

Building recommendation system to predict user preferences for entertainment product

Anurag

Dayalbagh Educational Institute, Agra

Ayushi Tripathi

Dayalbagh Educational Institute, Agra

Sugandha Tanwar

Dayalbagh Educational Institute, Agra

Satyam Sharma

Dayalbagh Educational Institute, Agra

Varun Parmar

Dayalbagh Educational Institute, Agra

1 ABSTRACT

Recommender engines (REs) also known as recommender systems are software tools and techniques providing suggestions to a user. The suggestions provided are aimed at supporting their users in various decision-making processes such as what items to buy, what music to listen, what profiles to browse, or what news to read. The vast amount of data available on the Internet has led to the development of recommendation systems. This project proposes the use of soft computing techniques to develop recommendation systems. It addresses the limitations of current algorithms used to implement recommendation systems, evaluation of experimental results, and conclusion. This report provides a detailed summary of the project “Buildings recommendation system to predict user preferences for entertainment product”. The report includes a description of the topic, system architecture, and provides a detailed description of the work done till point. Included in the report are the detailed descriptions of the work done: snapshots of the implementations, various approaches, and tools used so far.

2 INTRODUCTION

A recommendation system is a type of information filtering system which attempts to predict the preferences of a user, and make suggestions based on these preferences. There are a wide variety of applications for recommendation systems. These have become increasingly popular over the last few years and are now utilized in most online platforms that we use. The content of such platforms varies from movies, music, books and videos, to friends and stories on social media platforms, to products on e-commerce websites, to people on professional and dating websites, to search results returned on Google. Often, these systems are able to collect information about a user’s choices, and can use this information to improve their suggestions in the future. Two most ubiquitous types of personalized recommendation systems that we used in our project are Content-Based and Collaborative Filtering. The Content-Based Recommender relies on the similarity of the items being recommended. The basic idea is that if you like an item, then you will also like a “similar” item. It generally works well when it's easy to determine the context/properties of each item. A content based recommender works with data that the user provides, either explicitly movie ratings for the MovieLens

dataset. Based on that data, a user profile is generated, which is then used to make suggestions to the user. As the user provides more inputs or takes actions on the recommendations, the engine becomes more and more accurate. The Collaborative Filtering Recommender is entirely based on the past behaviour and not on the context. More specifically, it is based on the similarity in preferences, tastes and choices of two users. It analyses how similar the tastes of one user is to another and makes recommendations on the basis of that. For instance, if user A likes movies 1, 2, 3 and user B likes movies 2,3,4, then they have similar interests and A should like movie 4 and B should like movie 1. This makes it one of the most commonly used algorithm as it is not dependent on any additional information.

In general, collaborative filtering is the workhorse of recommender engines. The algorithm has a very interesting property of being able to do feature learning on its own, which means that it can start to learn for itself what features to use. It can be divided into Memory-Based Collaborative Filtering and Model-Based Collaborative filtering

3 TECHNOLOGY USED

Jupyter Notebook (Google Colab) – Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Also include data cleaning and transformation, Numerical simulation, statistical modelling data visualization, machine learning and much more.

3.1 LIBRARIES USED

Pandas – Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the python programming language.

NumPy – NumPy is a library for the python programming language, adding support for large multi-dimensional arrays and matrices, along with a large collection of high- level mathematical functions to operate on these arrays.

Scikit – Scikit-learn is probably the most useful library for machine learning in python. The sklearn library contains a lot of efficient tools for machine learning and statistical modelling, regression, clustering and dimensionality reduction.

Scipy – Scipy is a free and open source python library used for scientific computing. It contains modules for optimisation, linear algebra, integration, interpolation etc.

Surprise library – Surprise is an easy-to-use python scikit for recommender systems.

Matplotlib – Matplotlib is a plotting library which we have used in our project for data visualisation. It is a library for python programming language and its numerical extension NumPy.

Seaborn – Seaborn is a python data visualisation library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Word Cloud – Word Cloud is a data visualisation technique used for representing text data in which the size of each word indicates its frequency or important significant textual data points can be highlighted using a word cloud. Word Cloud is mostly used for analysing data from social network websites.

4 LITERATURE SURVEY

As we know that with an increasing no of online companies which are utilizing recommendation system to increase user interaction and enrich shopping potential. Use cases of recommendation systems have been expanding rapidly across any aspects of e-Commerce and online media over the last 4-5 years and we expect this trend to continue.

Recommendation systems (often called “recommendation engines”) have the potential to change the way websites communicate with users and to allow companies to maximize their ROI based on the information they can gather on each customer’s preferences.

5 DATASET AND FEATURE DESCRIPTION

5.1 Dataset Description

The MovieLens dataset consists of these files contain 1,000,209 anonymous ratings of approximately 3,883 movies made by 6,040 MovieLens users who joined MovieLens in 2000.

The MovieLens dataset contains 3 files i.e. Ratings.csv, Movies.csv, User.csv

Movies Dataset:

- *MovieID*: Each movie has a Unique ID
- *Title*: Titles (Name) of Movies
- *Genres*: Category of Movies (like: Drama, Comedy, Action etc.)
- *Titles* are identical to titles provided by the IMDB (including year of release)
- Shape: (3883,3)

User Dataset:

- *UserID*: Each user who rate the movies has a unique ID
- *Age*: Age of the user
- *Gender*: User's Gender (M or F)
- *Occupation*: It ranges (0-20) each number has represented Occupation of a particular of a User
 - 0: "Other" Or Not Specified
 - 1: "Academic/Educator"
 - 2: "Artist"
 - 3: "Clerical/Admin"
 - 4: "Co
 - 5: "Customer Service
 - 6: "Doctor/Health Care"
 - 7: "Executive/Managerial"

- 8: "Farmer"
- 9: "Homemaker"
- 10: "K-12 Student"
- 11: "Lawyer"
- 12: "Programmer"
- 13: "Retired"
- 14: "Sales/Marketing"
- 15: "Scientist"
- 16: "Self-Employed"
- 17: "Technician/Engineer "
- 18: "Tradesman/Craftsman"
- 19: "Unemployed"
- 20: "Writer"
- *Zip-code*: Area or Zip code of a user
- Shape: (6040,5)

Rating Dataset:

- *UserID*: Ranges (1 to 6040)
- *MovieID*: Range between 1 and 3952
- *Ratings*: 1-5-star rating by user
- *Timestamp*: Represented the time mentioned in seconds when the rating was provided
- Shape: (1000209,4)

6 EXPLORATORY DATA ANALYSIS

Exploratory data analysis is an approach to analysing the datasets to summarise their main characteristics, often with visual methods. A statistical test can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modelling or hypothesis testing task.

6.1 Word Cloud:

Word Cloud is used for representing text data in which the size of each word indicates its frequency or importance.

significant textual data points can be highlighted. Here, word cloud shows ‘Genres’ as word comedy size is larger, it indicates that it’s the most frequent word.

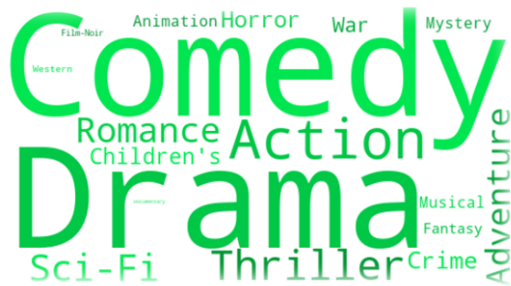


Figure 1- Word Cloud

6.2 Univariate Analysis

- *Numerical Features* – Ratings, UserID, MovieID, Age, Occupation, Timestamp
- *Categorical Features* – Title, Gender, Genre, Gender

6.2.1 Plot Histogram:

- Frequency is highest for rating 4 and least for rating 1.
- Frequency is highest for age group (22-24) and least for age group below 10 i.e. (0-10).
- Frequency for 0 occupation is highest i.e. for (‘others’ or ‘not-specified’) and least for occupation (7.5 to 10).

6.2.2 Occupation Distribution

By using bar plot visualize that users belong to which occupation are more and which are least.

Inferences:

- Highest frequency is for Occupation 4 i.e. (‘col’).
- Lowest frequency is for Occupation 8 i.e. (‘farmer’).
- Gender Distribution – Visualize with the help of histogram that Frequency of number of users is high for male as compare to female.

6.3 Bivariate Analysis:

6.3.1 Seaborn Facet Grid

Seaborn Facet Grid takes the given data onto multiple axes arrayed in a grid of rows and columns that correspond to levels of variables in the dataset.

Seaborn Facet Grid between gender and Rating

Inferences:

- Most of the users are male who watched or rate the movies at maximum.
- The users of age group 25-34 are most interested about giving rating. It means youngsters are more interested in giving ratings.
- The users of age group under 18 and 56+ are least concerned about giving rating.

Top 10 Most Rated

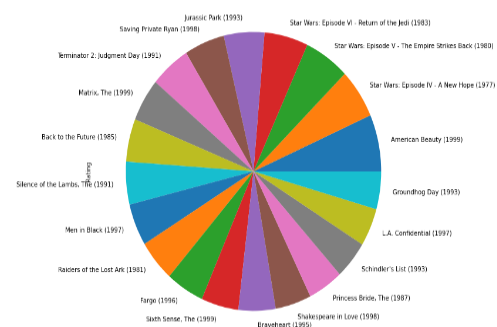


Figure 2- Top 10 most rated movies

- Rating distribution across all the Occupation

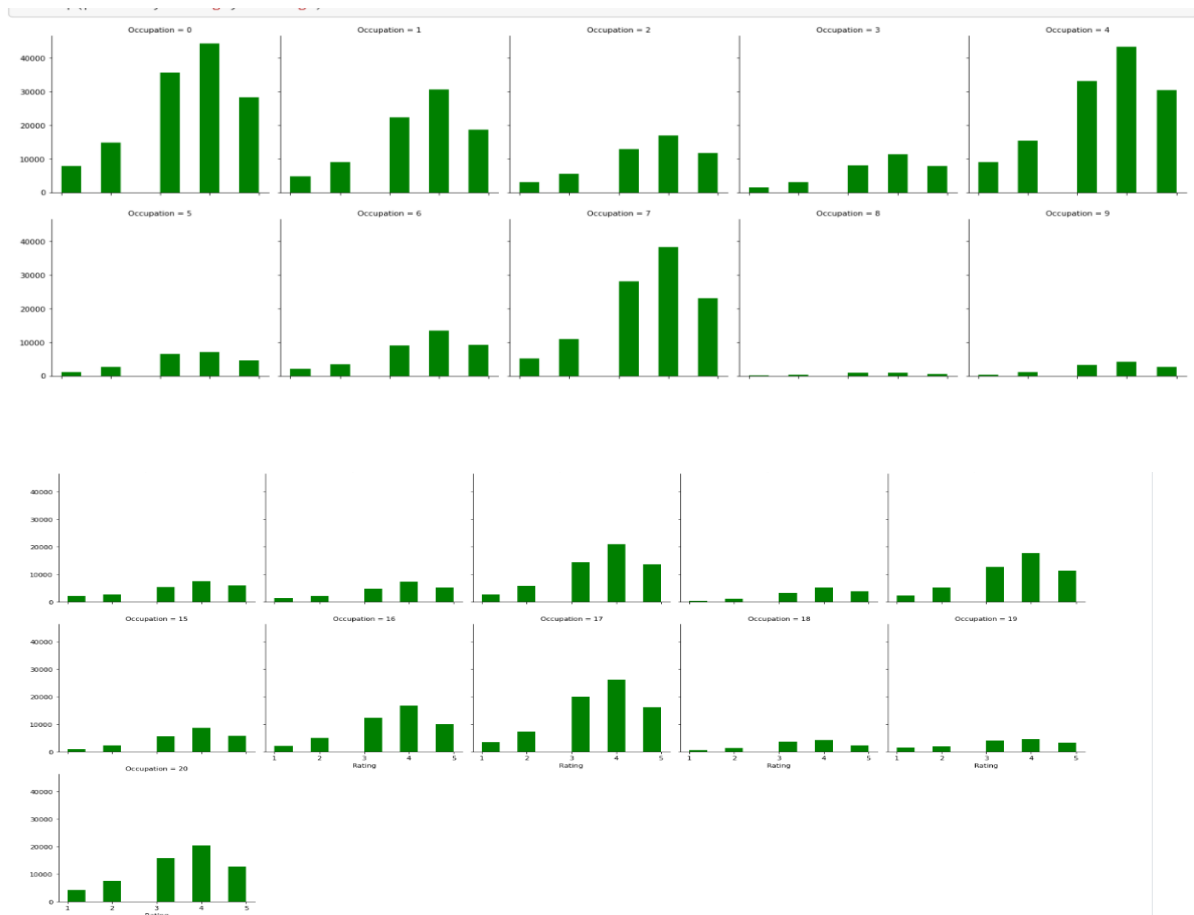


Figure 3- distribution across all the Occupation

- Rating distribution across all Genres

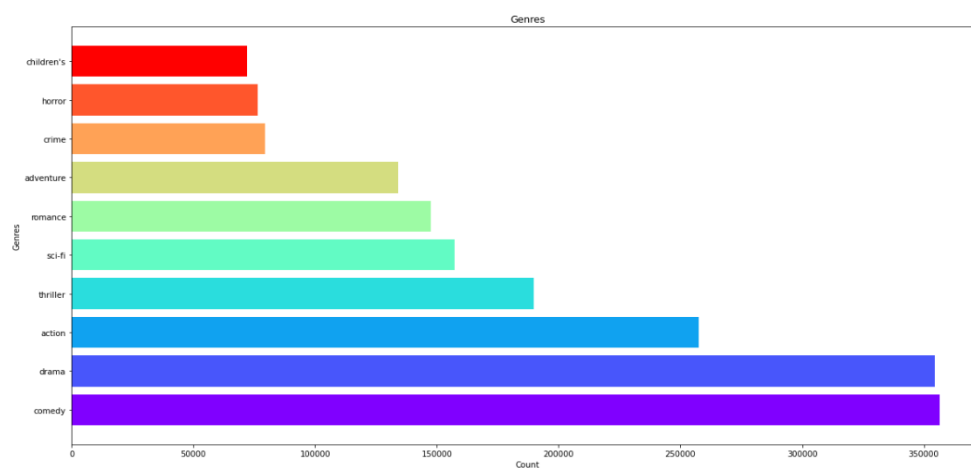


Figure 4-Rating distribution across all Genres

- Rating distribution across Gender

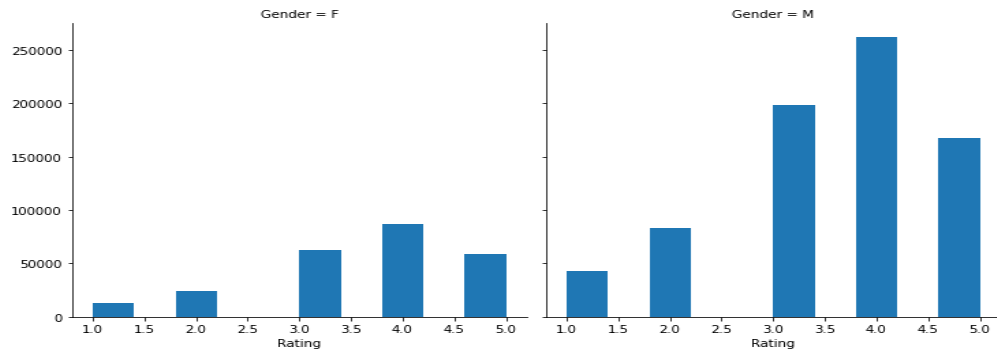


Figure 5-Rating distribution across Gender

7 FEATURE ENGINEERING

Feature engineering is the process by which knowledge of data is used to construct explanatory variables, features, that can be used to train a predictive model. Engineering and selecting the correct features for a model will not only significantly improve its predictive power, but will also offer the flexibility to use less complex models that are faster to run and more easily understood.

The MovieLens Dataset is already cleaned by grouplens research but for our concern and satisfaction we checked for NAN and NULL values and it is right that there are no NULL, NAN and missing values in our dataset. And also perform some technique to check whether outliers are present or not in dataset.

7.1 Detecting and handling missing values:

Then checked the missing values in the dataset using missing data percentage list which returns the percentage of missing values in each and every column present in the dataset. In this case 0.0% missing values in every column.

$$r = \frac{\sum(x - \bar{x})(y - \bar{y})}{\sqrt{\sum(x - \bar{x})^2 \sum(y - \bar{y})^2}}$$

Since, don't get any missing values in dataset, so there is no need to perform handling missing values techniques.

☐	UserID - 0.0%
	Gender - 0.0%
	Age - 0.0%
	Occupation - 0.0%
	Zip-code - 0.0%
	MovieID - 0.0%
	Rating - 0.0%
	Timestamp - 0.0%
	Title - 0.0%
	Genres - 0.0%

Figure 6- Missing Values

7.2 Outliers handling and detection

For detection outliers in data, check skew value for continuous features. If skewness value lies between -1 and 1 then the data is normally distributed and any major deviation from this range indicates the presence of extreme values in dataset. So, it seems only Timestamp skew value is 2.765691 which is greater than 1 so there may be chance of outliers in Timestamp. And another feature's skewness is bearable. But timestamp column which stores the last time when the rating was updated, it is the time mentioned in seconds

when the rating was provided. It is being like a primary key or it has more than 45000 unique values. It is different for each user. So, can't perform outlier treatment on this column.



	θ
UserID	0.005735
Age	0.398471
Occupation	0.404363
MovieID	0.092436
Rating	-0.553610
Timestamp	2.765691

Figure 7- Outliers

8 Feature Selection

8.1 Pearson Correlation

Pearson's Correlation

Coefficient Correlation is a technique for investigating the relationship between two quantitative, continuous variables, for example, age and blood pressure. Pearson's correlation coefficient (r) is a measure of the strength of the association between the two variables.

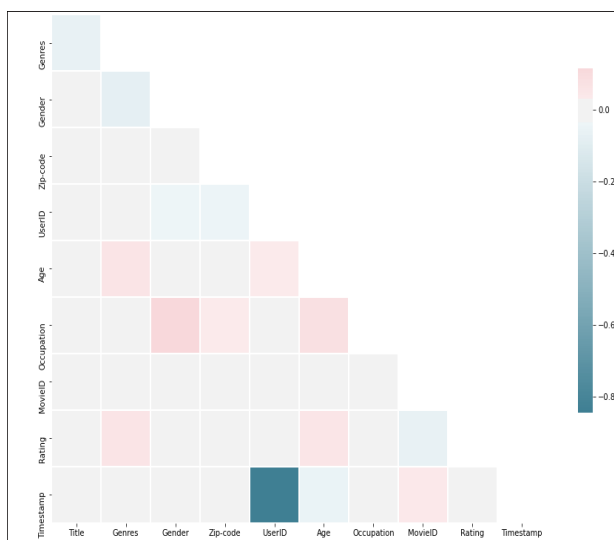


Figure 8 - Correlation

- A score closer to 1 or -1 is a positive or negative relationship. A perfect score of 1 is a direct correlation.
- In this heat map very light pink means positive, dark violet means negative. The stronger the color, the larger the correlation magnitude.
- So, by analyzing the above heatmap and correlation matrix Timestamp and Zip-code has very negative value (-0.843524), (-0.044845) and it goes on -1 side it negative correlation with other so we can think about to Drop it from dataset because it not contributes much information as compare to others.
- And also, if think about categories of recommendation's systems both are not contribute much as it need in Content Based Filtering based only need user related information which relate to recommend movies and Collaborative filtering is based on the assumption that people who agreed in the past will agree in the future and that they will like similar kind of objects as they liked in the past. So, according to this scenario don't need these features.
- Features needed Collaborative Filtering: Collaborative Filtering recommend movies on the basis of ratings provided by user or the similarity of user and it need only those features that provide information about the users (i.e. UserID, Age, Ratings, Occupation, Gender, Zip-code).

- Content Based filtering: Content based Filtering recommend movies to user on the basis of past history of user and content that it has. In this the algorithm will simply pick an item which similar content to recommender. So, it need only those features who related to a particular user (i.e. UserID, MovieID, Title, Genres, Ratings).

9 DIFFERENT APPROACHES TO RECOMMENDATION SYSTEM

There are several taxonomies for recommender systems but we used two most popular approaches: Collaborative and Content-based.

9.1 COLLABORATIVE FILTERING

Collaborative Filtering is the most popular and most implemented approach. Its original and simplest implementation this approach recommends to the active user, items that other users with similar tastes liked in the past. The similarity in taste of two users is calculated based on the similarity in the rating history of the users.

9.1.1 User-Based Collaborative Filtering

A target user's choices are compared with other users in the database to identify a group of "similar minded" people. Once this group is identified, highly rated content from the group are then recommended to the target user. This approach computes the correlation with all other users for each item and aggregate the rating of highly correlated users. User-based methods rely on the opinion of like-minded users to

predict a rating and generate recommendations.

9.1.1.1 IMPLEMENTATION (SINGULAR VALUE DECOMPOSITION)

The Singular Value Decomposition (SVD), a method from linear algebra that has been generally used as a dimensionality reduction technique in machine learning. SVD is a matrix factorization technique, which reduces the number of features of a dataset by reducing the space dimension from N-dimension to K-dimension (where $K < N$). In the context of the recommender system, the SVD is used as a collaborative filtering technique. It uses a matrix structure where each row represents a user, and each column represents an item. The elements of this matrix are the ratings that are given to items by users.

The factorisation of this matrix is done by the singular value decomposition. It finds factors of matrices from the factorisation of a high-level (user-item-rating) matrix. The singular value decomposition is a method of decomposing a matrix into three other matrices as given below:

$$A = USV^T$$

Where

- A is a $m \times n$ utility matrix.
- U is a $m \times r$ orthogonal left singular matrix, which represents the relationship between users and latent factors
- S is a $r \times r$ diagonal matrix, which describes the strength of each latent factor.
- V is a $r \times n$ diagonal right singular matrix, which indicates the similarity between items and latent factors.

The latent factors here are the characteristics of the items, for example, the genre of the

music. The SVD decreases the dimension of the utility matrix A by extracting its latent factors. It maps each user and each item into a r -dimensional latent space. This mapping facilitates a clear representation of relationships between users and items.

9.1.2 Item-Based Collaborative Filtering

Examines each item on the target user's list of chosen/rated items and finds other items in the choice set that seems similar to the item. In this case, similarity can be determined on predefined attributes (e.g. movie genre, lead actors, director, etc.) and/or by calculating correlations between items. The main advantage of Item-Based CF over User-Based CF is increased scalability due to the fact that items can be classified or pre-scored based on explicit attributes, making recommendation computation much faster. This approach computes for each user item the correlation with all other item and aggregates for each user the ratings for item that are already highly correlated. Item-based approaches look at ratings given to similar items and generate recommendation.

9.1.2.1 IMPLEMENTATION (KNN)

To implement an item based collaborative filtering, KNN is a perfect go-to model and also a very good baseline for recommender system development.

KNN is a non-parametric, lazy learning method. It uses a database in which the data points are separated into several clusters to make inference for new samples.

KNN does not make any assumptions on the underlying data distribution but it relies on item feature similarity. When KNN makes inference about a movie, KNN will calculate

the “distance” between the target movie and every other movie in its database, then it ranks its distances and returns the top K nearest neighbors movies as the most similar movie recommendations.

9.1.2.2 WORKING

Nearest Neighbors algorithms with `sklearn.neighbors` is used to compute the nearest neighbors where algorithm is “brute”, and we specify “metric=cosine”. So, that the algorithm will calculate the cosine similarity between rating vectors.

$$\text{sim}(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|}$$

Similarly, it will calculate distance of all the training cases with new case and calculates the rank in terms of distance. The smallest distance value will be ranked 1 and considered as nearest neighbors.

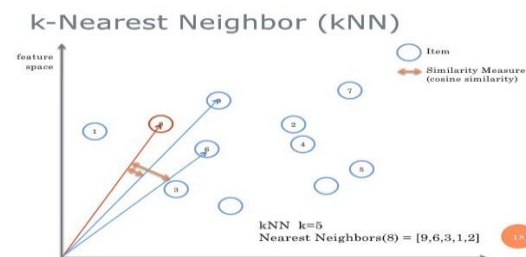


Figure 9 -KNN

9.1.2.3 Train model

Fit the model by trainset. And it use for recommend the rating to user for each movie that he didn't watch or rated.

9.1.3 TYPES OF KNN ALGORITHM (SURPRISE)

- KNNBasic is a basic collaborative filtering algorithm.
- KNNWithMeans (Centered KNN) is basic collaborative filtering algorithm, taking into account the mean ratings of each user.

- KNNBaseline is a basic collaborative filtering algorithm taking into account a baseline rating.

Recommendation:

1. Now extract the data in the testset using `data.build_anti_testset()`. The `build_anti_testset` return a list of ratings that can be used as a testset in the `test()` method the ratings are all ratings that are not in the testset.

2. Now predicted the rating for the movies that are not rated by the user. There is function `model.test` which returns the dataset that contains the predicted rating for all the movies. After this we created a function which returns predicted for all the movies in descending order to user.

9.2 CONTENT-BASED FILTERING

The system learns to recommend items that are similar to the ones that the user liked in the past. It exploits the content of data items to predict its relevance based on the user's profile. For instance, if the active user has rated positively a news article in the sports section, the system can learn to recommend other news articles in the same section.

Content-based filtering is based on the profile of the user's preference and the item's description. In CBF to describe items that use keywords apart from user's profile to indicate user's preferred liked or dislikes. In other words, CBF algorithms recommend those items are similar to those items that were liked in the past. It examines previously rated items and recommends best matching item.

Approaches

- TF-IDF
- CountVectorizer

9.2.1 TF-IDF

The concepts of Term Frequency (TF) and Inverse Document Frequency (IDF) are used in information retrieval systems and also content based filtering mechanisms (such as a content based recommender). They are used to determine the relative importance of a document / article / news item / movie etc.

Equation to calculate the TF-IDF score:

$$tfidf_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = total number of occurrences of i in j

df_i = total number of documents (speeches) containing i

N = total number of documents (speeches)

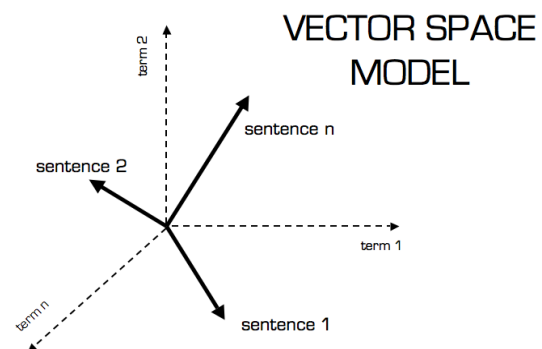


Figure 10- Vector Space Model

After calculating TF-IDF scores, how do determine which items are closer to each other. This is accomplished using the Vector Space Model which computes the proximity based on the angle between the vectors. In this model, each item is stored as a vector of its attributes in an n-dimensional space and the angles between the vectors are calculated to determine the similarity between the vectors. Next, the user profile vectors are also created based on his actions on previous attributes of items and the similarity between an item and a user is also determined in a similar way.

Sentence 2 is more likely to be using term 2 than using term 1. Vice-versa for Sentence 1. The method of calculating this relative measure is calculated by taking the cosine of the angle between the sentences and the terms. The ultimate reason behind using cosine is that the value of cosine will increase with decreasing value of the angle between which signifies more similarity. The vectors are length normalized after which they become vectors of length 1 and then the cosine calculation is simply the sum-product of vectors. Now compute Term Frequency-Inverse Document Frequency (TF-IDF) vectors for each document. This will give a matrix where each column represents a word in the overview vocabulary (all the words that appear in at least one document), and each column represents a movie, as before.

In its essence, the TF-IDF score is the frequency of a word occurring in a document, down-weighted by the number of documents in which it occurs. This is done to reduce the importance of words that frequently occur in plot overviews and, therefore, their significance in computing the final similarity score.

Fortunately, scikit-learn gives you a built-in `TfidfVectorizer` class that produces the TF-IDF matrix in a couple of lines.

9.2.1.1 IMPLEMENTATION

Do not have a quantitative metric to judge our machine's performance so this will have to be done qualitatively. In order to do so, use `TfidfVectorizer` function from scikit-learn, which transforms text to feature vectors that can be used as input to estimator.

Used the Cosine Similarity to calculate a numeric quantity that denotes the similarity between two movies. Since, it used the TF-IDF Vectorizer, calculating the Dot Product will directly give us the Cosine Similarity

Score. Therefore, used sklearn's `linear_kernel` instead of `cosine_similarity` since it is much faster.

Now, have a pairwise cosine similarity matrix for all the movies in the dataset. The next step is to write a function that returns the 20 most similar movies based on the cosine similarity score.

9.2.2 COUNTVECTORIZER

`CountVectorizer` is a great tool provided by the scikit-learn library in Python. It is used to transform a given text into a vector on the basis of the frequency (count) of each word that occurs in the entire text. This is helpful when dataset have multiple such texts, and wish to convert each word in each text into vectors.

`CountVectorizer` creates a matrix in which each unique word is represented by a column of the matrix, and each text sample from the document is a row in the matrix. The value of each cell is nothing but the count of the word in that particular text sample.

After performing count vectorizer, a matrix is used to calculate the cosine similarity between two movies same as already did in TF-IDF. And then it recommend the top n movie on the basis of similarity score.

10 EVALUATION

As for any machine learning algorithm, it need to be able to evaluate the performances of recommender system in order to decide which algorithm fit the best situation. So, here used two main methods in order to compare and measure the performance of different approaches of recommendation systems.

10.1 MEAN ABSOLUTE ERROR (MAE)

It is the difference between the actual value(rating) and the predicted value. MAE is always lower than RMSE. Lower the MAE value, better the model.

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

10.2 ROOT MEAN SQUARED ERROR (RMSE)

RMSE is similar to MAE but the only difference is that the absolute value of the residual is squared and the square root of the whole term is taken for comparison.

The advantage of using RMSE over MAE is that it penalizes the term more when the error is high. (Note that RMSE is always greater than MAE)

$$RMSE = \sqrt{\frac{\sum_{i=1}^N (Predicted_i - Actual_i)^2}{N}}$$

10.3 RMSE/MAE Comparison With Different Algorithms-Item Based

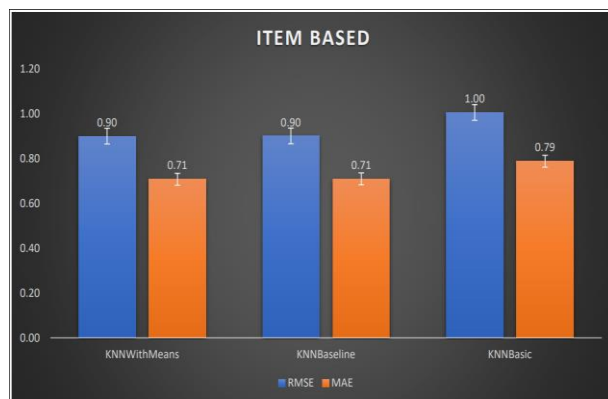


Figure 11- RMSE/MAE Comparison With Different Algorithms-Item Based

- KNNWithMeans and KNNBaseline shows same RMSE and MAE, while KNNBasic is slightly higher than from these two.

10.4 RMSE/MAE Comparison With Different Algorithms-User Based

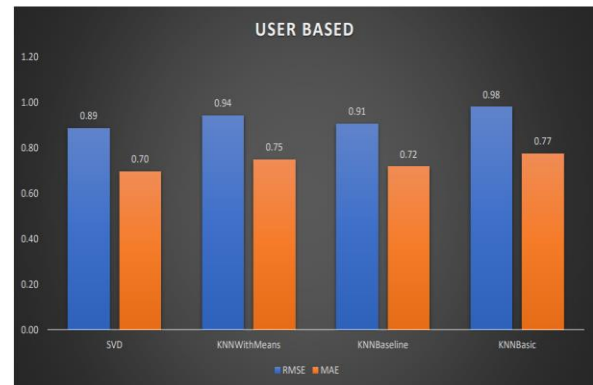


Figure 12- RMSE/MAE Comparison With Different Algorithms-User Based

- SVD shows best result in user based with RMSE:0.89 and MAE:0.70.

11 WEB APPLICATION

To recommend movie from the trained model used Flask to develop a frontend of the website.

11.1 UI Interface:

- User types in the URL for the system on a Web Browser.
- User logs into the system using his 'UserID' or 'Item id' or 'Movie Title'. This is because used three search bars i.e. for content based recommendation and user based and item based for collaborative filtering.
- The user chooses from amongst the different types of recommendation systems available.
- If the user chooses 'Collaborative Filtering' option, the system calculates similar users or similar items making use of engineering algorithms, and then recommends items to the users based on the most similar user or most similar items.
- If the user chooses 'Content based Filtering' option, the system then makes use of the content information to make recommendations.

- The System provides the user with the functionalities like if user search in search box depends on him which type of recommendation he prefers i.e. content based or collaborative filtering after pressing click here button it'll give n recommendations on the basis of his interest.

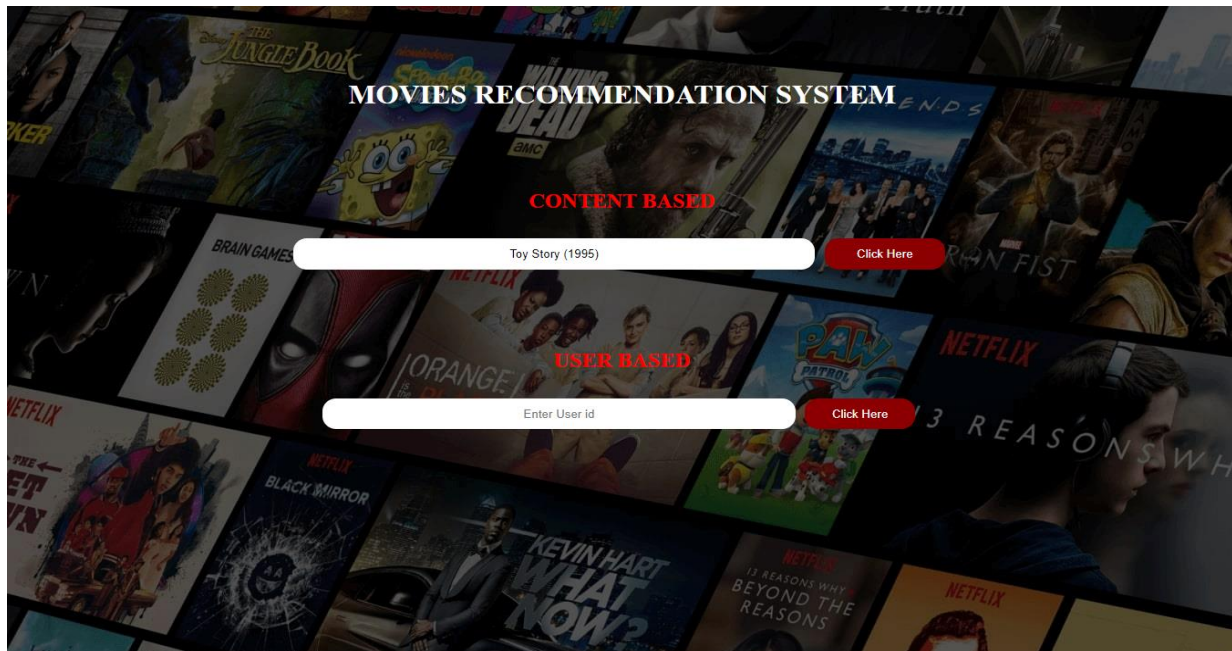


Figure 13 - User Interface

TOP RECOMMENDATIONS	
Showing results for: Toy Story (1995)	
Movie Title	Genres
Aladdin And The King Of Thieves (1996)	Animation children's comedy
American Tail, An (1986)	Animation children's comedy
American Tail: Fievel Goes West, An (1991)	Animation children's comedy
Rugrats Movie, The (1998)	Animation children's comedy
Bug's Life, A (1998)	Animation children's comedy
Toy Story 2 (1999)	Animation children's comedy
Saludos Amigos (1943)	Animation children's comedy
Chicken Run (2000)	Animation children's comedy
Adventures Of Rocky And Bullwinkle, The (2000)	Animation children's comedy
Goofy Movie, A (1995)	Animation children's comedy romance

Figure 14 – Recommendation on web UI

11.2 UI BACKEND

```
@app.route('/', methods=['GET', 'POST'])
def index(): return render_template('index.html')

@app.route('/predict/', methods=['POST'])
def main():
    if request.method == 'POST': m_name = request.form['movie_name'].title()
    if request.method == 'GET': m_name = request.args.get('movie_name').title()

    print(m_name)
    # check = difflib.get_close_matches(m_name, all_titles, cutout=0.50, n=1)
    if m_name not in all_titles: return(render_template('index.html'))
    else:
        result_final = get_recommendations(m_name)
        names = []
        dates = []
        for i in range(len(result_final)):
            names.append(result_final.iloc[i][0])
            dates.append(result_final.iloc[i][1])
        return flask.render_template('positive.html', result = 1, ret = zip(names, dates), search_name=m_name)

@app.route('/user/', methods=['POST'])
def userPred():
    print(request.method)

    try: a = int(request.form['user_id'])
    except: return render_template('index.html')
    # b = request.form['recommendation']
    print(a)
    b = 10
    t1, t2, t3 = rec(a, b)
    return render_template('positive.html', neut = 1, ret = zip(t1, t2, t3))
```

Figure 15- UI backend

12 CONCLUSION

In the time period that was given to us we tried several algorithms and model in order to implement different recommendation system approaches such as Collaborative Filtering and Content based filtering.

In item based collaborative filtering, implement model based algorithms that is KNNWithMeans, KNNBasic and KNNBaseline. Checked their performance using RMSE and MAE from this came to the conclusion that RMSE and MAE of KNNWithMeans is much better than another algorithms because of lowest RMSE value.

In user based collaborative filtering, implemented the above same algorithms along with matrix factorization based algorithm SVD (Singular Value Decomposition). Checked their RMSE and MAE values. From this we came to conclusion that performance of SVD is better than other algorithms.

In content based filtering, to detect similarities between movies, need to vectorize, using CountVectorizer. Choose CountVectorizer rather than TfidfVectorizer for one simple reason: Need a simple frequency counter for each word in Genres column. Tf-Idf tends to give less importance to the words that are more present in the entire data which is not what we want for this application, because every word is important to detect similarity. Once the matrix containing the count for each word, it can apply the cosine_similarity function.

13 REFERENCES

1. <https://medium.com/fnplus/evaluating-recommender-systems-with-python-code-ae0c370c90be>
2. <https://towardsdatascience.com/how-to-build-from-scratch-a-content-based-movie-recommender-with-natural-language-processing-25ad400eb243>
3. <https://surprise.readthedocs.io/en/stable/trainset.html>
4. <https://surprise.readthedocs.io/>
5. <https://www.kaggle.com/rounakbanik/movie-recommender-systems>
6. <https://www.edureka.co/blog/background-image-in-html/>
7. <https://towardsdatascience.com/building-and-testing-recommender-systems-with-surprise-step-by-step-d4ba702ef80b>

Notebook link:

<https://github.com/Internship-BVoc/Movies-Recommender-System>