CAPSTONE PROPOSAL

- ## Domain background:-

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

- **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. The bench mark model that I used comes under unsupervised learning, where we have to form clusters of related books. The optimized model 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) as collaborative filtering algorithm.

- **Datasets and inputs:-**

  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.

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

  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.

- **Solution statement:-**

  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 model ,which is considered to be an optimal model for recommendation .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.

- **Benchmark model:-**

  First I thought of recommendations as a case of clustering, since we need to find out the similar books to recommend the user.I used k-means clustering for doing this job.I trained the model on ratings dataset and found that books are formed into clusters mostly based on their ratings.This model is not so perfect in giving recommendations since it globalized the results rather than giving a personilized experience to the user.

- **Evaluation metrics:-**

For benchmark model that is by using kmeans clustering,I used silhouette distance as a metric The silhouette value is a measure of how similar an object is to its own cluster compared to other clusters . The silhouette ranges from −1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighbouring clusters.

$$s(i) = b(i)-a(i) / \max\{a(i),b(i)\}$$

Where a(i) represents the average distance between i and all other data within the same cluster. b(i) represents the smallest average distance of i to all points in any other cluster, of which i is not a member.

Even though this model works fine in forming clusters, its not so efficient in giving optimal recommendations to the user. So, we go for other optimised models which is specially designed for recommendation systems, which is

SVD, from surprise module. Here we use RMSE(root mean squared error) as an evaluation metric .

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

- **PROJECT DESIGN:**

  This project flows into 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.

  - **Applying benchmark model:-**Here,we will apply Kmeans clustering and find the corresponding evaluation metrics. The K-means algorithm starts with a first group of randomly selected centroids, which are used as the beginning points for every cluster, and then performs iterative (repetitive) calculations to optimize the positions of the centroids until there is no change in the position of the centroid.

  - **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 optimised model. 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.