Project Report

on

Movie Recommender Systems

Submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY

DEGREE

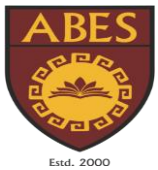
Session 2019-20 in

Information Technology

Ayush Plawat
1603213024

Under the guidance
of
Mr. Ashwin Perti (AHOD, IT)

DEPARTMENT OF INFORMATION TECHNOLOGY
ABES ENGINEERING COLLEGE, GHAZIABAD
AFFILIATED TO
DR. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY, U.P., LUCKNOW
(Formerly UPTU)



Student's Declaration

We hereby declare that the work being presented in this report entitled **"Movie Recommender Systems"** is an authentic record of our own work carried out under the supervision of **Mr. Ashwin Perti, AHOD, IT Department, ABESEC.**

The matter embodied in this report has not been submitted by us for the award of any other degree.

Date:

Ayush Plawat
1603213024
IT-B (VII)

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Signature of HOD

Dr. Amit Sinha
(IT Dept., ABESEC)

Date:.....

Signature of Supervisor

Mr. Ashwin Perti
(IT Dept., ABESEC)

Date:.....

Acknowledgement

I would like to express my special thanks of gratitude to my guide **Mr. Ashwin Perti** as well as our HOD **Dr. Amit Sinha,** who gave us the golden opportunity to do this wonderful project on the topic **"Movie Recommender Systems"**, which also helped me in doing a lot of Research and we were able to absorb a great amount of knowledge.

Ayush Plawat
1603213024
IT-B (VII)

List of Figures:

Table of Contents:

ABSTRACT

Recommender Systems Have Become Ubiquitous in Our Lives. Yet, At present, they are a long way from ideal.

In This Project, we endeavor to comprehend the various types of suggestion frameworks and look at their presentation on the Movie Lens dataset. We Attempt to construct an adaptable model to play out this investigation.

Here in this project, we are going to be looking at different approaches of recommender systems on the famous MovieLens dataset.
It is a dataset that consists of 100,000 ratings on a tons of movies along with relevant tags and other useful information.

(1) INTRODUCTION:

The recommendation system is a mechanism that aims to predict a user's interests and make recommendations based on those interests. There are a wide range of such systems and all of them have their separate and different implementations. These have been more common in recent years and are now being used in most of the web services we use.

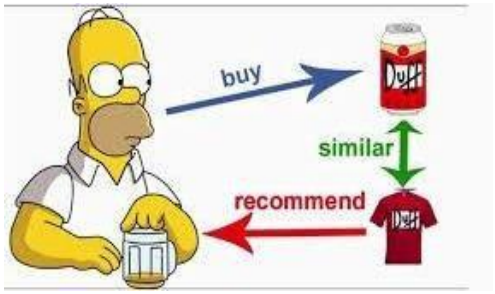


Fig.1.1 - Recommendation loop

The quality of these channels ranges from films, songs, books and videos, to tales of friends on social media networks, to websites of e-commerce goods, to users on technical and dating websites, to reports of searches received on Google. These programs are also capable of collecting information about the preferences of consumers and can utilize this data later on to improve their suggestions.

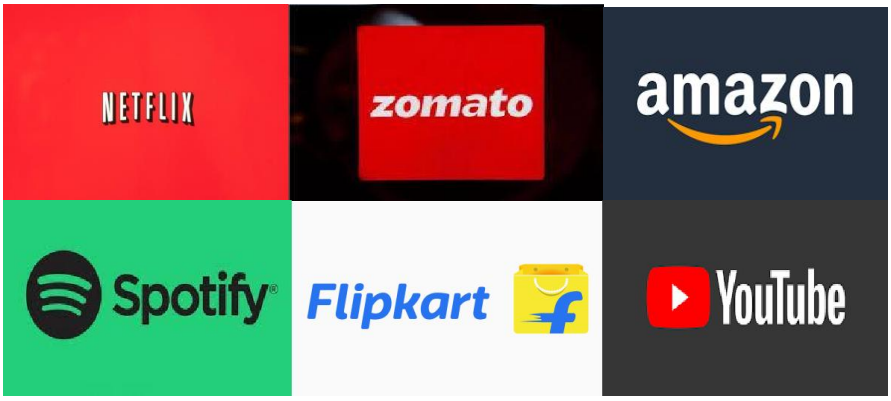


Fig.1.2 - Some organizations that use 'Recommendation Engines'

With the rise of big Internet Companies such as YouTube, Amazon, Netflix, Hulu, Spotify, Flipkart and many such Internet based enterprises, recommendation systems have become an inseparable part of our lives. These systems not only make users interact better with the websites but also increase their retention which in turn increases the companies business.

Fig. 1.3- Amazon's Recommendation System

For example, Facebook will track your engagement with various stories on your feed and learn which story styles relate and you. The recommender systems will also enable changes focused on a large range of events. For instance, if Amazon notices that a lot of customers buying new Apple MacBook's likewise purchase USB C to USB connectors, the connector can be recommended for a new customer to only put the MacBook into his bag.

Because of the advances in recommender structures, clients interminably anticipate inconceivable suggestions. This has incited a large supplement by tech relationship on improving their proposal structures. Regardless, the issue is amazingly offbeat. Each client has various propensities and tendencies. In addition, even the sort of a solitary client can differentiate subordinate upon unlimited parts, for example, point of view, season, or sort of improvement the client is doing.

The two essential techniques are commonly used for these systems. One is **content-based filtering**, where we endeavor to recognize the customer's focal points using information assembled, and propose things based on his/her data. The other is **collaborative filtering**, where we endeavor to assemble equivalent customers and use the data we obtain about the social event to make the recommendations**.**

(2) PROBLEM OBJECTIVE:

(2.1) Aim:

The main aim is to utilize the user provided information and data to give recommendations and a better end-user experience such that the user keeps on visiting or using our product, website etc. We also want the user to stay for prolonged periods of time thereby increasing the user time which in turn is good for the business.

Basically, we want to serve the user with the most specific content he/she demands or wishes to consume without his explicit instructions. Seeing more relatable content according to his expectations will provide a more personalized experience for every user. So no user will see or interact with the same content

Recommender systems are significant to both pro associations and customers. They reduce trade costs of finding and picking things in a web purchasing condition. Proposition systems have in like manner showed to improve dynamic method and quality. In a let's say a Shopify store, recommender systems update salaries, for the way that they are suitable strategies for selling more things. Recommender systems support customers by allowing them to go along their custom tags. Along these lines, the need to use capable and careful recommendation strategies inside a structure that will give the most likeable and choice specific proposition to customers can't be over-underlined.

Recommender structure is described as a unique system for customers under complex information circumstances. In like manner, the recommender system was portrayed by E-business as an instrument that helps customers with glancing using heaps of data which is related for customers' potential benefit and tendency. The Recommendation system was portrayed as some strategies for aiding and expanding the social approach to users i.e. being more user friendly. A customer after setting up an account can now enjoy a personalized experience based on his choices and preferences. The more content or substance he consumes, the better his user-experience gets.

(2.2) DATA FILTERING APPROACHES:

The most popular approaches for building recommendation systems are:

- collaborative filtering
- content-based filtering
- hybrid filtering.

(2.2.1) Collaborative Filtering:

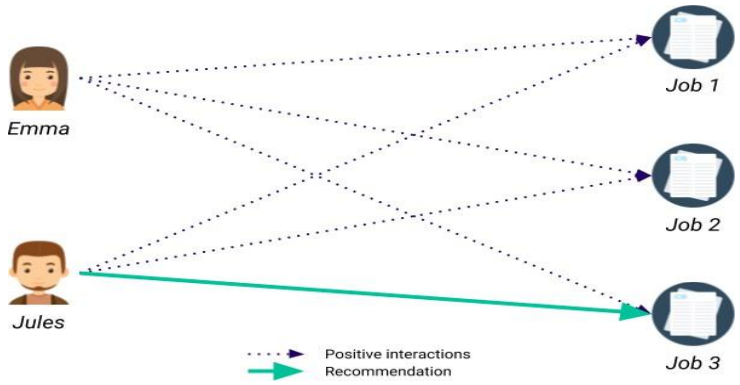


Fig. 2.2.1 Collaborative Recommendation

Collaborative filtering strategy is the fullest evolved and the most usually executed. It proposes things by perceiving different clients with essentially same taste; it utilizes their tendency to prescribe things to the dynamic client. These systems have been executed in various application zones. Get-together Lens is a news-based structure which utilizes collaborative procedures in helping clients to find articles from enormous news databases. Ringo is an online social data filtering framework that utilizes collaborative filtering to fabricate clients' profiles dependent on their evaluations on music groupings. Amazon utilizes theme growing calculations to improve its proposition. The framework utilizes collaborative filtering procedures to beat adaptability issues by making a table of close to things isolated using the user provided data in a framework. The framework by then proposes different things which are proportional online by the clients' buy history and his user account.

Collaborative Filtering systems make recommendation for a customer subject to appraisals and affinities data of various customers. The central key idea is that in case two customers have both respected certain standard things, by then the things that one customer has worshiped that the other customer has not yet endeavored can be kept up to him. We see these procedures, considering, on various Internet stages, for instance in shopify stores.

We are grasped things subordinate upon the examinations and purchase data that these stages gather from their customer base. We inspect two figuring for this as mentioned below.

(2.2.1.1) Nearest Neighbors Collaborative Filtering:

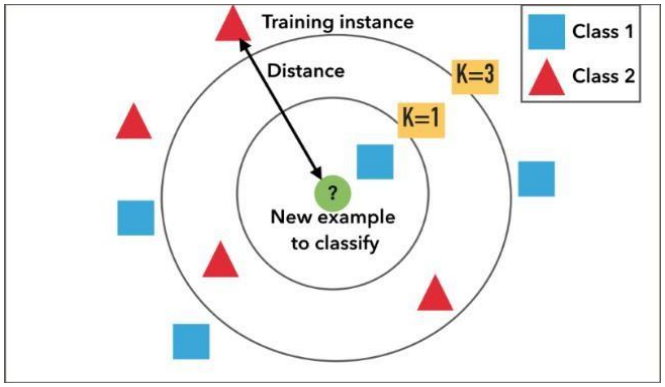


Fig.2.2.1.1 - KNN Algorithm

In this approach customers or users with similar taste or choices will be grouped as a whole and will be provided almost the same suggestions. It works on the theory that people with similar preferences will prefer the same things as well.

The count at first registers the similarity between customers by using the line vector in the evaluation's framework identifying with a customer as a depiction for that customer. The most popular approach to this is the cosine similarity or Pearson's coefficient.

For us to recognize the rating of a given user for a given vector 'j', we find the points(users) closest to it by let's say cosine similarity and at the same time take weight of the examinations of 'k' similar users within the same regard.

(2.2.1.2) Latent Factor Methods:

- Users and items are connect by latent features.

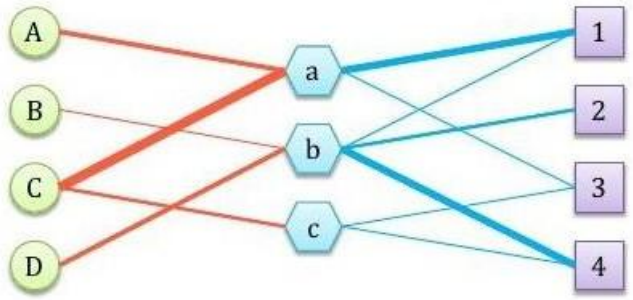


Fig. 2.2.1.2 - Latent Factor Algorithm

This algorithm is one of the most widely used approaches in collaborative filtering. It decomposes the appraisals matrix let's say '**F**' into 2 tall matrices '**G**' & '**H**' with a matrix 'G' &'H' having dimensions [num_users, x, k], here k=number of latent factors. The end goal is: -

$$F = G.H^T$$

We know take the dot product of f_i of matrix and matrix to compute rating f_{ij} . Matrices 'G' & 'H' are randomly initialized or we can perform the SVD on the ratings matrix.

Then we will minimize the error between the actual and observed ratings using the algorithm by performing chastic gradient descent to find the one with minimum error starting from the initial matrices.

(2.2.2) Content Based Filtering:

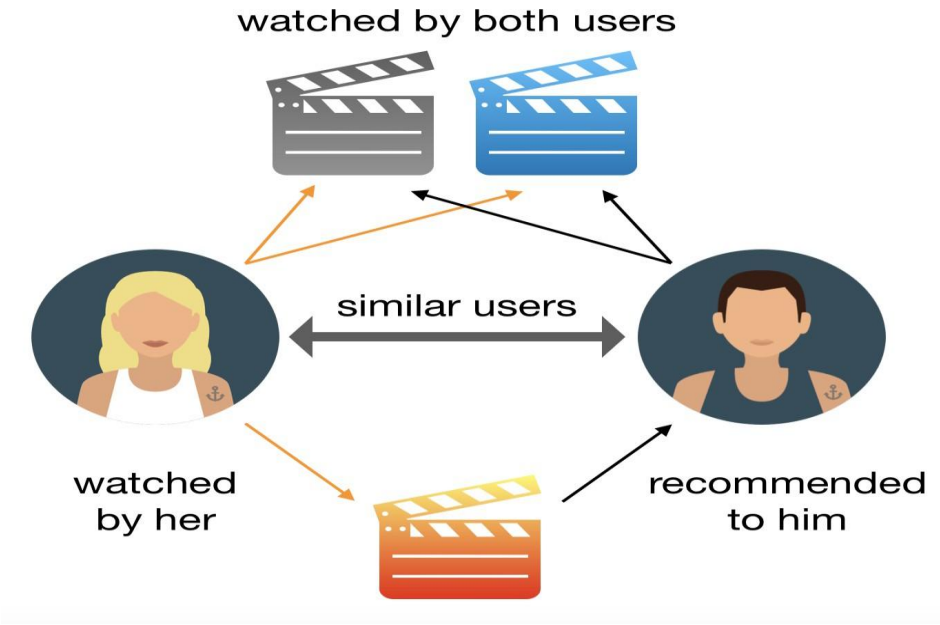


Fig. 2.2.2.1 - Content- Based Algorithm

This framework solely focuses on the content to provide recommendations. These algorithms basically take user's account data, watch history, search data etc. to provide a personalized experience. Since a lot of users can have similar choices or preferences, the chances are of experience similar suggestions or content is quite high. Still this approach is one of the most robust and tested approaches in the field.

The suggestions in the search are usually influenced by the tag or genre of the content. Here this algorithm plays a key role in providing suggestions that are usually of user's taste. The Bayesian classifier is a very good example of this. In the simplest it categorizes things into categorizes and works exceptionally well despite it's simplicity.

(2.2.3) Hybrid Filtering Techniques:

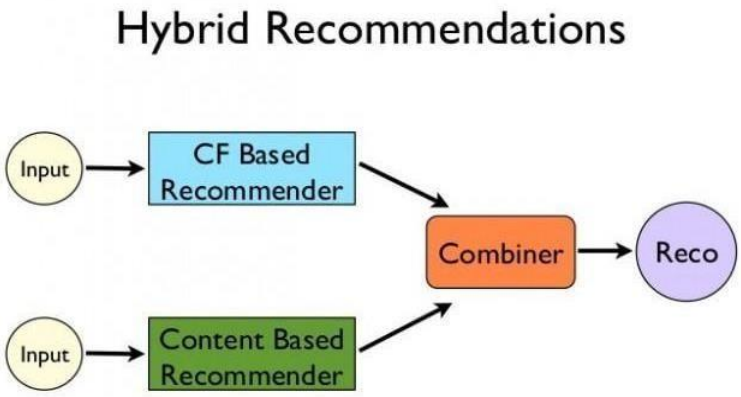


Fig. 2.2.3.1 - Hybrid Algorithms

Notwithstanding the accomplishment of these two filtering strategies, a few obstructions have been seen. Some difficulties often encountered with these structures are bound substance evaluation, overfitting and lack of relevant data. In like manner, absolute structures show cold-start, sparsity and adaptability issues. These issues generally reduce the chance of recommendation. To ease a bit of the issues clear,

Hybrid filtering, which unites at any rate two filtering method or more to build up the precision and execution of such structures has been proposed. These systems take an interest in any event two filtering approaches to manage handle their qualities while removing or reducing their separating inadequacies. They can be classified based on their operations into**:**

- Weighted Hybrid
- Mixed Hybrid
- Switching Hybrid
- Feature-Combination Hybrid
- Cascade Hybrid
- Feature-Augmented Hybrid

Collaborative filtering and content-based filtering approaches are mainly used today by executing these techniques contrastingly and the eventual aftereffects of this gets us the best parts of both the algorithms. Finally, a general joined model which wires both algorithms after filtering properties can be made.

The issue of lack of relevant data and the cold start leads to high error between the observed and actual ratings but as the algorithm gets more data through users the accuracy of the algorithm also increases. The lack of availability of clean data is one of the biggest blocks in this industry. As the companies grow the data gathered also grows but cleaning it is a big task which leads the rising need of data analysts and scientists to extract relevant data.

After clean data is obtained and relevant working parameters are substituted in the algorithm the suggestions improve drastically. This process does not end here but is an ever going, ever changing one that needs to constantly evolved to keep up with the rising user base. Scaling of such platforms is also substantial work that needs to be done right while keeping things like technical debt low.

(3) PHASES OF RECOMMENDATION PROCESS:

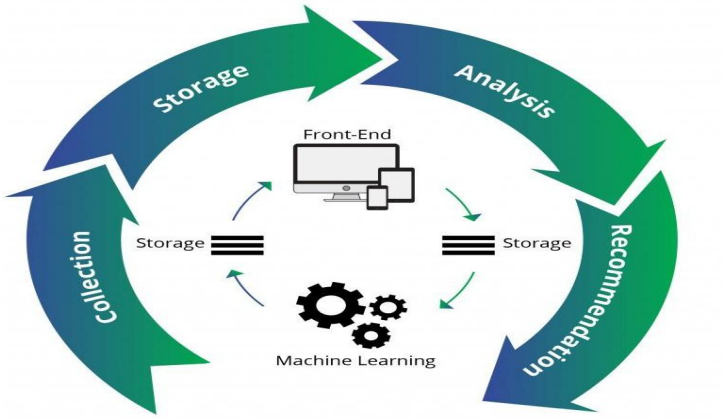


Fig.3.1 - Phases of Recommendation

This assembles critical information of customers to deliver a customer profile or model for the desire tasks including customer's quality, practices or substance of the advantages the customer gets to. A proposal expert can't work correctly until the customer profile/model has been all around fabricated. The structure needs to know anyway much as could sensibly be normal from the customer to give reasonable recommendations legitimately from the earliest starting point.

Recommender structures rely upon different kinds of data, for instance, the most favorable first class unequivocal analysis, which fuses express commitment by customers as for their eagerness for thing or certain contribution by initiating customer tendencies by suggestion through watching customer lead.

The feedback in a hybrid model can be obtained from various sources such as common search patterns, inclination towards a certain category etc. This comprises of user psychology, learning patterns, adaptability to platform, retention rate etc.

Usually more than one channels for feedback are used in these models to better understand the behavior. External events also affect how the content is consumed or reviewed. All of these aid in the process of providing relevant suggestions to the user.

The various phases involved in the process of recommendation of an object are:

- Information Collection Phase
- Learning/Analysis Phase
- Prediction/Recommendation Phase
- Feedback

(3.1) Information Collection Phase:

The dataset will consist of just over 100,000 ratings applied to over 9,000 movies by approximately 600 users. The dataset was downloaded from **MovieLens**.

The data is distributed in four different CSV files which are named as ratings, movies, links and tags. For building this recommender we will only consider the ratings and the movies datasets.

The **ratings** dataset consists of 100,836 observations and each observation is a record of the ID for the user who rated the movie (userId), the ID of the Movie that is rated (movieId), the rating given by the user for that particular movie (rating) and the time at which the rating was recorded(timestamp).

The movies dataset consists of the ID of the movies(movieId), the corresponding title (title) and genre of each movie(genres).

(3.2) Learning Phase:

It is perhaps the most important step in the working of the algorithm. Any model or hypothesis will work as good as it is trained. The algorithm we used i.e. k-Nearest

Neighbors (kNN) on the dataset uses proximity of data objects to one another to provide the most relevant content to the user.

This pattern recognition aspect of kNN is extensively used in both classification & regression-based problems. The output depends on the type of problem we are applying it on.

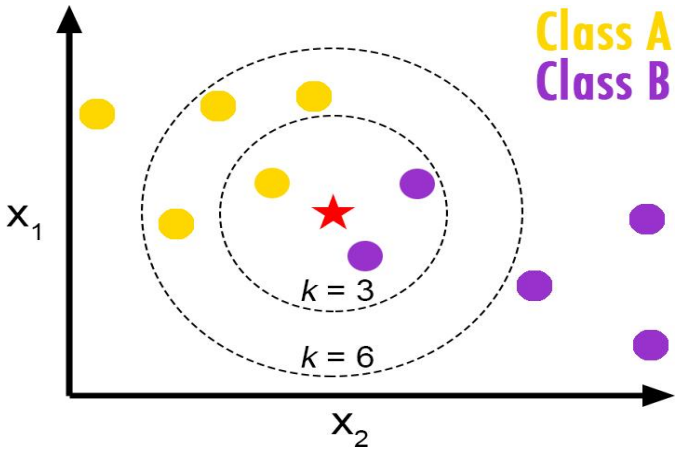


Fig. 3.2.1 - KNN Regression

The ***k-NN algorithm***, considers the weight or value of an object with respect to its proximity to its neighbors or other data points and then assigns them to the class of neighbors they are close at.

In ***k-NN regression***, the output is the average of the class of neighbors the object is assigned to and is independent of its own value.

k-NN is a kind of example-based learning where the capacity is just approximated to the closest points and all calculation is conceded until work assessment.

For any type of problems, the logic revolves around the fact to assign more value to contributions of the closest data points aka neighbors. The closer one should contribute more than the distant ones.

The neighbors are a set of data points or objects for which the class (for *k*-NN classification) or the object property value (for *k*-NN regression) is known. This can essentially be considered training the algorithm.

(3.3) Prediction/Recommendation Phase:

This is essentially the fun or the main part where we provide value to the user. The suggestions or recommendations can be provided both explicitly or implicitly. The suggestions can put under a separate tab for the user or can be mixed in his feed for a seamless experience. It depends on the deployer to deploy them at the most appropriate place for better results. The steps to this can be: -

- The data i.e. to be predicted can be predicted row by row. For every row, the closest neighbors can be determined using say Euclidian distance, the parameters that will determine the weight of these neighbors need to be set beforehand by considering their relevance.
- The weighted sum of these nearest neighbors is now calculated to determine how much each of them contribute to the value.
- Keep repeating this process until we get all the data we want to predict and the use this data to feed the feedback loop as well.

Computing time increments as 'k' increments; however, the preferred position is that higher estimations of k give smoothing that diminishes weakness to clamor in the Training Set. Ordinarily, k is in units of many units, as opposed to in hundreds or thousands.

(3.4) Feedback:

The feedback is essential for the proper development of the model. It can be provided by a user in the following ways:

(3.4.1) Explicit Feedback:

Sometimes asking the user directly or explicitly for feedback can be extremely. It not only completes the feedback look but also helps the developers to gain a client-side perspective of the product. Many shortcomings are often overlooked in the process that can be highlighted by the user.

In the initial phase of product deployment this type of feedback helps shorten the bridge between client and enterprise. It also increased the trust and interest in the product and leads to more user and more better suggestions as well.

(3.4.2) Implicit Feedback:

Most of the times, the client implicitly signals the shortcomings of the model. Their behavior with how they interact with the product is directly correlated to the quality of it. Many times, a highly engineered product fails just because the expectations of user were entirely different and they were not able to adapt to it. Just by gathering some basic user data that they essentially provide the quality and working of the model can be increased by leaps and bounds.

(3.4.3) Hybrid Feedback:

The qualities of both implicit and explicit feedback can be joined in a hybrid system so as to get the best of both worlds and by overcoming each other's shortcomings or at least keeping them to a minimum. In the end, the target is to provide the best suggestions and experience to the user.

(4)EVALUATION METRICS FOR RECOMMENDATION ALGORITHMS:

The idea of a recommendation algorithm can be evaluated using different kinds of estimation which can be accuracy or consideration. The sort of estimations used depends upon such a filtering procedure. One of them is accuracy. It is the amount of correct suggestions/recommendations among total suggestions provided.

The quality of every measurement relies upon the highlights of the dataset and the sort of assignments that the recommender system will do.

Statistical Accuracy Metrics evaluate accuracy of a filtering technique by comparing the predicted ratings directly with the actual ones.

Mean Absolute Error (MAE), Root Mean Square Error (RMSE) and **Correlation** is a statistical accuracy metric. It is one of the most widely used metric to measure the deviation of obtained values from actual values.

$$\textbf{Mean absolute error} = \sum_{u,i} | p_{u,i} - r_{u,I} |$$

where $P_{u,i}$ is the predicted rating for user ***u.*** We ideally want the mean absolute error to be as low as possible. Another approach that is the Root mean square is:-

Also, the **Root Mean Square Error (RMSE)** is given as:

$$\text{Mean absolute error} = \sum_{u,i} | p_{u,i} - r_{u,i} |$$

This approach provides better recommendation accuracy. There are several other metrics like F-measure, Precision recall curve etc.

These metrics not only help in improving recommendations but also help us outline and discover better parameters for our model. One of the other things that we are concerned with is Precision and Recall.

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

F-***measure*** is the metric that helps us simplify precision and recall in a single matrice and helps us compare different algorithms.

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Substituting Precision and Recall:

$$F_1 = 2 \cdot \frac{\frac{TP}{TP+FP} \cdot \frac{TP}{TP+FN}}{\frac{TP}{TP+FP} + \frac{TP}{TP+FN}}$$

(5)FUTURE:

The advancement in the field of recommendation systems has been going on in an unprecedented rate. With more and more companies becoming consumer facing and competing for the same piece of pie.

Tons of data scientists are employed to develop and deploy hybrid systems at a scalable level. These systems need to constantly evolve and grow with the company.

With the advancement in Neural nets and deep learning methods using the vast amount of data we are generating every day; we are seeing some exceptional results.

Every consumer centric business is looking to increase their retention rate and reach a more wider consumer base as soon as possible. The possibilities in the field is immense, with engineers now pairing up with neurologists to better map human behavior. All in all, we will be seeing some immense development here in the field.

(6)CONCLUSION:

This field of these systems open new odds of providing content to the userbase or customers. It moreover helps with moderating the issue of content over-trouble that is a run of the mill wonder with information recuperation systems and engages customers to move toward things and organizations which are not immediately available to customers on the structure.

Fig.6.1 - Recommendations

This report showed some prominent approaches in building recommendation systems while also describing the method to build, evaluate and derive feedback for them. We also discussed some parameters for evaluating them.

(7)RESULTS:

- The input movie given by the user is ‘V for Vendetta (2006)’, hence the possible matches or recommendations a user will receive after watching this movie are:

```
Input movie: V for Vendetta
Possible matches for: ['V for Vendetta (2006)']

Recommendation system starts
Recommendations for V for Vendetta:
1: Kill Bill: Vol. 2 (2004), with distance of 0.45811667820588053
2: Kill Bill: Vol. 1 (2003), with distance of 0.45113614263931523
3: Lord of the Rings: The Return of the King, The (2003), with distance of 0.44164255679675435
4: Casino Royale (2006), with distance of 0.44065041434654284
5: Prestige, The (2006), with distance of 0.439073634334578
6: 300 (2007), with distance of 0.43626464974891654
7: Iron Man (2008), with distance of 0.43219217884683925
8: Sin City (2005), with distance of 0.42203184608146216
9: Dark Knight, The (2008), with distance of 0.4187473948012579
10: Batman Begins (2005), with distance of 0.3800680734223937
```

- The next input movie given by the user is ‘Shanghai Triad (1995)’, hence the possible matches or recommendations the user will receive after watching this movie are :

```
Input movie: Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)
Possible matches for: ['Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)']

Recommendation system starts
Recommendations for Shanghai Triad (Yao a yao yao dao waipo qiao) (1995):
1: Amateur (1994), with distance of 0.812136512046904
2: Chungking Express (Chung Hing sam lam) (1994), with distance of 0.8052131703346157
3: Farinelli: il castrato (1994), with distance of 0.7994004038621397
4: Farewell My Concubine (Ba wang bie ji) (1993), with distance of 0.7874038762523473
5: Madness of King George, The (1994), with distance of 0.7848752995357182
6: Eat Drink Man Woman (Yin shi nan nu) (1994), with distance of 0.7830370983465151
7: Mrs. Parker and the Vicious Circle (1994), with distance of 0.7810640790405625
8: Richard III (1995), with distance of 0.7629395484286996
9: Queen Margot (Reine Margot, La) (1994), with distance of 0.7624432396973219
10: To Live (Huozhe) (1994), with distance of 0.7587464776779457
```

(8)APPENDIX (Code) :

1. Importing the necessary libraries

+ Code + Markdown

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.sparse import csr_matrix
from sklearn.neighbors import NearestNeighbors
%matplotlib inline
plt.rcParams['figure.figsize'] = (7.0, 5.0)
```

[]:

```
# Fuzzywuzzy is a python package for string matching
# If you don't have it use pip install fuzzywuzzy in the Anaconda Prompt to install it
from fuzzywuzzy import fuzz
```

[]:

```
df_movies = pd.read_csv('movies.csv')
df_movies.head()
```

[]:

```
df_ratings = pd.read_csv('ratings.csv')
df_ratings.head()
```

+ Code + Markdown

[]:

```
df_movies.shape
```

[]:

```
df_movies.describe()
```

[]:

```
df_ratings.shape
```

[]:

```
df_ratings.describe()
```

[]:

```
num_users = len(df_ratings.userId.unique())
num_items = len(df_ratings.movieId.unique())
print('There are {} users and {} movies in this data set that are unique.'.format(num_users, num_items))
```

[ ]:
```
num = df_ratings['rating'].value_counts()
print('The number of ratings of each type are:--')
print(num)
```

+ Code + Markdown

2. Exploratory Data Analysis and Wrangling

[ ]:
```
sns.countplot(x="rating", data=df_ratings)
```

[ ]:
```
sns.distplot(df_ratings['rating'])
```

[ ]:
```
# Here the movies and ratings dataframe are joined to make a single data frame
# Left join is performed here & movieID is used as keys here

movie_rat = pd.merge(left=df_movies, right=df_ratings, how="left", left_on='movieId', right_on='movieId')
```

[ ]:
```
movie_rat.head()
```

+ Code + Markdown

[ ]:
```
movie_rat.tail(10)
```

[ ]:
```
movie_rat.isna()
```

Null values exist in the dataframe, so we have to handle that.

[ ]:
```
movie_rat.info()
```

[]:
```
# NaN values exist only in userId, rating & timestamp column
# Calculating the missing values

miss_val = movie_rat.isnull().sum()
miss_val = miss_val[miss_val>0]
print(miss_val)
```

+ Code + Markdown

[]:
```
miss_val.sort_values(inplace=True)
```

[]:
```
print(miss_val)
```

[]:
```
# Visualizing the missing values
miss_val = miss_val.to_frame()
miss_val.columns = ['count']
miss_val.index.names = ['rating']
miss_val['rating'] = miss_val.index

# Plotting the missing values
sns.set(style="whitegrid", color_codes=True)
sns.barplot(x = 'rating', y = 'count', data=miss_val)
plt.show()
```

[]:
```
# Ratings Count
df_ratings_cnt_tmp = pd.DataFrame(df_ratings.groupby('rating').size(), columns=['count'])
df_ratings_cnt_tmp
```

+ Code + Markdown

[]:
```
# There are still multiple ratings of zero

total_cnt = num_users * num_items
rating_zero_cnt = total_cnt - df_ratings.shape[0]

# Appending counts of zero in ratings

df_ratings_cnt = df_ratings_cnt_tmp.append(
    pd.DataFrame({'count': rating_zero_cnt}, index=[0.0]),verify_integrity=True,
).sort_index()
print(df_ratings_cnt)
```

[]:
```
# Since the count is very high and values are not normalizer, we will log of the values
df_ratings_cnt['log_count'] = np.log(df_ratings_cnt['count'])
print (df_ratings_cnt)
```

[]:
```
ax = df_ratings_cnt[['count']].reset_index().rename(columns={'index': 'rating score'}).plot(
    x='rating score',
    y='count',
    kind='bar',
    figsize=(11, 7),
    title='Count for Every Rating Score (in Log Scale)',
    logy=True,
    fontsize=13
)
ax.set_xlabel("Movie rating score")
ax.set_ylabel("No of ratings")
```

+ Code + Markdown

[]:
```
# Checking the rating frequency by moiveId

df_movies_cnt = pd.DataFrame(df_ratings.groupby('movieId').size(), columns=['count'])
df_movies_cnt.head()
```

[]:
```
# Plotting the rating frequency
ax = df_movies_cnt .sort_values('count', ascending=False).reset_index(drop=True).plot(
        figsize=(12, 8),
        title='Rating Frequency of All Movies',
        fontsize=12
    )
ax.set_xlabel("movie Id")
ax.set_ylabel("number of ratings")
```

[]:
```
# Filtering Data

popularity_thres = 50
popular_movies = list(set(df_movies_cnt.query('count >= @popularity_thres').index))
df_ratings_drop_movies = df_ratings[df_ratings.movieId.isin(popular_movies)]
print('Shape of original ratings data: ', df_ratings.shape)
print('Shape of ratings data after dropping unpopular/low rated movies: ', df_ratings_drop_movies.shape)
```

[]:
```
# Getting number of ratings given by every user to find less active users
df_users_cnt = pd.DataFrame(df_ratings_drop_movies.groupby('userId').size(), columns=['count'])
df_users_cnt.head()
```

[]:

```
# Plot rating frequency of all users
ax = df_users_cnt \
    .sort_values('count', ascending=False) \
    .reset_index(drop=True) \
    .plot(
        figsize=(10, 6),
        title='Rating Frequency of All Users',
        fontsize=12
    )
ax.set_xlabel("user Id")
ax.set_ylabel("number of ratings")
```

+ Code + Markdown

[]:

```
# Filtering data
ratings_thres = 50
active_users = list(set(df_users_cnt.query('count >= @ratings_thres').index))
df_ratings_drop_users = df_ratings_drop_movies[df_ratings_drop_movies.userId.isin(active_users)]
print('Shape of original ratings data: ', df_ratings.shape)
print('shape of ratings data after dropping both unpopular movies and inactive users: ', df_ratings_drop_users.shape)
```

[]:

```
# pivot and create movie-user matrix
movie_user_mat = df_ratings_drop_users.pivot(index='movieId', columns='userId', values='rating').fillna(0)

# create mapper from movie title to index using movie ID
movie_to_idx = {
    movie: i for i, movie in
    enumerate(list(df_movies.set_index('movieId').loc[movie_user_mat.index].title))
}
# transform matrix to scipy sparse matrix
movie_user_mat_sparse = csr_matrix(movie_user_mat.values)
```

3. Model defining

[]:

```
# We will be using KNN Algorithm here to recommend movies
# define model
model_knn = NearestNeighbors(metric='cosine', algorithm='brute', n_neighbors=20, n_jobs=-1)
# fit
model_knn.fit(movie_user_mat_sparse)

# If no space on drive error add %env JOBLIB_TEMP_FOLDER=/tmp above this code
```

[]:

```
def fuzzy_matching(mapper, fav_movie, verbose=True):

    match_tuple = []
    # get match
    for title, idx in mapper.items():
        ratio = fuzz.ratio(title.lower(), fav_movie.lower())
        if ratio >= 60:
            match_tuple.append((title, idx, ratio))
    # sort
    match_tuple = sorted(match_tuple, key=lambda x: x[2])[::-1]
    if not match_tuple:
        print('Oops! No match is found')
        return
    if verbose:
        print('Found possible matches in our database: {0}\n'.format([x[0] for x in match_tuple]))
    return match_tuple[0][1]
```

[]:

```
def make_recommendation(model_knn, data, mapper, fav_movie, n_recommendations):

    # fit
    model_knn.fit(data)
    # get input movie index
    print('You have input movie:', fav_movie)
    idx = fuzzy_matching(mapper, fav_movie, verbose=True)
    # inference
    print('Recommendation system start to make inference')
    print('......\n')
    distances, indices = model_knn.kneighbors(data[idx], n_neighbors=n_recommendations+1)
    # get list of raw idx of recommendations
    raw_recommends = \
        sorted(list(zip(indices.squeeze().tolist(), distances.squeeze().tolist())), key=lambda x: x[1])[:0:-1]
    # get reverse mapper
    reverse_mapper = {v: k for k, v in mapper.items()}
    # print recommendations
    print('Recommendations for {}:'.format(fav_movie))
    for i, (idx, dist) in enumerate(raw_recommends):
        print('{0}: {1}, with distance of {2}'.format(i+1, reverse_mapper[idx], dist))
```

4.Testing the Model

+ Code + Markdown

Test 1

[]:
```
my_favorite = 'V for Vendetta'

make_recommendation(
    model_knn=model_knn,
    data=movie_user_mat_sparse,
    fav_movie=my_favorite,
    mapper=movie_to_idx,
    n_recommendations=10)
```

Test 2

+ Code + Markdown

[]:
```
my_favorite = 'Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)'

make_recommendation(
    model_knn=model_knn,
    data=movie_user_mat_sparse,
    fav_movie=my_favorite,
    mapper=movie_to_idx,
    n_recommendations=10)
```

(9)REFERENCES:

- https://www.sciencedirect.com/science/article/pii/S1110866515000341
- https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada
- https://medium.com/analytics-vidhya/recommendation-system-using-collaborative-filtering-cc310e641fde
- https://www.solver.com/k-nearest-neighbors-k-nn-prediction
- Isinkaye, Folasade Olubusola, Yetunde Folajimi and Bolanle Adefowoke Ojokoh. “Recommendation systems: Principles, methods and evaluation.” Egyptian Informatics Journal 16 (2015): 261-273.