Book Recommendation System

- **Project Overview:-**

The most trending field where machine learning is heavily deployed is in Recommendation systems. They are even been one of the most successful and widespread application of machine learning technologies in business. Recommender systems goes really handy in the scenarios where many users interact with many items. It gives user a best experience and attracts them towards the service in future scenarios. We can find large scale recommender systems in retail, video on demand, or music streaming.

Being a Bibliophile, I was always wondering what to read next. In this kind of scenarios, Recommender system comes into existence and makes our task done. So ,I thought of doing a book specific recommendation system, which recommends books of user interest based on his past rating ,given to various other books.

Link for some academic work similar to this project:-
https://www.kaggle.com/abhinav97dutt/book-recommendation-collaborative-filtering
Here in the above reference the author calculated the correlation between every pair of books. If a user rates a book high then he will be recommended with all those books which are correlated with the highly rated book.It seems to be interesting for me and thought of giving my own touch to the problem
For this project we will use Good-books dataset from kaggle.The dataset consists of following sub datasets. Ratings dataset consists of 981756 data points with 3 columns of data. Books dataset consists of 10000 books each book with 23 attributes.

Link for dataset:-
https://www.kaggle.com/zygmunt/goodbooks-10k

- **Problem statement:-**

Here the problem is all about giving related recommendations to the user based on his/her interest. We can keep track of the user interests based on the ratings that he has given to the books in the past. Now, our task is to find the books which are highly correlated to the books that are been highly rated by the user. The problem looks somewhat cumbersome because each user differs in his/her interests. So, we can't create a model that globally recommends but we have to personalize the model such that it gives user related recommendations. Here the models that we use comes under collaborative filtering. Collaborative filtering is a method to predict a rating for a user item pair based on the history of ratings given by the user and given to the item. It is based on the idea that people who agreed in their evaluation of certain items in the past are likely to agree again in the future. Most CF algorithms are based on user-item rating matrix where each row represents a user, each column an item. The entries of this matrix are ratings given by users to items. Here we will use SVD (singular value decomposition) and NMF(non-negative matrix factorization) as collaborative filtering algorithm.

In this project our aim is to recommend user interested books. So, we have to predict whether a user is interested in a particular book or not. we can achieve this by importing scikit-surprise module. In this we will use SVD and NMF model , which are considered to be optimal for recommendations .This model is able to predict how much a user is interested in a particular book in terms of estimation score. If the score is high it implies that the user is interested in that book or else not. Now, for every user we consider top 10 book_id's with highest estimation score and recommend these books with book title and author to the user.

- **Metrics:-**

The Bench mark and optimised models i.e NMF and SVD respectively which are specially designed for recommendation systems, from surprise module use RMSE(root mean squared error) and MAE(mean absolute error) as their evaluation metrics. Here we will use Root mean Square Error as the metric to determine how well our model performs.

$$\text{RMSE} = \sqrt{\frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} (r_{ui} - \hat{r}_{ui})^2}.$$

In RMSE we will first calculate the residuals by subtracting the predicted values from the actual value.Then we will square those residuals and take average of them. At last we will determine square root of the average.This RMSE score represents dispersion of the predictions from the actual value. A RMSE value of zero represents that the model has accurately predicted the value.

- **Data Exploration:-**

For this project I am using good-books dataset. I found this dataset in kaggle and it was really clean and free from Nan values. Since good-books dataset itself has many sub datasets. I considered only two of them as for my project specification. First dataset is **books.csv** ,it contains all the information regarding the books available. It include attributes like unique book_id, author, title,average_rating etc., . The next dataset is ratings.csv, it contains information regarding ratings given by the user. It specifies user_id,book_id and rating.

Since the problem is about knowing user interests we require the ratings that he has given in the past in order to refer him in the future. Ratings dataset is used to identify the user interests based on the ratings given to the books that he had read in the past. from this we can infer whether a user is interested in a particular book or not. Based on the ratings given by the user for the books,he has gone through we will attain some patterns .Now,we will use these patterns to find whether we can recommend a particular book or not.

Books dataset is majorly used for getting book details based on the book_id . So,that we can recommend the book title and author to the user. Ratings dataset consists of 981756 data points with 3 columns of data. Books dataset consists of 10000 books each book with 23 attributes,but we use only few attributes like book_id,author,title.

Lets investigate some more about the required features:-

In rating dataset we have following features:-

1) user_id→represents a user with unique user_id
2) book_id→represents a book with unique book_id
3) rating →represents the rating given by the user to a particular book

In books dataset we do consider the following important features
1)book_id → unique id represents a book
2)authors → represents the authors of the book with corresponding book_id
3)original_title → represents the title of the book
4)average_rating →represents the average rating given per book

Let’s have a look at a sample of each data set

Books Data Frame:-

	id	book_id	best_book_id	work_id	books_count	isbn	isbn13	authors	original_publication_year	original_title	...	ratings_count
0	1	2767052	2767052	2792775	272	439023483	9.780439e+12	Suzanne Collins	2008.0	The Hunger Games	...	4780653
1	2	3	3	4640799	491	439554934	9.780440e+12	J.K. Rowling, Mary GrandPré	1997.0	Harry Potter and the Philosopher's Stone	...	4602479
2	3	41865	41865	3212258	226	316015849	9.780316e+12	Stephenie Meyer	2005.0	Twilight	...	3866839
3	4	2657	2657	3275794	487	61120081	9.780061e+12	Harper Lee	1960.0	To Kill a Mockingbird	...	3198671
4	5	4671	4671	245494	1356	743273567	9.780743e+12	F. Scott Fitzgerald	1925.0	The Great Gatsby	...	2683664

5 rows × 23 columns

Ratings Data Frame:-

	book_id	user_id	rating
0	1	314	5
1	1	439	3
2	1	588	5
3	1	1169	4
4	1	1185	4

Statistical description of both the Data Frames:-

	id	book_id	best_book_id	work_id	books_count	isbn13	original_publication_year	average_rating	ratings_count
count	10000.00000	1.000000e+04	1.000000e+04	1.000000e+04	10000.000000	9.415000e+03	9979.000000	10000.000000	1.000000e+04
mean	5000.50000	5.264697e+06	5.471214e+06	8.646183e+06	75.712700	9.755044e+12	1981.987674	4.002191	5.400124e+04
std	2886.89568	7.575462e+06	7.827330e+06	1.175106e+07	170.470728	4.428619e+11	152.576665	0.254427	1.573700e+05
min	1.00000	1.000000e+00	1.000000e+00	8.700000e+01	1.000000	1.951703e+08	-1750.000000	2.470000	2.716000e+03
25%	2500.75000	4.627575e+04	4.791175e+04	1.008841e+06	23.000000	9.780316e+12	1990.000000	3.850000	1.356875e+04
50%	5000.50000	3.949655e+05	4.251235e+05	2.719524e+06	40.000000	9.780452e+12	2004.000000	4.020000	2.115550e+04
75%	7500.25000	9.382225e+06	9.636112e+06	1.451775e+07	67.000000	9.780831e+12	2011.000000	4.180000	4.105350e+04
max	10000.00000	3.328864e+07	3.553423e+07	5.639960e+07	3455.000000	9.790008e+12	2017.000000	4.820000	4.780653e+06

From the above table we can see that there are around 10k books available and most of them have rating as four as the average_rating column suggests.

	book_id	user_id	rating
count	981756.000000	981756.000000	981756.000000
mean	4943.275636	25616.759933	3.856534
std	2873.207415	15228.338826	0.983941
min	1.000000	1.000000	1.000000
25%	2457.000000	12372.000000	3.000000
50%	4921.000000	25077.000000	4.000000
75%	7414.000000	38572.000000	5.000000
max	10000.000000	53424.000000	5.000000

From the above table we can infer that there are around 10 lakh ratings available.

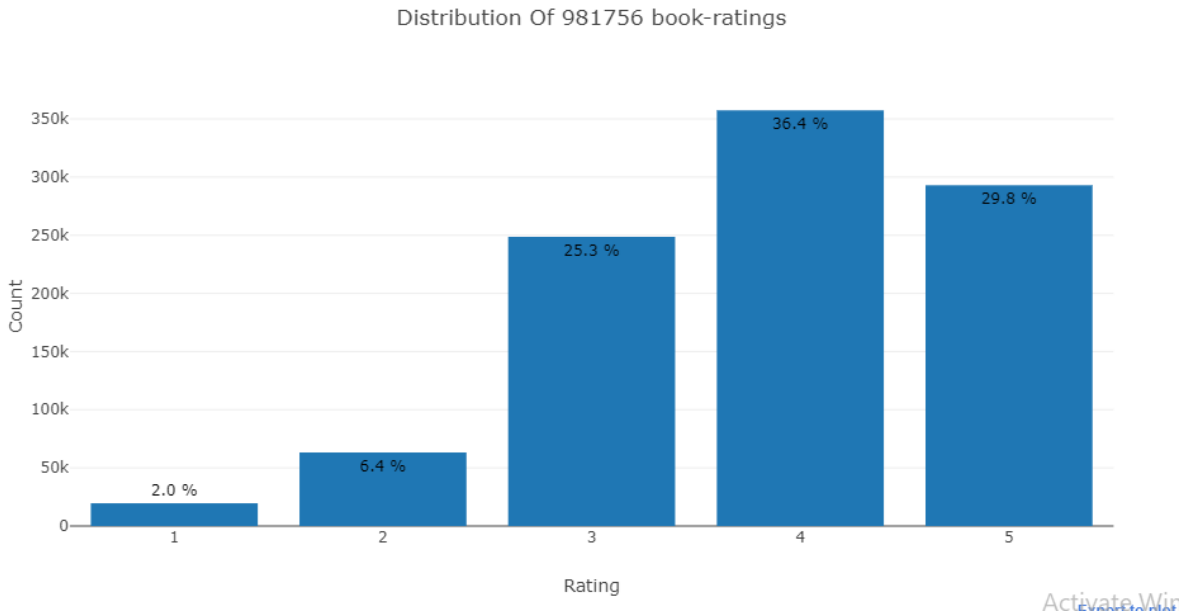
We can obtain the rating scale available by implementing the following code

```
ratings.rating.unique()
```

```
array([5, 3, 4, 1, 2])
```

- **Exploratory Visualization:-**

The most important feature of the data can be visualized as follows



Here the above bar plot visualizes the percentage of books present with each of the rating ranging from 1 to 5. we can clearly see that there are around 36% of the books which are rated as 4 and only 2% of the books are rated as 1. Around 30% of the books are rated as 5. Almost 25% of the books are rated as 3 and 6% of the books are rated as 2.

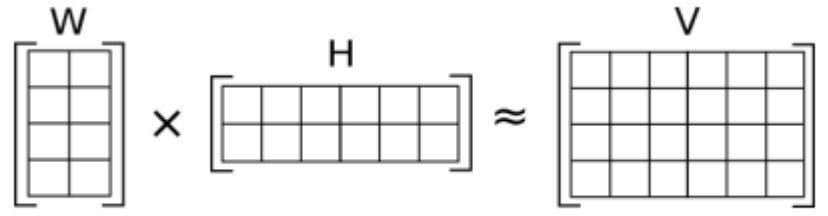
Since the dataset is clean and no Nan values found. It is free from abnormalities and there is no need to address them.

- **Algorithms and Techniques:-**

Since, the project is about giving recommendations to the user, we should be able to predict the user interests based on his past reviews. so, we will go with collaborative filtering algorithms. Here we will use two most efficient algorithms which are based on matrix factorization. They are NMF(Non-negative matrix factorization) and SVD(singular value decomposition.

1) NMF(Non-negative matrix factorization):-

This algorithm is all about Non-negative matrix factorization. In this we will first construct a matrix with users in the columns and books as the rows with values filled with corresponding ratings. Now, we construct two new matrices whose dot product is same as the original matrix.



https://en.wikipedia.org/wiki/Non-negative_matrix_factorization

The new matrices consists of non-negative values. Here a positive value represents user interest in that book and zero represents user unwillingness toward the book. In this way we will find out user interests and recommend

books of some similar kind. This algorithm has vast number of applications in image processing, text mining and collaborative filtering.

Reference:- https://medium.com/logicai/non-negative-matrix-factorization-for-recommendation-systems-985ca8d5c16c

```
nmf = NMF()

nmf.fit(trainset)

predictions_nmf = nmf.test(testset)

accuracy.rmse(predictions_nmf, verbose=True)
```

2) SVD (singular value decomposition):-

This is one of the most famous collaborative filtering algorithm based on matrix factorization. In this too we will initially constructs a matrix with users and books as columns and rows respectively with ratings as values.Then we will approximate into two matrices whose dot product is similar to the original matrix.The major difference between SVD and NMF is SVD even contains negative values in the decomposition matrices unlike NMF.

Here,we decompose the matrices in order to make them more compact and to identify hidden relationships like correlation, orthogonality and projections in the original matrix. SVD algorithm takes less computational time and gives far better results than other collaborative filtering algorithms.

```
svd = SVD()

svd.fit(trainset)

predictions_svd = svd.test(testset)

accuracy.rmse(predictions_svd, verbose=True)
```

- **Bench Mark model:-**

  I have chosen NMF as bench mark model for this problem since in this project we have to apply collaborative filtering algorithm to find out user correlation with different books based on his past reviewing experience. NMF applies non-negative matrix factorization on the data and finds out user's hidden relationship with different books and recommends some. Using NMF I got an root mean square error of 0.818, which defines it as a good model for the problem. However we can further optimize the model by using some other better model.

- **Data Pre-processing:-**

  Since the datasets ratings.csv and books.csv that we have used are clean and free from Nan values. They don't even have any outliers nor requires normalization to be done. So ,there is no need to apply pre-processing on them.

  However we will merge the two datasets to obtain the available features in a single Data Frame by implementing the following statement .

  book_data = pd.merge (ratings , book, on='book_id')

  So, It will be ease to use. We will even split the data into two sets. One for training and the other for testing. We do so in order to find the evaluation metric root mean square error for the model. However we can even skip this splitting and can train the model on the entire data. Here we don't scale or apply any transformation function like log, because the problem of recommendations doesn't require any of this kind. Even though if we do so there might be a chance of getting falsified recommendations since the values in decomposition matrices greatly vary in the range if we scale the original ratings there by loosing the original essence.

- **Implementation:-**

This project has been implemented in the following steps

- **Setting up the environment:-**In this we install all the packages required. Here I have to install scikit-surprise. I will also import all the required modules like numpy, pandas and some other modules from surprise.
- **loading the data:**-In this we load .csv files into user named DataFrames. I will load books.csv into Books DataFrame and ratings.csv into ratings DataFrame
- **exploring and preprocessing the data**:-Here we will explore the data and find various statistical measures like mean, median, mode using pandas describe() method. I even have to explore the rating scale from ratings datframe. Here the data is clean and it is free from Nan values. So, it doesn't require any preprocessing. But we will split the data for testing purpose.
- **Applying benchmark model:-**Here, we will apply non-negative matrix factorization on the data i.e NMF. Since a collaborative algorithm better suits our problem. So,we choose NMF as a bench mark model. NMF applies non-negative matrix factorization on the data and finds out user's hidden relationship with different books and recommends some.
- **Implementing optimised model**:-Now, we will use SVD from surprise. SVD is majorly used in recommendation system. here we fit the model by passing the data, which is in the of format ('user_id' , 'book_id' , 'ratings'). Based on this training data model will predict an estimation score for every user and book pair. if the score is high it implies that user is more interested in that book or else not.

svd = SVD()

svd.fit(trainset)

predictions = svd.test(testset)

- **Evaluating the model**:-Here, we evaluate our models. The metric that we use here is rmse. we will pass predictions into the function and it will determine whether user interest is correctly estimated or not using est score.

accuracy.rmse(predictions, verbose=True)

- **Giving recommendations:-** In this we will give top k recommendations to the user based on the estimation score predicted between user and the book. Then we sort all these books in the order of estimation score predicted by model in the descending manner. now, we will pick top k books and recommend the user. This type of methodology gives the user best personalised service only based on his/her interest.

  Complications encountered:-At first I thought of using KNN as bench mark model .But I encountered some problems while doing so. KNN implementation has been taking lots of CPU time and computational power. It's not even running on virtual machine,since the dataset is very large. I even tried by reducing the dataset considering only those ratings above 3. Even though the model sucks. so,I considered the models better than KNN which are NMF and SVD.

- ## Refinement:-

  Initially, the bench mark model I have used provided 0.9 root means square error .
  Later I have applied optimized model on the data and error has been reduced to 0.84.
  Then I though like there is no need of having information regarding book which are been rated as 1. So, I removed them and again applied both the algorithms
  This time for NMF I got an error of 0.818
  And for SVD the root mean square error reduced to 0.77
  So far this been the optimized model for my solution.
  Here in this project I didn't tuned any parameters due to the computational power constraint. But In matrix factorization algorithms we can tune the number of latent factors via cross validation. These latent factors are nothing but the features in the lower dimension latent space projected from user-item interaction matrix. Their number determine the amount of abstract information desired to be stored in the low dimension space.

Reference:- https://towardsdatascience.com/prototyping-a-recommender-system-step-by-step-part-2-alternating-least-square-als-matrix-4a76c58714a1

- **Model evaluation and validation:-**

  Here the final optimized model is SVD(singular value decomposition).In this we will pass user_id and book_id to the predict method in SVD class and it will produce the estimation score. Based on this score we recommend books to the user.
  In order to check the Robustness of the model, first we have to add new user data with their corresponding ratings towards some books. Then only we can recommend books to the user since our recommendations are solely based on users past reviewing experience as we are using collaborative filtering.

Now ,lets look at the book recommendations made by the model and justify the model using our intuition

Lets consider a user_id 315

He has rated the following books

```
books_data[books_data.user_id== 315]
```

	book_id	user_id	rating	authors	original_title
21497	2915	315	5	Gary Paulsen	The River
51000	5954	315	5	Hermann Hesse, Ursule Molinaro	Narziß und Goldmund

He rated the two books as 5.The first book ‘The River’ is a novel which describes the adventures made by a man in order to survive in wilderliness.
The second book ‘Narziss und Goldmund’ is a novel about two different characters Narziss, who is a scholar who searches for meaning in abstractions, whereas Goldmund is a sensualist who always looks for worldly pleasures.

Now lets look at the recommendations made by the SVD model

```
recommendation(315,svd)
```

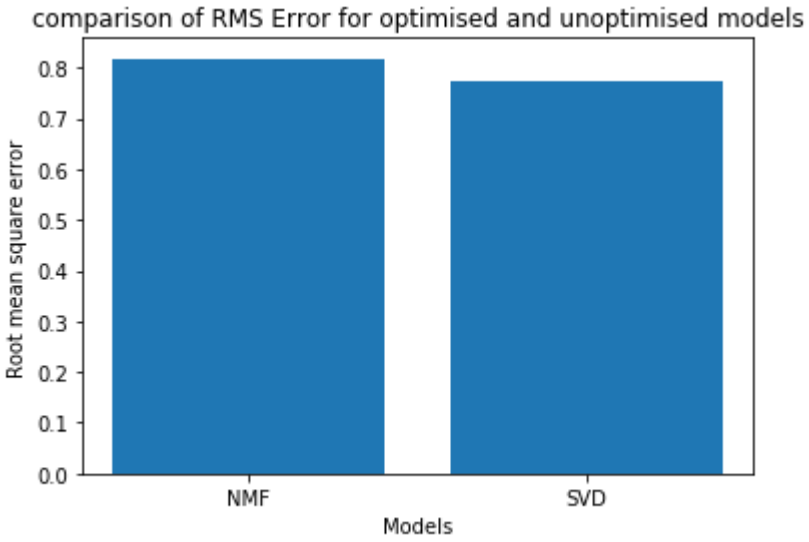
```
                                    authors  \
1985                            John Grisham
3507  Marcel Proust, Simon Vance, Lydia Davis
2670                            Maeve Binchy
155                             Emily Giffin
8703  Leo Tolstoy, Louise Maude, Aylmer Maude
3790                            Michel Faber
2740               Roald Dahl, Quentin Blake
4166                                Garth Nix
1552                     Laurell K. Hamilton
156                      Dr. Seuss, לאה נאור

                                      original_title
1985                                 The Innocent Man
3507                            Du côté de chez Swann
2670                                Circle of Friends
155                                Something Borrowed
8703                                       Воскресение
3790                  The Crimson Petal and the White
2740                 Danny: The Champion of the World
4166                                     Mister Monday
1552  Circus of the Damned (Anita Blake, Vampire Hun...
156                                Green Eggs and Ham
```

Lets have a look at the first recommendation which is “The Innocent Man” by John Grisham. In this the author cites some true crime stories where an innocent has been accused so easily by creating false evidences.This can be a good recommendation to the user 315,since he was interested in some adventurous and thrilling novels. This book can better suit him. With this we can infer that the model works fine in giving recommendations to the user.

- **Justification:-**

  In this section we will make a comparison between the final results and the bench mark results using root mean square error as the metric. For bench mark model we got an error of 0.82
  Later in the final result this error has been reduced to 0.77
  Based on the above explanation done for robustness, we can conclude that our model works with real world problems.

- **Free-Form Visualization:-**



From the above drawn graph, it is clear that NMF has high root mean square error which is 0.818 and SVD has comparitively low root mean square error as 0.77. So, we will consider SVD (singular value decomposition) as the optimized model.

- **Reflection:-**

For this project, The end to end problem solution is to train the SVD model using the given data which is in the format of (‘user_id’, ’book_id’, ’rating’).Then we can get recommendations for a particular user based on the estimation score provided by the algorithm for each every book. After getting the estimation score we will sort the books in the descending order of estimation scores and retrieve top 10 records to recommend the given user. The most interesting part of this algorithm is getting personalised recommendations based on user interests. Since, There is no predefined code provided for getting recommendations directly. we should write a user defined function to give the recommendations by comparing the estimation scores provided by the algorithm.

- **Improvement:-**

  Here in this project, we were only recommending the user solely based on his past reviewing experience. So, we cannot provide recommendations for a new user. But we can improve this model such that it can even recommend the newly joined user .It can be done by having additional information regarding users. So, when a new user comes we will recommend him based on the users who are having features similar to him.

  This kind of approach comes under hybrid recommenders which is a combination of collaborative filtering and content based recommenders. Here we can implement hybrid recommenders by combining KNN from content based recommenders and SVD from collaborative filtering, In which each of these models complement each other by providing solutions in different kind of scenarios.