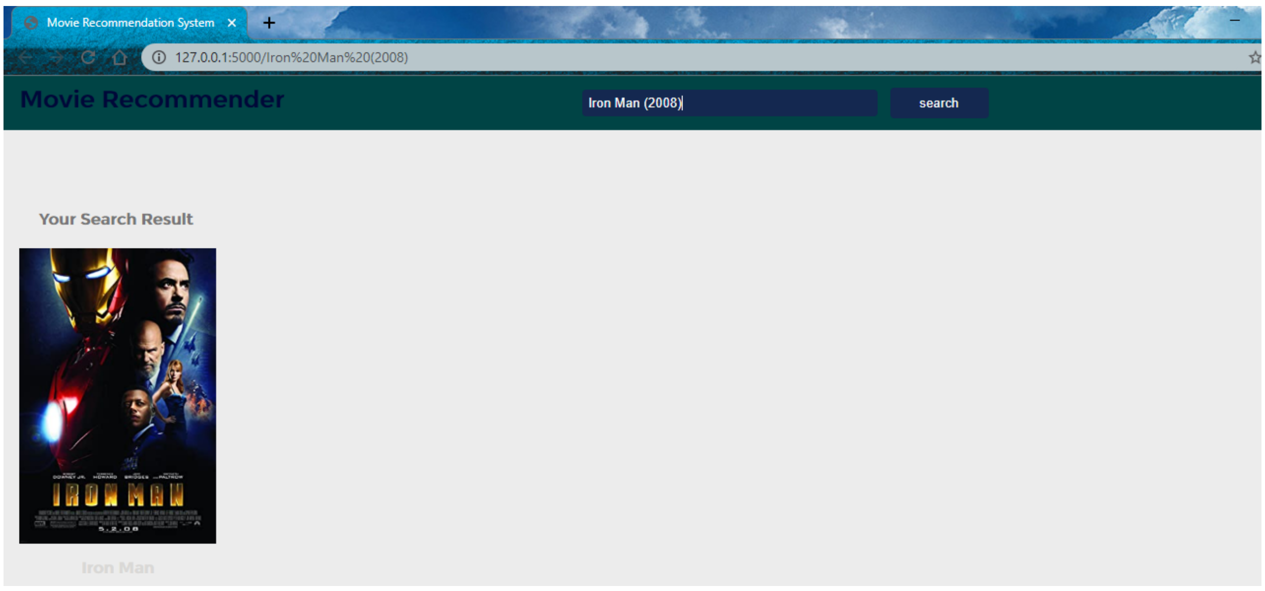
```
(env) F:\Web Development\flask\introduction>python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 189-034-264
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

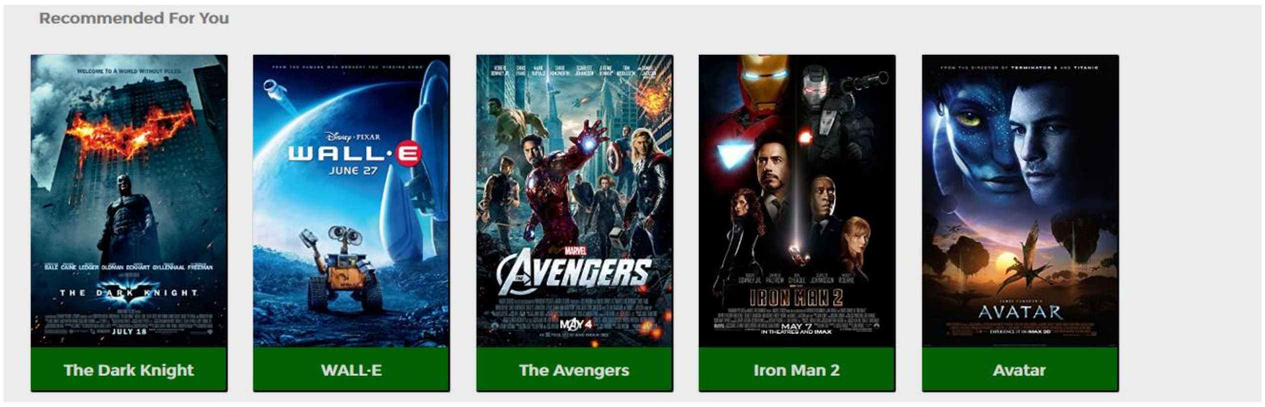
Figure 8 server side response

Website home Page



Search Result of Iron Man





Recommended For You

The Dark Knight

WALL-E

The Avengers

Iron Man 2

Avatar

6. Conclusion

In our proposed model we used collaborative filtering technique to for recommending movies. In this model we have used the data set from movie lens which is latest and generated in 2018. That is consist of four csv files out of one is ratings.csv in this file ratings are given by use are stored. In this project we have used cosine similarity for comparing two vector and we know that cosine value ranges from -1 to 1 but in case of our data everything is positive so we are getting values between 0 and 1 that is used to decide whether given data is likely to be similar to the item or not and we used top-k approach and we just returned top k item from a list of recommended items.

In this project we build a website using flask and another technologies like html, css, JavaScript, python. In this project I not just write algorithm but also I embed the algorithm with website that and we used flask for build Rest Api Services. Flask is lightweight and easy to use with another python libraries.

7. REFERENCES

[1] Joan E. Ball-Damerow, et. al., “Research applications of primary biodiversity databases in the digital age”, PLoS One, 2019.

[2] Richard A. Niesenbaum, “The Integration of Conservation, Biodiversity, and Sustainability”, sustainability MDPI, 2019.

[3] Alho C., et. al., “The value of biodiversity”, Brazilian Journal of Biology,2008.

[4] Irving Vasquez, “VIGIA: Autonomous Surveillance of Biosphere Reserves” Proposal : 2016-01-2341.

[5] Karen Simonyan, et. al., “Very Deep Convolutional Networks for Large-Scale Image Recognition”, ICLR, 2015.

[6] Kaiming He, et. al.,”Deep Residual Learning for Image Recognition”, IEEE, 2016.

[7] Sara Oldfield ,“Cactus and Succulent Plants”,IUCN, 1997.

[8] Stephane Lathuili ´ ere , et.al., “A Comprehensive Analysis of Deep Regression”, IEEE, 2018.

[9] Ramasubramanian, K., et. al.,” Machine Learning Using R”, Springer, 2019.

[10] Rodrigo Fernandes de Mello, et. al., “Machine Learning A Practical Approach on the Statistical Learning Theory”, Springer, 2018.

[11] Jianlong Zhou, et. al.,“Human and Machine Learning: Visible, Explainable, Trustworthy

and Transparent”, Human–Computer Interaction Series -Springer,2020.

[12] Raffaele Cioffi, et. al., ”Artificial Intelligence and Machine Learning Applications in

Smart Production: Progress, Trends, and Directions”, MDPI, 2020.

[13] Efren López-Jiménez, et. al., “Columnar cactus recognition in aerial images using a deep

learning approach”, Ecological Informatics , 2019.

[14] Yu Sun, Yuan Liu, et. al., Deep Learning for Plant Identification in Natural

Environment”, Hindawi Computational Intelligence and Neuroscience”, 2017.

[15] Geoffrey A. Fricker, et. al., “A Convolutional Neural Network Classifier Identifies Tree

Species in Mixed-Conifer Forest from Hyperspectral Imagery”, Remote Sensing MDPI,

2019.

[16] Mathieu Carpentier , et. al., “Tree Species Identification from Bark Images Using

Convolutional Neural Networks”, IEEE, 2018.

[17] Soon Jye Kho, et. al., “Automated plant identification using artificial neural network and

support vector machine”, Frontiers in Life Science, 2017.

[18] Munisami T.,et. al., “Plant Leaf Recognition Using Shape Features and Colour

Histogram with K-nearest Neighbour Classifiers”, Procedia Computer Science, 2015.

[19] Julia Marrs, et. al., “Machine Learning Techniques for Tree Species Classification Using

Co-Registered LiDAR and Hyperspectral Data”, Remote Sensing MDPI, 2019.