Readme:

My information:

Qingyi Wang (22014713)

Github Link： https://github.com/Qingyi-Www/PersonalMovieRecommendation

Reference:

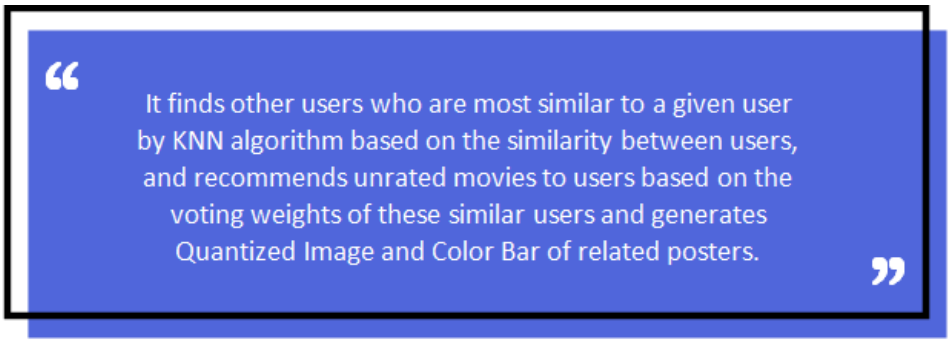
KNN： https://github.com/LJSthu/Movie-Analysis

Movie Poster Color Bars： https://github.com/joshwcheung/movie-poster-color-bars

IMDB Movie Posters' Dates： https://www.kaggle.com/datasets/neha1703/movie-genre-from-its-poster

Chatgpt:
Cleaning database.
KNN and Color Bars code merge section problem solving.
The new fig, ax = plt.subplots(1, 2, figsize=(10, 6)) fiction in Color Bars.
Don't show movies without posters Part.

Algorithm & Implementation:



Work Steps:

1. Model testing:

1) The evaluate function is outdated and cannot be used in the latest version of the Surprise library. So I use the cross_validate function to evaluate the performance of the model.

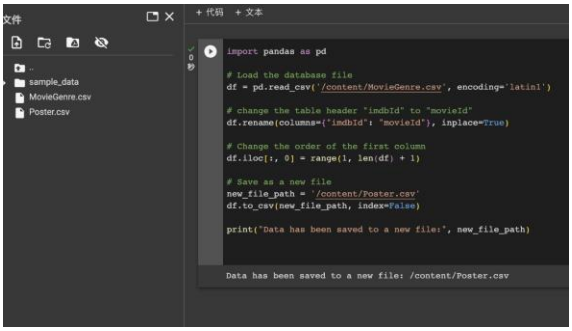
```
# data.split(n_folds=5)
self.svd = SVD(n_epochs=20, n_factors=100, verbose=True)
cross_validate(self.svd, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
trainset = data.build_full_trainset()
```

```
test = Personal_KNN_recommender()
user_id = int(input("Enter user ID: "))
result = test.recommend(user_id, 10)
for movie_title in result:
    print(movie_title)
```

```
Estimating biases using als...
Computing the pearson_baseline similarity matrix...
Done computing similarity matrix.
Enter user ID: 8
Quiz Show (1994)
Aladdin (1992)
Stargate (1994)
Dead Man Walking (1995)
Heat (1995)
Star Trek: Generations (1994)
Speed (1994)
Much Ado About Nothing (1993)
Piano, The (1993)
Leaving Las Vegas (1995)
```

2) By adding the input function, the program will prompt the user to enter the user ID and pass the input value to the recommendation function, and then output the corresponding recommendation result.

2. Modify the dataset (import a csv containing the poster url):



I need to modify the name within the dataset(I asked **Chatgpt**) the *imdbId* of the poster csv to *movieId*, as the identifier of the two databases, while the original author dataset of movies is sorted according to from 1, did not follow the *imdb* id, but I looked carefully and found that the order of movies inside the two datasets is the same, so I in So I renamed the first column from 1 in poster.csv.

3. Add Posters Links

```
class Personal_KNN_recommender:
    def __init__(self, mode=0):
        self.index = pd.read_csv('/content/movies.csv')
        self.poster_urls = pd.read_csv('/content/Poster.csv').set_index('movieId')['Poster'].to_dict()
        self.reader = Reader()
        self.ratings = pd.read_csv('/content/train.csv')
        self.testings = pd.read_csv('/content/test.csv')
        data = Dataset.load_from_df(self.ratings[['userId', 'movieId', 'rating']], self.reader)
        trainset = data.build_full_trainset()
        sim_options = {'name': 'pearson_baseline', 'user_based': True}
        if mode == 0:
            self.algo = KNNBaseline(sim_options=sim_options)
        elif mode == 1:
            self.algo = KNNWithMeans(sim_options=sim_options)
        elif mode == 2:
            self.algo = KNNBasic(sim_options=sim_options)
        else:
            exit(0)
        self.userid = []
        for i in range(len(self.testings['userId'])):
            if not self.testings['userId'][i] in self.userid:
                self.userid.append(self.testings['userId'][i])
        self.algo.fit(trainset)
```

1) Use the *pd.read_csv()* function to read the CSV file data from the file path *"/content/Poster.csv"*, and return a DataFrame object containing the data.
2) The *.set_index()* method sets the index of the DataFrame object to the value of the *"movieId"* column. This will make the *"movieId"* column become the index column of the data, so that it can be easily retrieved according to the movie ID.

3) *['Poster']:* This part of the code selects the "Poster" column of the DataFrame object by indexing

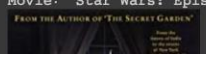
4. Don't show movies without posters

(dataset problem, some posters don't have URLs)

```
test = Personal_KNN_recommender()
user_id = int(input("Please enter your user ID:"))
result, poster_urls = test.recommend(user_id, 10)
for i in range(len(result)):
    print("Movie:", result[i])
    if poster_urls[i]:
        urllib.request.urlretrieve(poster_urls[i], f"poster_{i+1}.jpg")
        display(DisplayImage(filename=f"poster_{i+1}.jpg"))
    print()
```


```
Movie:  The Martian (2015)

Movie:  Star Wars: Episode IV - A New Hope (1977)
```

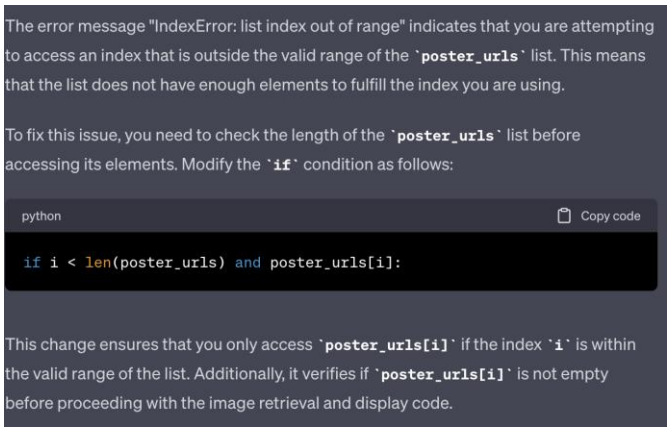


There is a problem, some movies don't have posters/url need to be updated, it's a database problem, then I used **Chatgpt** to help me change the code so that the code doesn't recommend movies that don't have posters

```python
for i in result:
    movie_id = i[0]
    recommending.append(self.index[self.index.movieId == movie_id]['title
    poster_url = self.poster_urls.get(movie_id)
    if poster_url:
        try:
            response = urllib.request.urlopen(poster_url)
            poster_urls.append(poster_url)
        except urllib.error.HTTPError:
            continue  # Skip invalid poster URLs
    else:
        continue  # Skip invalid poster URLs
```

1. After retrieving the poster URL for a movie, we added an `if poster_url:` condition to check if the URL exists.
2. Inside the `if` block, we wrapped the code that attempts to open the URL in a try-except block to catch `urllib.error.HTTPError` exceptions.
3. If an HTTPError exception occurs, indicating that the poster URL is invalid or cannot be accessed, we use the `continue` statement to skip the rest of the loop and move on to the next movie.
4. Similarly, if the `poster_url` itself is `None`, indicating that the movie doesn't have a valid poster URL, we also use the `continue` statement to skip the rest of the loop.

By adding these modifications, the code will skip movies with invalid poster URLs and continue with the next movie in the loop.



The error message "IndexError: list index out of range" indicates that you are attempting to access an index that is outside the valid range of the `poster_urls` list. This means that the list does not have enough elements to fulfill the index you are using.

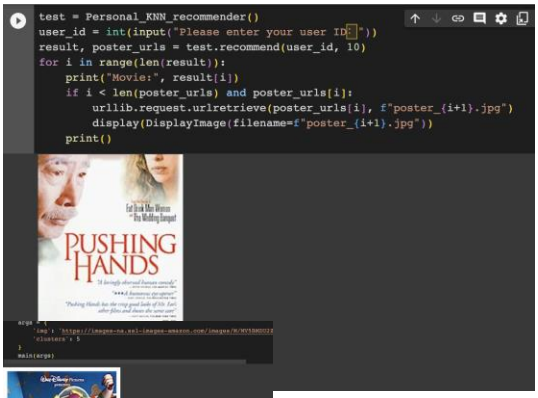
To fix this issue, you need to check the length of the `poster_urls` list before accessing its elements. Modify the `if` condition as follows:

python Copy code

```python
if i < len(poster_urls) and poster_urls[i]:
```

This change ensures that you only access `poster_urls[i]` if the index `i` is within the valid range of the list. Additionally, it verifies if `poster_urls[i]` is not empty before proceeding with the image retrieval and display code.

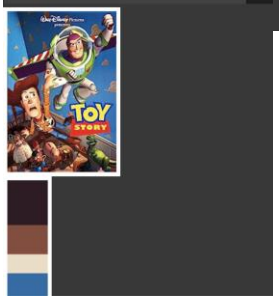
Results:



```
test = Personal_KNN_recommender()
user_id = int(input("Please enter your user ID: "))
result, poster_urls = test.recommend(user_id, 10)
for i in range(len(result)):
    print("Movie:", result[i])
    if i < len(poster_urls) and poster_urls[i]:
        urllib.request.urlretrieve(poster_urls[i], f"poster_{i+1}.jpg")
        display(DisplayImage(filename=f"poster_{i+1}.jpg"))
    print()
```


I don't want to save the posters to a folder, but just want to display them. so I use *if i < len (...) and poster_urls [i]:* to check whether the index of the current recommended movie is within the valid range of the poster URL list and whether the poster URL exists

5. Add Color Bars



1) On the basis of the original code (https://github.com/joshwcheung/movie-poster-color-bars), test whether the poster of the URL link in the database can be recognized properly, the result is success.

2) Import the required libraries and modules in: Color Bars

```
!pip install matplotlib
import argparse
import cv2
from PIL import Image
import io
import matplotlib.pyplot as plt
```

3) Modify the final display, here I want to display the Quantized Image and Color Bar in the url section of the poster.

A graph window with two subplots is created using the Matplotlib library. **1** represents one row and **2** represents two columns, and the size of the graph window is **10** in width and **6** in height.

Set the display image title for the subgraphs.

```python
def generate_color_bar(img_url, clusters, movie_title):
    response = urllib.request.urlopen(img_url)
    img = np.array(Image.open(BytesIO(response.read())))
    clustered, label, center = kmeans_color_quant(img, clusters)
    hist = get_histogram(label)
    bar = draw_color_bar(hist, center)

    fig, ax = plt.subplots(1, 2, figsize=(10, 6))
    ax[0].imshow(clustered)
    ax[0].axis('off')
    ax[0].set_title('Quantized Image')

    ax[1].imshow(bar)
    ax[1].axis('off')
    ax[1].set_title('Color Bar')

    # Add a movie title
    fig.suptitle(movie_title, fontsize=16, fontweight='bold')

    plt.show()
```

4) In the Input Part, the code calls the *generate_color_bar* function, passing the poster URL of the current recommended movie, the number of clusters and the movie title as parameters to generate the corresponding color bar.

```python
recommender = Personal_KNN_recommender()

# Get recommended movies
user_id = int(input("Please enter your user ID: "))
recommended_movies, recommended_poster_urls = recommender.recommend(user_id, 10)

# Call the Code Color Bars function to generate a color bar
clusters = 5
for i in range(len(recommended_movies)):
    print("Movie: ", recommended_movies[i])
    if i < len(recommended_poster_urls) and recommended_poster_urls[i]:
        generate_color_bar(recommended_poster_urls[i], clusters, recommended_movies[i])
    print()
```