YourNextReads - a web app built using the Django framework

A Book recommendation engine - using machine learning algorithms

Divyata Gosai

29th May 2022

1. Abstract

The YourNextReads is a web app built for book lovers with a Book Recommendation engine. A web app uses a hybrid recommendation engine with different machine learning algorithms to power the web app's book recommendations. Using data like rating books from a user, Recommendation Engine uses Machine Learning algorithms to provide a user with highly personalized book recommendations.

Recommender engines are at the forefront of innovation in how content-serving companies such as Facebook, Amazon, and Spotify interact with their users. Given this scenario, websites must aim to provide the most personalized content possible. A recommendation engine filters the data using different algorithms and recommends the most relevant items to users. It first captures the past behaviour of a customer and based on that, recommends products which the users might be likely to buy.

I have developed a hybrid recommender system to provide recommendations from the dataset given by Goodreads users (ratings and item features). In addition, it also provides more 'traditional' recommendations that only use the book's features.

2. Introduction

I have used a hybrid recommendation engine which uses different kinds of machine learning algorithms to power a web app's book recommendations. A hybrid recommendation system is the combination of two diffrent types of recommender systems: content-based filtering and collaborative filtering.

Firstly, content-based filtering is a method of recommending items by the similarity of the said items. For example, if I like the first book of The Hunger Games, then it gonna recommend me similar books such as Catching Fire (the second book of The Hunger Games), and Mockingjay (the third book of The Hunger Games). For content-based filtering, I implemented cosine similarity metrics machine learning algorithm.

Secondly, collaborative filtering is a method by which user ratings are used to determine user or item similarities. For instance, If there is a high correlation between users rating the first Lord of the Rings book and the second Lord of the Rings book, then they are deemed to be similar. For collaborative filtering, I have applied two machine learning algorithms called K-means clustering and Gaussian mixture modelling to cluster the users and reach the best silhouette score.

2.2 **Tools/Technologies Used**

Programming languages: HTML, CSS, Javascript, Python, MySQL

Front-end : HTML, CSS, Javascript, bootstrap

Technologies: Django framework, machine learning model

Databases/Data storage: MySQL

ML Libraries in python 3: NumPy, pandas, sklearn

Tools: Visual Studio Code, Jupyter Notebook, MySQL workbench, MySQL server

3. Software Requirement Specification

3.1 Product Scope

Our main goal is to provide the best recommendation to users.

3.2 Types of User

1. End-user

3.3 System Functional Requirements

R1: Registration

Description: The user can register

Input: Enter user details like email, password, name etc.

Output: Register successfully

R2: Login

Description: User can login

Input: Enter login details

Output: Login successfully and redirect to the home page

R3: Logout

Description: The user can log out after finishing work.

Input: User selection.

Output: Redirection to Login page.

R4: End-user

R-1.1: Get suggested books.

Description: The machine can automatically suggest books using different algorithms of machine learning.

Input: Area of interest

Output: All the suggested books

R-1.2: Add an area of interest

Description: Using this data machine can automatically suggest books accordingly user's interest.

Input: Different Genres

Output: Genre added to the area of interest.

R-1.3: View book details

Description: using data to show details of the book

Input: to view book details

Output: all book-related details such as average rating, genres, Format, price, edition, etc.

R-1.4: Submit book rating

R-1.5: Show rated books

R-1.6 Add a book on

R-1.6: Search book

Description: Users can search books using different parameters.

R-1.2.1: Search book with book name.

Input: Book name

Output: Book details of the given book

R-1.2.2: Search book with author name.

Input: Author name

Output: Book details of every book written by that author

R-1.2.3: Search book with genre name.

Input: Genre

Output: Book details of every book fall in the given genre.

3.4 Other Non-functional Requirements

1. Performance

The system must be interactive and the delays involved must be less. So in every action-response of the system, there are no immediate delays. In the case of opening App components, popping error messages and saving the settings or sessions there is a delay much below 3 seconds.

2. Safety

User details should be securely stored on the server. The main security concern is for user accounts hence proper login mechanism should be used to avoid hacking.

3. Reliability

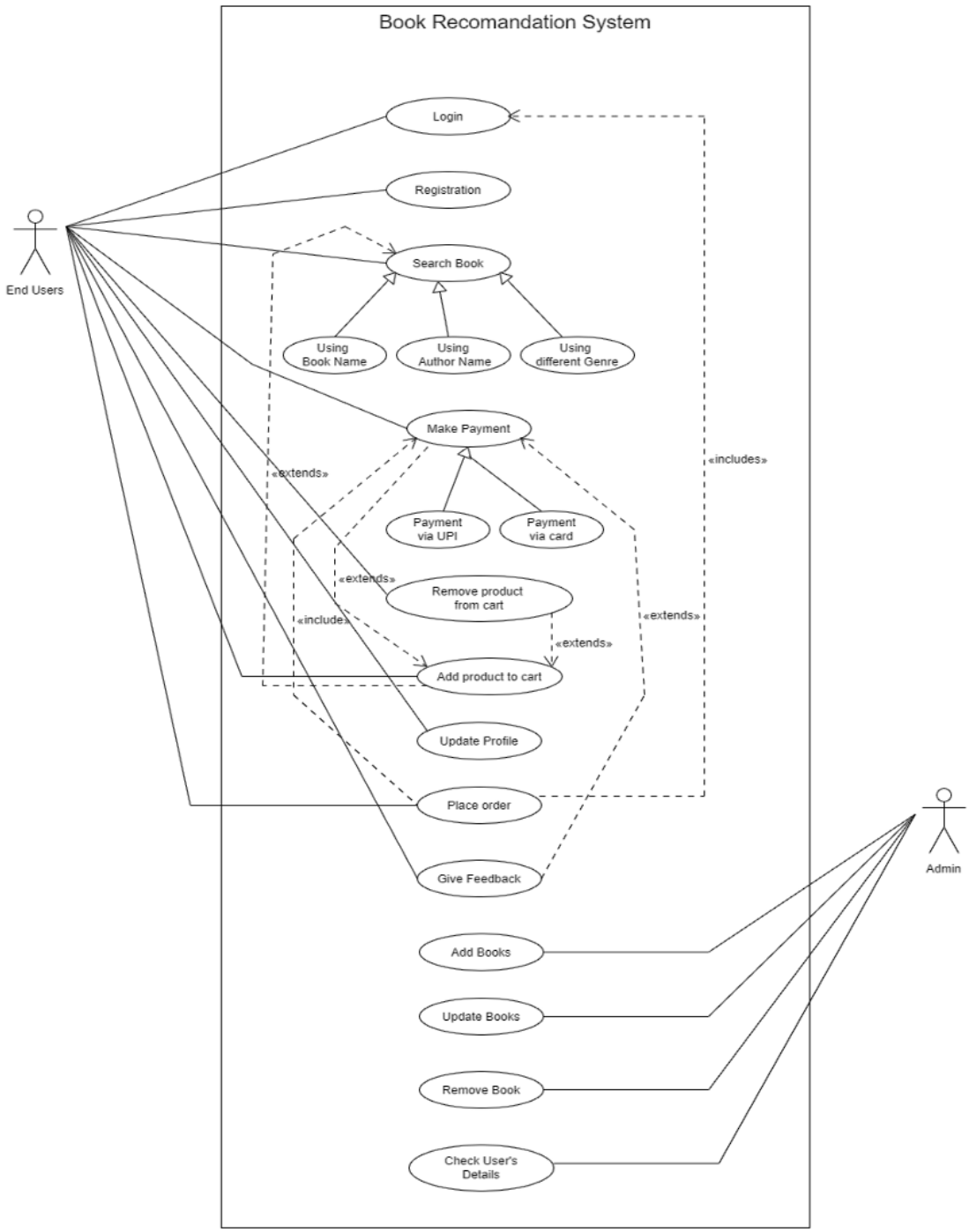
As the system provides the right tools for discussion and problem-solving, it must be made sure that the system is reliable in its operations and securing sensitive details.

4. Database

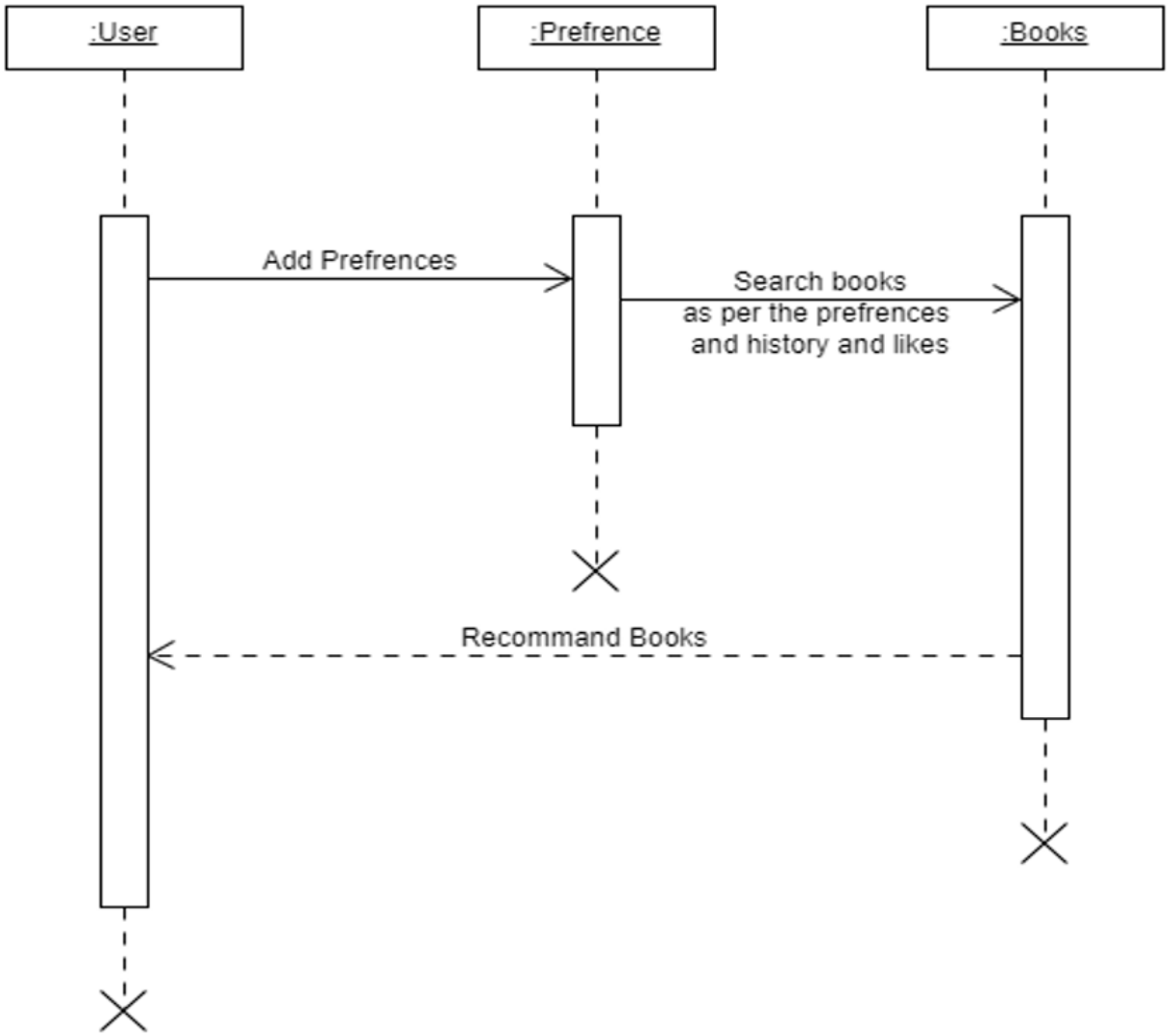
The system requires to access users' data fast to maintain its performance.

4. Design

4.1: User case diagram



4.2 Sequence Diagram:



4.3 Data set Details:

Dataset: https://github.com/zygmuntz/goodbooks-10k

Dataset's fields:

- book_authors
  Description: This field include the name of the author
  Usage: using this field in searching functionality, so Users can search books using the author's name.

- book_desc
  Description: This field gives descriptions of books
  Usage: using this field to recommend people which kind of book, they are interested in. I am using a Machine learning algorithm to find semantic similarity to find similar books based on the description.

    - book_format
      Description: This field tells book is on hardcover or in paper cover.

    - book_rating
      Description: This field shows the average rating of a book.
      Usage: Using the book_rating field to recommend books to the people based on public review.

    - book_rating_count
      Description: This field counts the total number of rating counts i.e. how many people had given ratings to the book?

- book_title
  Description: This field shows the title of the book.
  Usage: I am using this field in searching functionality, so Users can search books using the author's name.

- genres
  Description: This field shows the genres of the book. There can be many genres related to one book i.e. Catching fire, this book contains Young adult and science fiction etc. genres.
  Usage: using this field in searching functionality. In that users can search books using the author's name. Also, use this field to predict books which are having same genres.

- image_url
  Description: This field contains the URL of the book cover page, so using this field while displaying the books in UI. This is a good way to display images using URL rather than storing all the photos in a database. All the URLs are stored online so you must have Internet Connection to access it.

5. Implementation Details

5.1: Implementation of Collaborative Machine Learning Model:

After loading and analyzing the data set, I have performed Data preprocessing to reduce the data size, merge the required data and drop unnecessary columns and rows.

In [4]:
```
# Merge the two tables then pivot so we have Users X Books dataframe.
ratings_title = pd.merge(ratings, books[['book_id', 'book_title']], on='book_id' )
user_book_ratings = pd.pivot_table(ratings_title, index='user_id', columns= 'book_title', values='rating')

print('dataset dimensions: ', user_book_ratings.shape, '\n\nSubset example:')
user_book_ratings.iloc[:25, :10]
```

```
dataset dimensions:  (53424, 2945)
```

In [5]:
```
# Drop users that have given fewer than 100 ratings of these most-rated books
user_book_ratings = user_book_ratings.dropna(thresh=100)

print('dataset dimensions: ', user_book_ratings.shape, '\n\nSubset example:')
user_book_ratings.iloc[:25, :10]
```

```
dataset dimensions:  (14421, 2945)
```

To further reduce dimensionality, I applied truncated SVD, which is available for sparse matrices. I've determined that if I use 200 features, these features will explain about 45 percent of the variance in the data.

So I have scored the remaining 14,421 users on 200 features, each of which represents some combination of opinions about various books.

```
In [7]: # view result in a Pandas dataframe, applying the original indices
indices = user_book_ratings.index

book_ratings_for_clustering = pd.DataFrame(data=user_book_ratings_tsvd).set_index(indices)
print('dataset dimensions: ', book_ratings_for_clustering.shape, '\n\nSubset example:')

book_ratings_for_clustering.iloc[:25, :10]
```

```
dataset dimensions:  (14421, 200)
```

Then, I separated 20 percent of the users to create a test set.

```
In [11]: from sklearn.model_selection import train_test_split
book_ratings_training, book_ratings_testing = train_test_split(book_ratings_for_clustering, test_size=0.20, random_state=42)

print('Training data shape: ', book_ratings_training.shape)
print('Testing data shape: ', book_ratings_testing.shape)
```

```
Training data shape:  (16874, 200)
Testing data shape:  (4219, 200)
```

Then I performed clustering on the training set. I have applied K-means clustering and Gaussian mixture modelling to cluster the users and reach the best silhouette score. After applying these 2 methods with various cluster counts, I have got the conclusion that K-means clustering with the cluster count of 7 gives the best silhouette score.

```
In [19]: # trying with the training data after preprocessing
from sklearn.cluster import KMeans

clusterer_KMeans = KMeans(n_clusters=7).fit(book_ratings_training)
preds_KMeans = clusterer_KMeans.predict(book_ratings_training)

from sklearn.metrics import silhouette_score
kmeans_score = silhouette_score(book_ratings_training, preds_KMeans)
print(kmeans_score)
```

```
0.0433968332584411
```

```
In [18]: # trying with the training data after preprocessing
from sklearn.mixture import GaussianMixture

clusterer_GMM = GaussianMixture(n_components=7).fit(book_ratings_training)
preds_GMM = clusterer_GMM.predict(book_ratings_training)

GMM_score = silhouette_score(book_ratings_training, preds_GMM)
print(GMM_score)
```

```
0.016778872313688933
```

Then I analyzed each cluster.

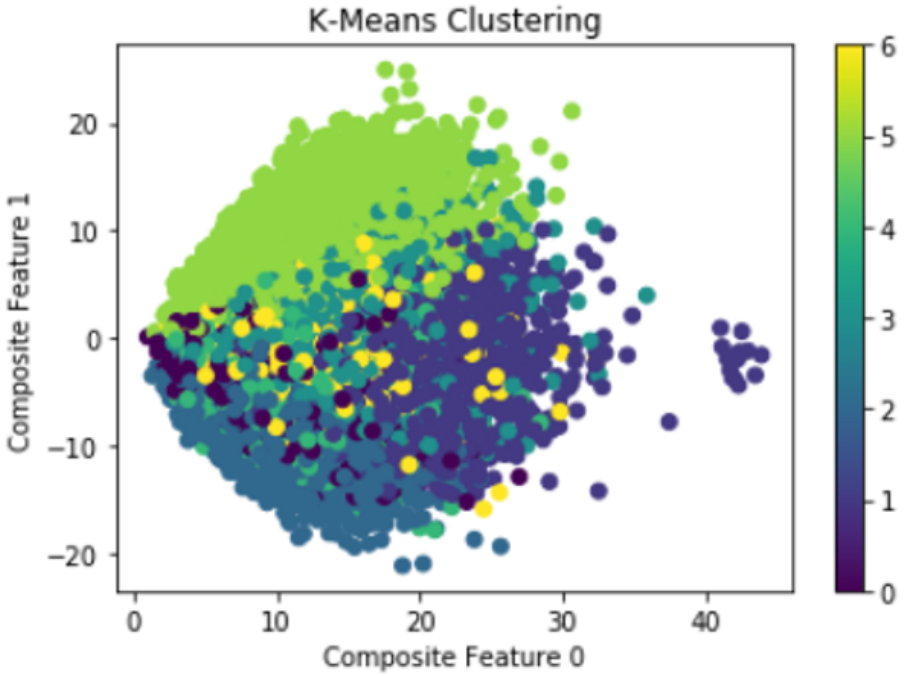
```
In [35]: cluster0_books_storted = get_cluster_favorites(0)
         cluster0_mean = get_cluster_mean(0)

         print('The cluster 0 mean is:', cluster0_mean)
         cluster0_books_storted[0:10]
```

```
The cluster 0 mean is: 3.9207207760045235
```

```
Out[35]: title
Guess How Much I Love You                                                    4.833333
Harry Potter Boxset (Harry Potter, #1-7)                                     4.811966
The Boys in the Boat: Nine Americans and Their Epic Quest for Gold at the 1936 Berlin Olympics   4.800000
Harry Potter and the Deathly Hallows (Harry Potter, #7)                      4.762994
A Court of Mist and Fury (A Court of Thorns and Roses, #2)                   4.751773
The Harry Potter Collection 1-4 (Harry Potter, #1-4)                         4.750000
Words of Radiance (The Stormlight Archive, #2)                               4.736842
Schindler's List                                                             4.727273
Acheron (Dark-Hunter #14)                                                    4.718750
Talking to Dragons (Enchanted Forest Chronicles, #4)                         4.714286
dtype: float64
```

Cluster 0 appears to be at least somewhat focused on children's books ("Guess How Much I Love You", "Harry Potter") and books related to history ("The Boys in the Boat", "Schindler's List").

I have performed clustering on a dataset that included 200 composite features. It is difficult to create a visualization that effectively illustrates all of these features. Therefore, I have selected the two top features, which played the most significant role in the clustering and created a scatterplot that illustrates the clusters across those features.



This indicates that the clustering model was effective in grouping users based on their values for composite features 0 and 1. The scatterplot also indicates that a few users had outlier values for composite feature 0.

Then I tested our testing dataset with the model and found the cluster associated with each user.

In [74]:
```
# associate each test user with a cluster
test_set_preds = clusterer_KMeans.predict(book_ratings_testing)
test_set_indices = book_ratings_testing.index
test_set_clusters = pd.DataFrame(data=test_set_preds, columns=['cluster']).set_index(test_set_indices)

test_set_clusters.head()
```

Out[74]:

user_id	cluster
27229	3
8667	5
25466	5
18077	6
9285	6

After associating each user with clusters, I have determined the cluster favourites to recommend the book to the user of a particular cluster.

```
import random
def recommend(cluster_assignments, user_id):
    user_cluster = cluster_assignments
    favorites = get_cluster_favorites(user_cluster).index
    favorites = random.choices(favorites, k=9)
    return favorites
```

5.2: Implementation of content-based filtering:

I have used cosine similarities in the book description to determine a similar book to a given book.

```
# Cosine similarities for similar book
# Content based fitering
def recommend(bookid):
    book_description = pd.read_csv("home/book_data2.csv",engine="python")
    books_tfidf = TfidfVectorizer(stop_words='english')
    book_description['book_desc'] = book_description['book_desc'].fillna('')
    book_description_matrix = books_tfidf.fit_transform(book_description['book_desc'])
    # book_description_matrix.shape
    # Cosine similarity metrics using linear_kernal of sklearnCosine similarity metrics using linear_kernal of sklearn
    cosine_similarity = linear_kernel(book_description_matrix, book_description_matrix)
    # Then, the pairwsie similarity scores of all books compared to the book passed by index,
    # And,after sorting them and getting top 5 books
    similarity_scores = list(enumerate(cosine_similarity[bookid]))
    similarity_scores = sorted(similarity_scores, key=lambda x: x[1], reverse=True)
    similarity_scores = similarity_scores[1:6]
    books_index = [i[0] for i in similarity_scores]
    print (book_description['book_title'].iloc[books_index])
    viewdata = book_description.iloc[books_index].values.tolist()
    return viewdata
```

5.3: Implementation of web app:

With the use of the above 2 models, I am taking users' interest in books and showing them recommended books, popular books and searched books.

```
    # for searched books
    # taking user interest in book
    # collaborative filtering
    if 'sbutton' in request.POST:
        print(request.POST.get('stype'))
        viewdata1 = []
        if (request.POST.get('stype') == '0'):
            title = request.POST.get('searchbox')
            viewdata = mydata[mydata['book_title'] == title ]
            viewdata = viewdata.values.tolist()
            viewdata1.append(viewdata)
            sameauth['auth'] = viewdata1
        if (request.POST.get('stype') == '1'):
            author = request.POST.get('searchbox')
            viewdata = mydata[mydata['book_author'] == author ]
            sameauth['auth'] = viewdata.values.tolist()
        return render(request, 'index.html', sameauth)
    return render(request,'index.html',sameauth)
else:
    return redirect('account/login')
```

```
# for popular books
# to show popular books on index page
# taking user interest in book
popular = mydata.sort_values(by=['book_rating'],ascending=False)
popular = popular.head(10)

if data.empty:
    print("empty")
    top=top.values.tolist()
    top1.append(top)
    sameauth["auth"] = top1
else:
    print("not empty")
    # using function called get_recommendations
    # collaborative filtering
    # To get recommendation using a weighted average of all other users using Euclidean distance score
    rec_books = get_recommendations(d, userId)
    print(rec_books)
    for i in range(10):
        book_id = rec_books[i][1]
        rbooks = mydata[mydata["book_id"] == book_id]
        rbooks = rbooks.values.tolist()
        rbooks1.append(rbooks)
    sameauth["auth"] = rbooks1
    sameauth["auth2"] = popular.values.tolist()
```

6. Testing

```
# put each cluster's sorted book list in an array to reference
cluster_favorites = [cluster0_books_storted, cluster1_books_storted, cluster2_books_storted, cluster3_books_storted, cluster4_book

# for each user, find the 10 books the reader has rated that are the top-rated books of the cluster.
# get the reader's average score for those books
for index, row in test_set_ratings.iterrows():
    user_cluster = test_set_clusters.loc[index, 'cluster']
    favorites = cluster_favorites[user_cluster].index
    user_ratings_of_favorites = []
    for book in favorites:
        if np.isnan(row[book]) == False:
            user_ratings_of_favorites.append(row[book])
        if len(user_ratings_of_favorites) >= 10:
            break
    mean_rating_for_favorites = sum(user_ratings_of_favorites) / len(user_ratings_of_favorites)
    mean_ratings_for_cluster_favorites.append(mean_rating_for_favorites)

mean_favorites_rating = sum(mean_ratings_for_cluster_favorites) / len(mean_ratings_for_cluster_favorites)

print('Mean rating for 10 random books per test user: ', mean_benchmark_rating)
print('Mean rarting for 10 books that are the cluster\'s favorites: ', mean_favorites_rating)
print('Difference between ratings: ', mean_favorites_rating-mean_benchmark_rating)
```

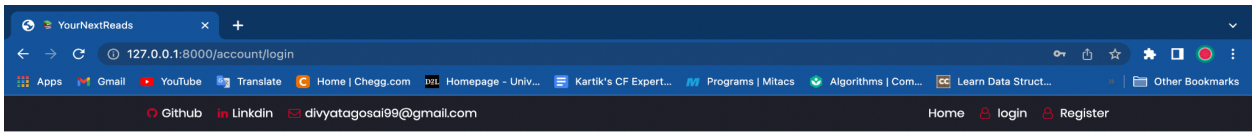
```
Mean rating for 10 random books per test user:  3.8876392510073416
Mean rarting for 10 books that are the cluster's favorites:  4.35972979378998
Difference between ratings:  0.4720905427826385
```

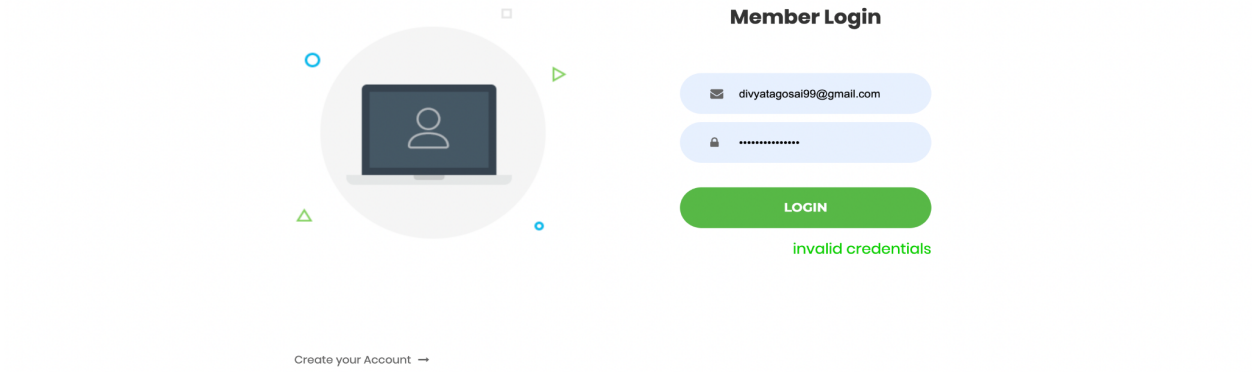

The users in the test set, on average, rated their clusters' favourite books higher than a random set of 10 books by 0.47 stars, or nearly half a star.

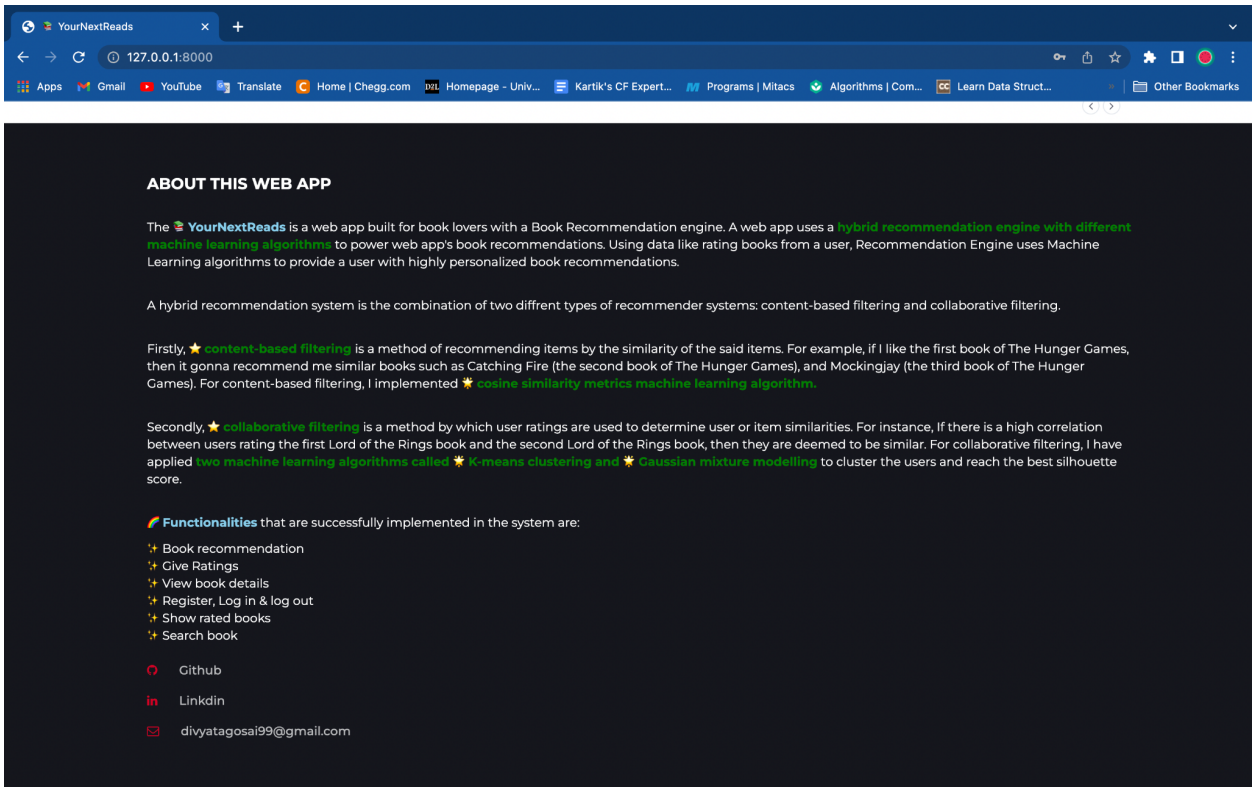
This provides some evidence that the model does a better job of recommending books than if it simply picked books at random. However, it is not clear whether the clustering itself provided this benefit, or if the benefit came from simply recommending books that were highly rated overall across clusters.

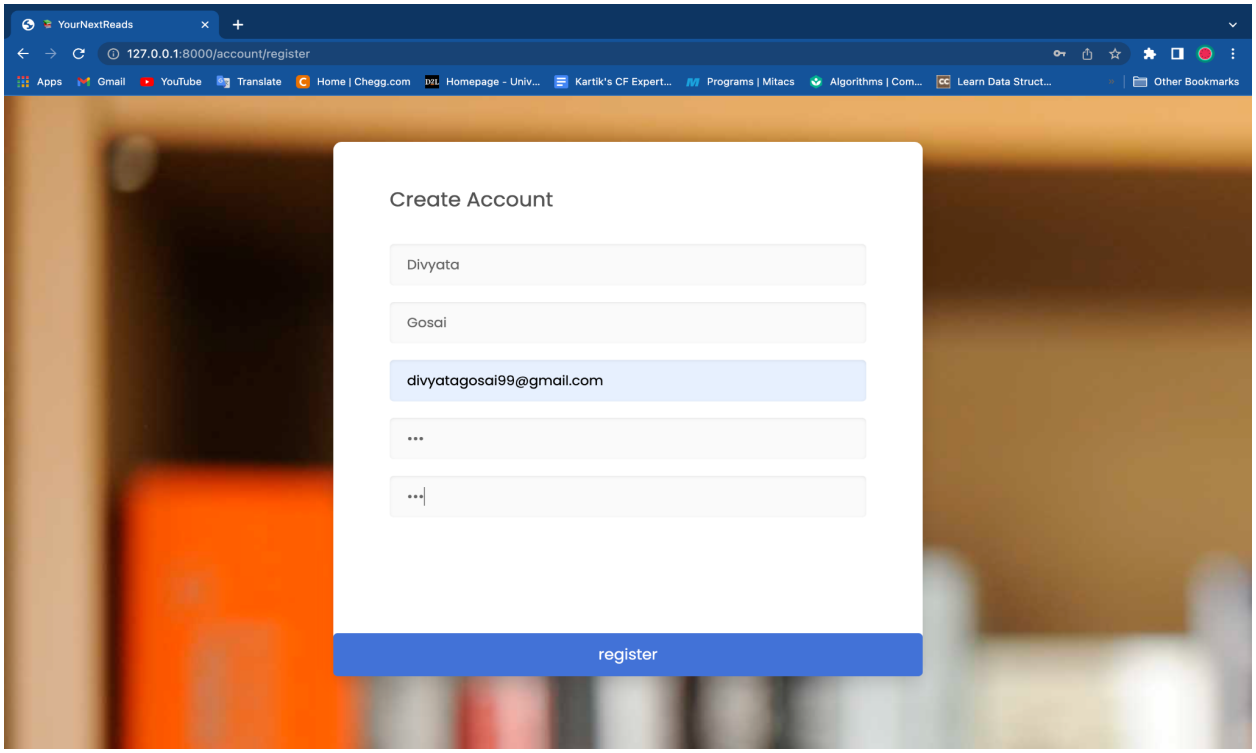
I have also performed test cases for a web app.

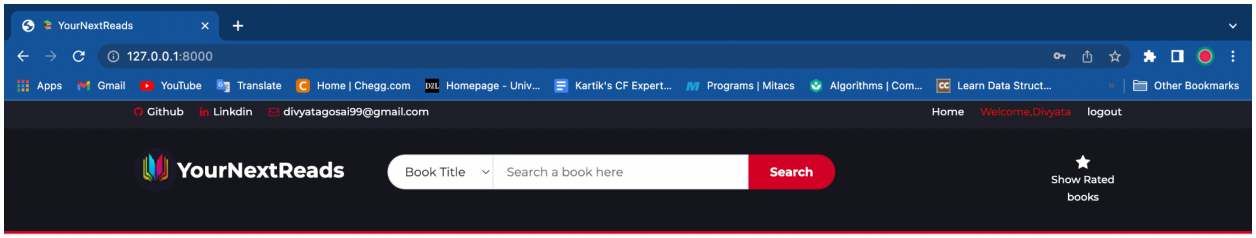
7. Screenshots



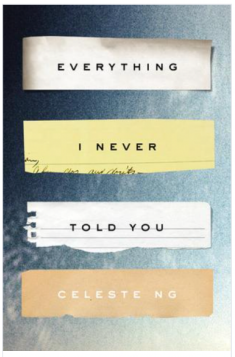






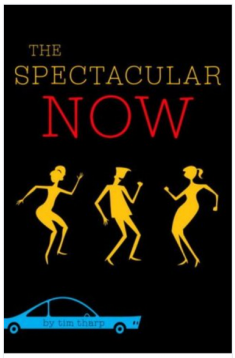


RECOMMENDED BOOK



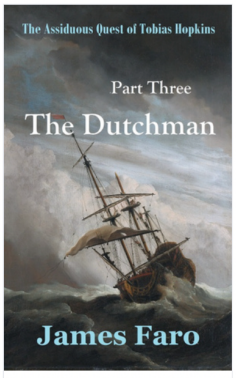
CELESTE NG

EVERYTHING I NEVER TOLD YOU



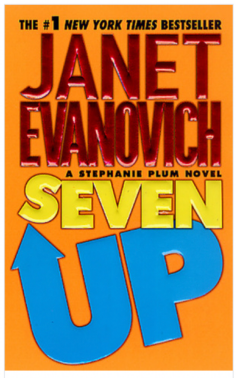
TIM THARP

THE SPECTACULAR NOW



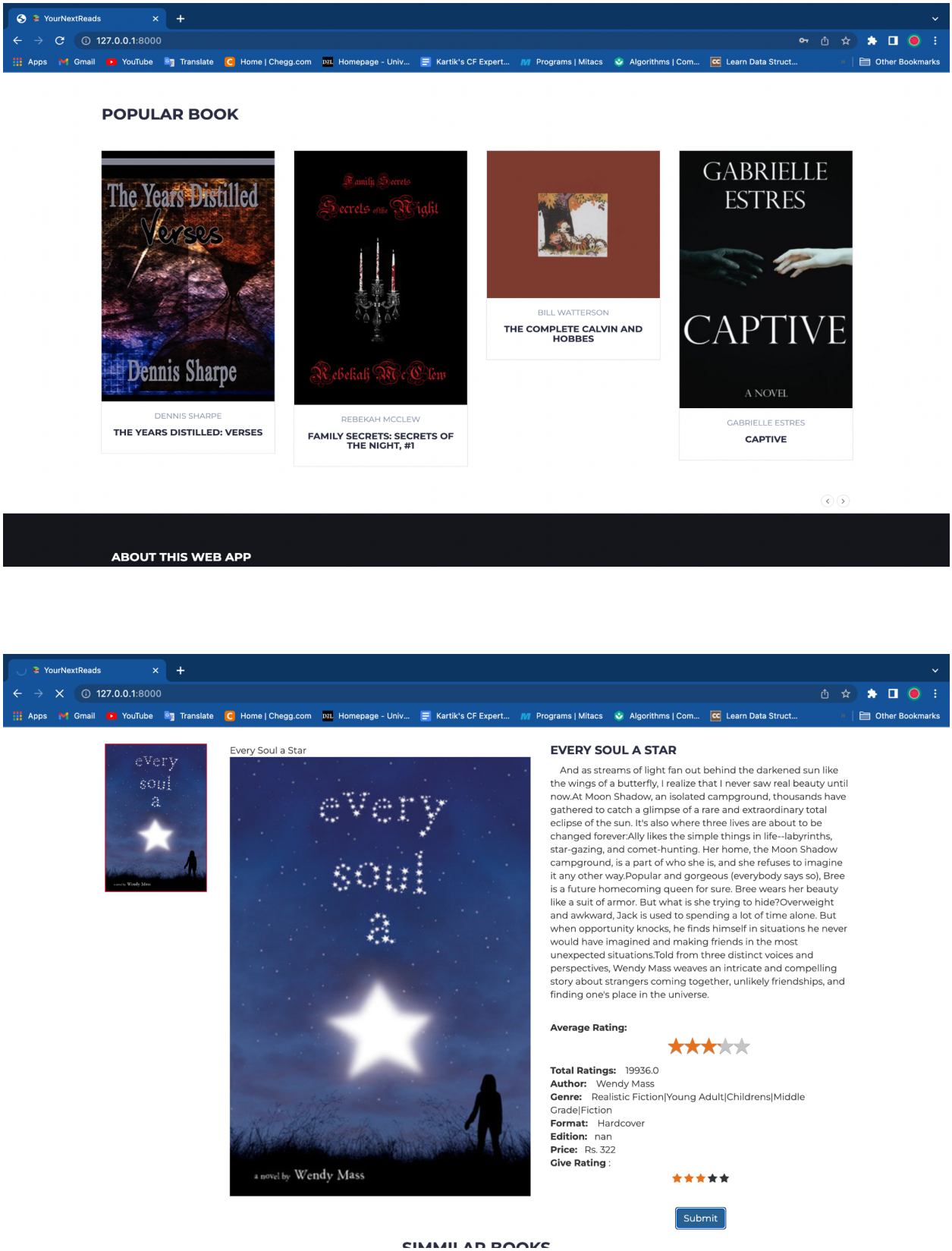
JAMES FARO

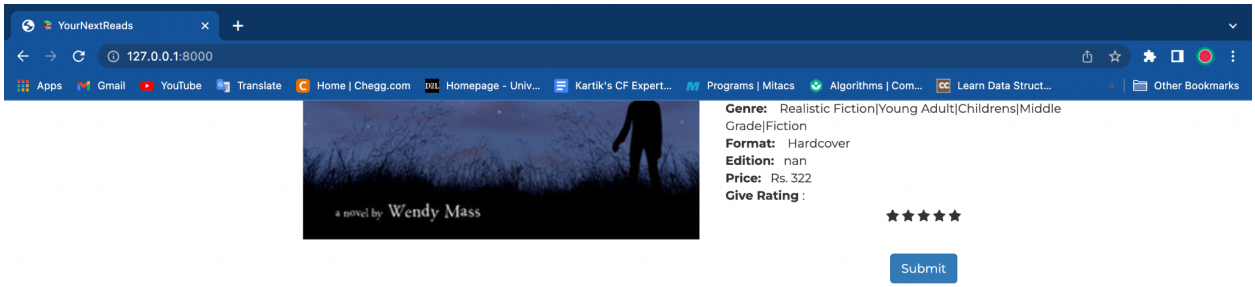
THE ASSIDUOUS QUEST OF TOBIAS HOPKINS: PART THREE: THE DUTCHMAN



JANET EVANOVICH

SEVEN UP


YourNextReads
127.0.0.1:8000
Apps
Gmail
YouTube
Translate
Home | Chegg.com
Homepage - Univ...
Kartik's CF Expert...
Programs | Mitacs
Algorithms | Com...
Learn Data Struct...
Other Bookmarks
POPULAR BOOK
DENNIS SHAREPE
THE YEARS DISTILLED: VERSES
REBEKAH MCCLEW
FAMILY SECRETS: SECRETS OF THE NIGHT, #1
BILL WATTERSON
THE COMPLETE CALVIN AND HOBBES
GABRIELLE ESTRES
CAPTIVE
ABOUT THIS WEB APP
YourNextReads
127.0.0.1:8000
Apps
Gmail
YouTube
Translate
Home | Chegg.com
Homepage - Univ...
Kartik's CF Expert...
Programs | Mitacs
Algorithms | Com...
Learn Data Struct...
Other Bookmarks
Every Soul a Star
EVERY SOUL A STAR
And as streams of light fan out behind the darkened sun like the wings of a butterfly, I realize that I never saw real beauty until now.At Moon Shadow, an isolated campground, thousands have gathered to catch a glimpse of a rare and extraordinary total eclipse of the sun. It's also where three lives are about to be changed forever:Ally likes the simple things in life--labyrinths, star-gazing, and comet-hunting. Her home, the Moon Shadow campground, is a part of who she is, and she refuses to imagine it any other way.Popular and gorgeous (everybody says so), Bree is a future homecoming queen for sure. Bree wears her beauty like a suit of armor. But what is she trying to hide?Overweight and awkward, Jack is used to spending a lot of time alone. But when opportunity knocks, he finds himself in situations he never would have imagined and making friends in the most unexpected situations.Told from three distinct voices and perspectives, Wendy Mass weaves an intricate and compelling story about strangers coming together, unlikely friendships, and finding one's place in the universe.
Average Rating:
Total Ratings: 19936.0
Author: Wendy Mass
Genre: Realistic Fiction|Young Adult|Childrens|Middle Grade|Fiction
Format: Hardcover
Edition: nan
Price: Rs. 322
Give Rating :
Submit
SIMMILAR BOOKS



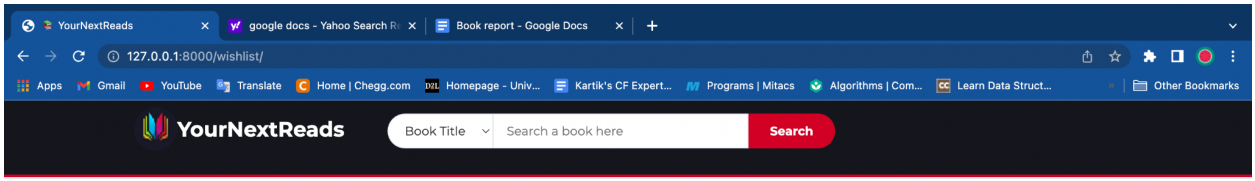
SIMMILAR BOOKS

STEPHENIE MEYER
THE TWILIGHT SAGA: THE OFFICIAL ILLUSTRATED GUIDE

STEPHENIE MEYER
THE SHORT SECOND LIFE OF BREE TANNER

STEPHENIE MEYER
THE TWILIGHT SAGA COMPLETE COLLECTION

AMANDA HOCKING
SWITCHED



BOOKS YOU HAVE RATED:

J.K. ROWLING|MARY GRANDPRÃ©
HARRY POTTER AND THE ORDER OF THE PHOENIX

WENDY MASS
EVERY SOUL A STAR

BLAKE CROUCH
DARK MATTER

8. Conclusion

The functionalities are implemented in the system after understanding all the system modules according to the requirements. Functionalities that are successfully implemented in the system are:

- Book recommendation
- Give Ratings
- View book details
- Registration, Log in & log out
- Show rated books
- Search book

After the implementation and coding of the system, comprehensive testing was performed on the system to determine the errors and possible flaws of the system.

I was able to train efficient models that had high accuracy. The model performed well on random inputs outside the dataset as well. I tried pursuing different approaches to solve the same problem,

9. Limitations and Future Enhancements

If required, the System can be easily modified in the future.

Development and launching of Mobile app and refining existing services and adding more services, System security, data security and reliability are the main features which can be done in future.

The API for the shopping and payment gateway can be added.

In the existing system, there are only some selected categories, so as an extension to the site we can add more categories as compared to an existing site. Also, we can add the admin side with some functionalities like book management, User management etc.

10. References

Following links and websites were referred to during the development of this project.

https://github.com/zygmuntz/goodbooks-10k

http://stackoverflow.com/

https://scikit-learn.org/stable/

https://www.kaggle.com/

https://towardsdatascience.com

https://docs.djangoproject.com/en/3.0/