MOVIE RECOMMENDATION SYSTEM
README

I. Problem Statement

- Given a user, The recommendation system should be capable of recommending some movies/tv shows based upon his interests.

II. Dataset

- The dataset was collected from kaggle. There are a total of **4500 movies** and **470758 users** in the existing system (dataset) . The data format is of the form as shown below.
  <movie_id_1> :
  <user_id_1> , <rating of user_id_1 for movie_id_1> , <rated_date>
  <user_id_2> , <rating of user_id_2 for movie_id_1> , <rated_date>
  ……
- The dataset has been arranged in the dataframe of format as shown below (sorted by date column since the data is temporal format). The model would be more robust if we train the model on past data and test on future data,

	User_id	Movie_id	Rating	Date
9056171	510180	1798	5	1999-11-11
14892677	510180	2866	3	1999-11-11
19585852	510180	3730	4	1999-11-11
20393918	510180	3870	2	1999-11-11
6901473	510180	1367	5	1999-11-11

III. Mapping To a Machine Learning Problem

The given problem can be solved in two ways,

- The first way is, posing it as a **regression problem** where we will be given a set of features pertaining to a particular movie-id and user-id we need to predict the rating that can be given by the user to that particular movie. So when we fill out all the NA values with predicted ratings then we can recommend the user the movies that can be possibly highly rated by him.
- The other approach is to use **user-user and item-item** similarity approaches.
- The Third approach is to perform **singular value decomposition (SVD)**. Where in which we decompose the single matrix containing User_id, Movie_id, Rating into 2 matrices which are orthogonal and a diagonal matrix. A Matrix “X” is said to be orthogonal iff $\mathbf{X.X^T = X^TX = I}$ **(identity matrix).**
- **The decomposition would look like,** $A = U \sum V^T$ **, U and V are orthogonal Matrices. ‘∑’ is the diagonal matrix.**

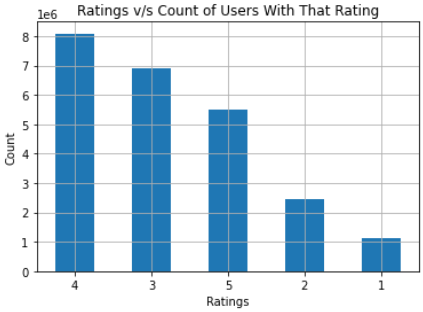
MOVIE RECOMMENDATION SYSTEM
README

- See the below youtube videos for clear explanation,
  Machine Learning | Singular Value Decomposition ,
  Singular Value Decomposition (the SVD)
  If we perform, A^TA then the RHS would become as $A^TA = (V\sum^T\mathbf{U^T})(\mathbf{U}\sum V^T)$; Since U is an orthogonal matrix $UU^T = U^TU = I$. Therefore, the above equation becomes, $(V\sum^T\sum V^T)$. When we have an equation of this form implicitly V becomes the eigenvector matrix of A^TA. So, $\mathbf{V = A^TA.}$ The matrix E will be a diagonal matrix of singular values which is the square root of eigenvalues. Similarly, AA^T yields result as $(U\sum V^T)(V\sum^T\mathbf{U^T})$**.** This gives final result as $U\sum\sum^T U^T$**.** When we have a positive definite (det > 0) symmetric matrix both eigen and singular values would be the same. **Therefore, $\mathbf{U = AA^T}$.** See the numerical example below,
-

IV. EDA (Exploratory Data Analysis)

The observations and analysis on the data are as follows (No empty or Null values in Dataset),

- **Count of Movies and Users**
  ```
  Number of Users :  470758
  Number of Movies :  4499
  ```
- **Overall Average User Rating**
  ```
  3.678843334175959
  ```
- **Overall Movie Average Rating**
  ```
  3.22194467751688
  ```
- **Rating Statistics**
  ```
  count    2.405376e+07
  mean     3.599634e+00
  std      1.086118e+00
  min      1.000000e+00
  25%      3.000000e+00
  50%      4.000000e+00
  75%      4.000000e+00
  max      5.000000e+00
  ```
- **Count of Users with That Rating**

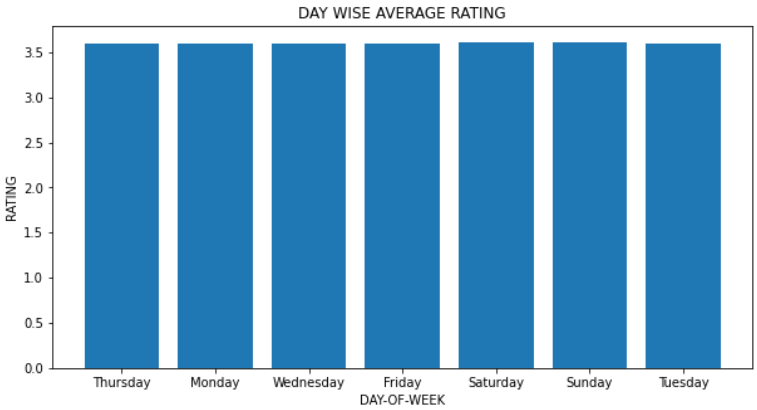


INFERENCE

It seems that most of the movies/shows available in OTT are highly rated (ratings >= 3). It can be inferred that most of the users are satisfied with the content available in OTT.

MOVIE RECOMMENDATION SYSTEM
README

- **Average Rating Day wise**



INFERENCE

Seems that the rating of the user cannot be predicted with the feature "Day of the week" as the average rating for every day is almost similar

V. Train-Test Split

- The given data is split into train and test sets. The train size is 90% of the entire data whereas test data is 10%
- Shape of Training Data : `(21648387, 3)`
  Shape of Test Data : `(2405377, 3)`
  New Users present in the Test Data : `36337`
  New Movies present in the Test Data : `7`
- The cold start problem is severe with users compared to movies.

VI. Converting Data into Sparse Matrix

- Using the csr_matrix library of scipy.sparse the given matrix is converted into rows of user-ids and columns of movie-ids.
- The cells contain the rating data.

VII. User-User Similarity

- In the user-user similarity approach the cosine-similarity between given user vector (user under consideration) and remaining all the user vectors. The obtained results would be sorted in decreasing order of cosine similarity scores. Since, more similar the users, higher will be the cosine-similarity between them.
- Then top-k users would be picked up and movies watched by them would be recommended to the user under consideration.
- This is the simplest approach but computationally expensive.

MOVIE RECOMMENDATION SYSTEM
README

VIII. Movie-Movie (Item-Item) Similarity

- For a given user, consider that he/she has watched a movie m1. Now we consider the vector of movie m1 and perform cosine similarity with remaining all the movie vectors and extract more similar movies to the movie m1. Say the obtained similar movies are m2,m3,m4,m5.
- Now the user under consideration would be recommended with the movies m2,m3,m4,m5.
- Say the user-movie matrix(A) is of dimension UxM. So when we perform $\mathbf{A^T*A}$ gives the resultant matrix with dimensions **MxM**. Which gives the similarity scores between every pair of movies. Therefore, for extracting similar movies of a **movie $\mathbf{m_i}$** we can access the i^{th} row of that similarity matrix and can extract top-k column indexes with high cosine-similarity values and recommend them to that user under consideration.
- This is the simplest solution and computationally feasible too. The Movie-Movie similarity for a sample film has been computed and the results obtained were as follows,

Movies Similar to the given movie 'Isle of Man TT 2004 Review' are..

	Movie_id	Release_Year	Movie_Name
456	457	2004.0	Kill Bill: Vol. 2
1219	1220	2004.0	Man on Fire
1904	1905	2003.0	Pirates of the Caribbean: The Curse of the Bla...
2371	2372	2004.0	The Bourne Supremacy
3623	3624	2003.0	The Last Samurai

IX. Hand-Crafted Features

- The other way of solving this problem is to pose it as a regression problem and fill up all the zeros/NA's with the predicted ratings . So for doing so we need to build the feature vectors (for users and movies) and train them with their respective ground truths.
- Total of 8 features were considered and they are as follows,
  - Global Average of ratings present in the training set.
  - User average Rating
  - Movie average Rating
  - Given a user-id (U_i) and a movie-id (M_i) the Non-Zero(or 'NA') Ratings of top-5 similar movies to that of movie-id (M_i) watched by user with user-id (U_i).

X. Hand-Crafted Features with XGBoost

- The data after adding hand-crafted features looks as follows,

MOVIE RECOMMENDATION SYSTEM
README

	User_id	Movie_id	Rating	Gavg	Average_user_rating	Average_movie_rating	sim_movie_1_rating	sim_movie_2_rating	sim_movie_3_rating	sim_movie_4_rating	sim_movie_5_rating
0	510180	1798	5	3.589	3.366	3.957	3.0	5.0	4.0	4.0	4.0
1	510180	2866	3	3.589	3.366	3.414	5.0	3.0	4.0	4.0	4.0
2	510180	3730	4	3.589	3.366	3.955	4.0	5.0	3.0	5.0	4.0
3	510180	3870	2	3.589	3.366	3.179	2.0	4.0	5.0	4.0	3.0
4	510180	1367	5	3.589	3.366	3.588	5.0	4.0	5.0	4.0	3.0

- For a user if the number of non-zero/NA ratings found are less than 5 (we will traverse until top-30 similar movies) then user average rating will be used for 5-k times (where k (<5) is number of non-zero ratings found in top-30 similar movies. As we go on traversing the similarity between movies also decreases. So it would be better to take the average rating of that user to build a 5-dim vector.

XI. SVD with 25 & 100 Factors

- SVD with 25 factors (dimensions) and 100 factors has been implemented. The SVD function of the **surprise module** takes only the **user-id, movie-id, rating** as **inputs**.
- The important hyper-parameters here are factors, learning rate and number of epochs for which the algorithm has to be run.
- The loss function of SVD is as follows,

$$Min(x,\ y,\ b_i,\ b_u) \sum_{(u,\ i)\in K} (r_{ui} - x_i^T y_u - \mu - b_i - b_u)^2 + \lambda(||x_i||^2 + ||y_u||^2 + b_i^2 + b_u^2)$$

- The predicted rating would be **($x_i^T y_u + \mu + b_i + b_u$).** The first term is similar to squared-loss term and the second is a regularization term. **This is only performed for non-zero values in the dataset.**
- The representation of symbols are as follows,
  r_{ui} : original rating
  x_i^T : User vector
  y_u : Item (movie vector)
  μ : Global average rating
  b_i : Bias of item (here average rating of movie mi)
  b_u : Bias of user (here average rating of user ui)

XII. Handling Cold Start Problem

- When we have a new user in the test data (who watched only one movie and has no historical data). The similar movies vector and average rating of that user will be all zeroes.
- When we have a new movie in the system (which has no historical data of ratings and may also not be similar to any other movie) then the similar movies vector will be a vector of that user's average ratings.
- If a user watched less than 5 similar movies to that of a given movie then remaining values will be filled by that user's average rating.

MOVIE RECOMMENDATION SYSTEM
README

XIII. Final Results

Technique Used	RMSE ON Test Data
XgBoost with Hand-Crafted Features	1.6799188772496727
SVD with 25 Factors	0.9760240635034816
SVD with 100 Factors	0.9756567860204827

It can be seen that SVD works better than hand crafted features. Probably by adding additional hand crafted features such as similar user ratings for a movie mi would yield better results in the case of just using hand-crafted features.

XIV. Extensions/Future Work

- Due to memory constraints (<12Gb of RAM) we could not try out some different methodologies such as,
    - Including 5 similar user ratings (users similar to that of user U_i who is under consideration) for movie M_i which is under consideration.
    - Including the predictions of SVD as another set of features to the existing feature set and training xgboost on the newly obtained features.
    - Training other ensemble techniques such as Random Forest, Gradient Boosting, etc.
    - Training SVD with factor count >100.