

Project Template

**Course Code and Name:** MIT-WPU-MBD-2106 Mini Project (Machine Learning)

**Project Team:**

|  |  |
| --- | --- |
| Roll No | Name |
| 8 | ABHISHEK KASHYAP |
| 15 | AJAY PRAKASH OSTAWAL |

**Project Title:** Movie Recommender System

**Type of the problem:** Recommendation Prediction

**Domain:** Media and Entertainment

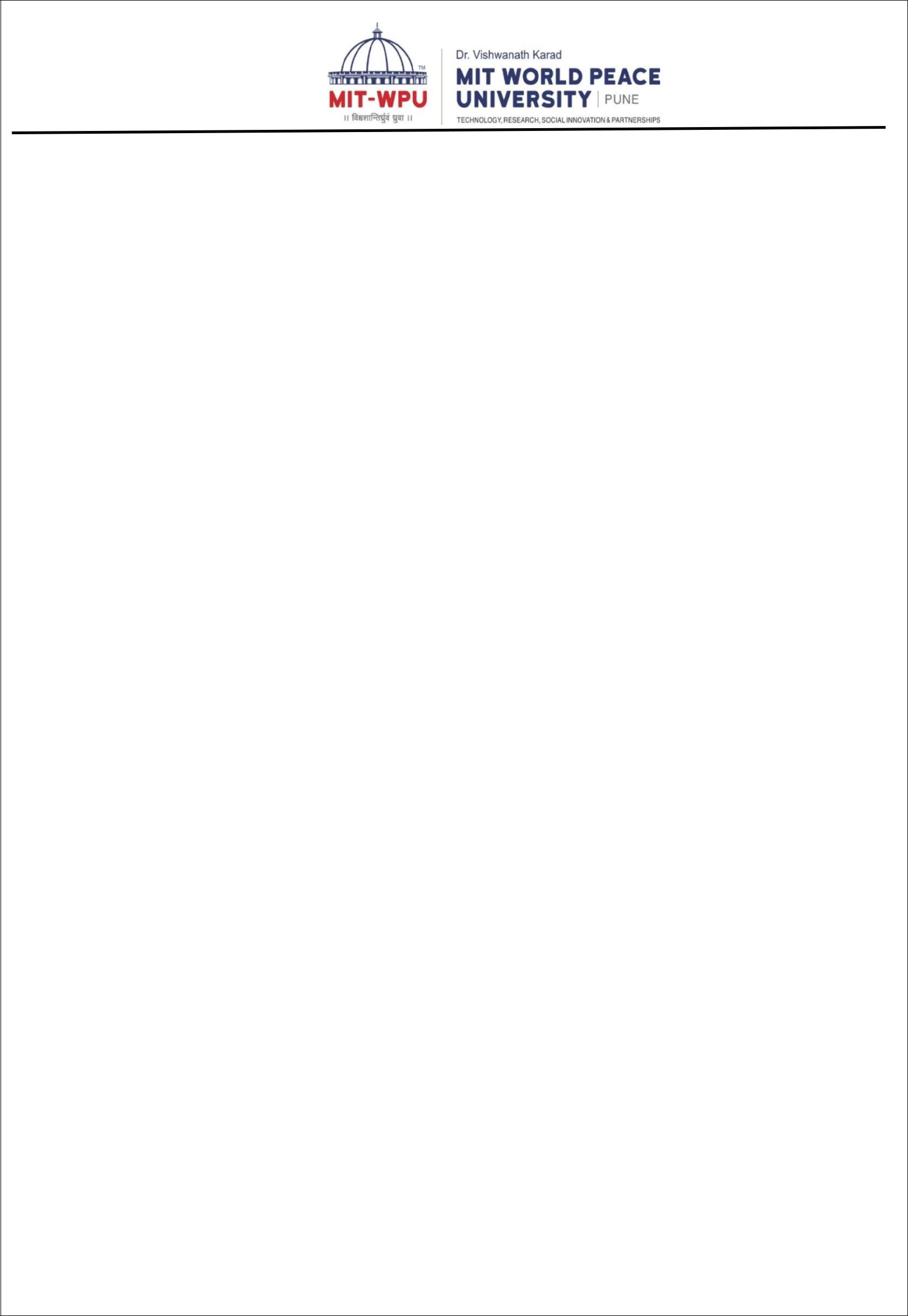
**Problem Description:** To recommend top 10 movies to user without repetition on basis of his interest which he has not seen.



**Related Works/Literature:** Many Data Scientist has done recommendation system on amazon dataset, Netflix dataset, Movie Lens dataset. They all use different approaches for recommendations. After studying few of the research papers we decided to work on Movie Lens dataset for recommending movies.

Below are the URL that we have referred:

1. https://towardsdatascience.com/how-to-build-a-simple-recommender-system-in-python-375093c3fb7d
2. https://www.geeksforgeeks.org/python-implementation-of-movie-recommender-system/
3. https://www.kaggle.com/redroy44/movielens-dataset-analysis
4. https://www.kaggle.com/rounakbanik/movie-recommender-systems



1. https:// www.kaggle.com/petejadhav/recommender-movielens-large-sparsity-0-1/notebook
2. https://towardsdatascience.com/building-and-testing-recommender-systems-with-surprise-step-by-step-d4ba702ef80b
3. https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-recommendation-engine-python/
4. https://www.kaggle.com/laowingkin/netflix-movie-recommendation/comments
5. https://github.com/ajaysh2193/Netflix-Movie-Recommendation-System/blob/master/Netflix.ipynb
6. https://github.com/ramsane/NetflixPrizeRS/blob/master/01\_EDA\_on\_whole\_data\_and\_models\_on\_small\_sample.ipynb
7. https://github.com/anoop93/Netflix\_movie\_recommendation\_system/blob/master/Movie\_Recommendation\_System.ipynb
8. https://github.com/ajaysh2193/Netflix-Movie-Recommendation-System/blob/master/Netflix.ipynb
9. https://www.kaggle.com/laowingkin/netflix-movie-recommendation/comments
10. https://github.com/ajaysh2193/Netflix-Movie-Recommendation-System/blob/master/Netflix.ipynb
11. https://www.kaggle.com/neocryan/amazon-fine-food-recommendation-system-pmf-svd
12. https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-recommendation-engine-python/

**Proposed Tools/Libraries:** For all of the code we are using surprise open source library for building and analysing recommender system.

http://surpriselib.com/

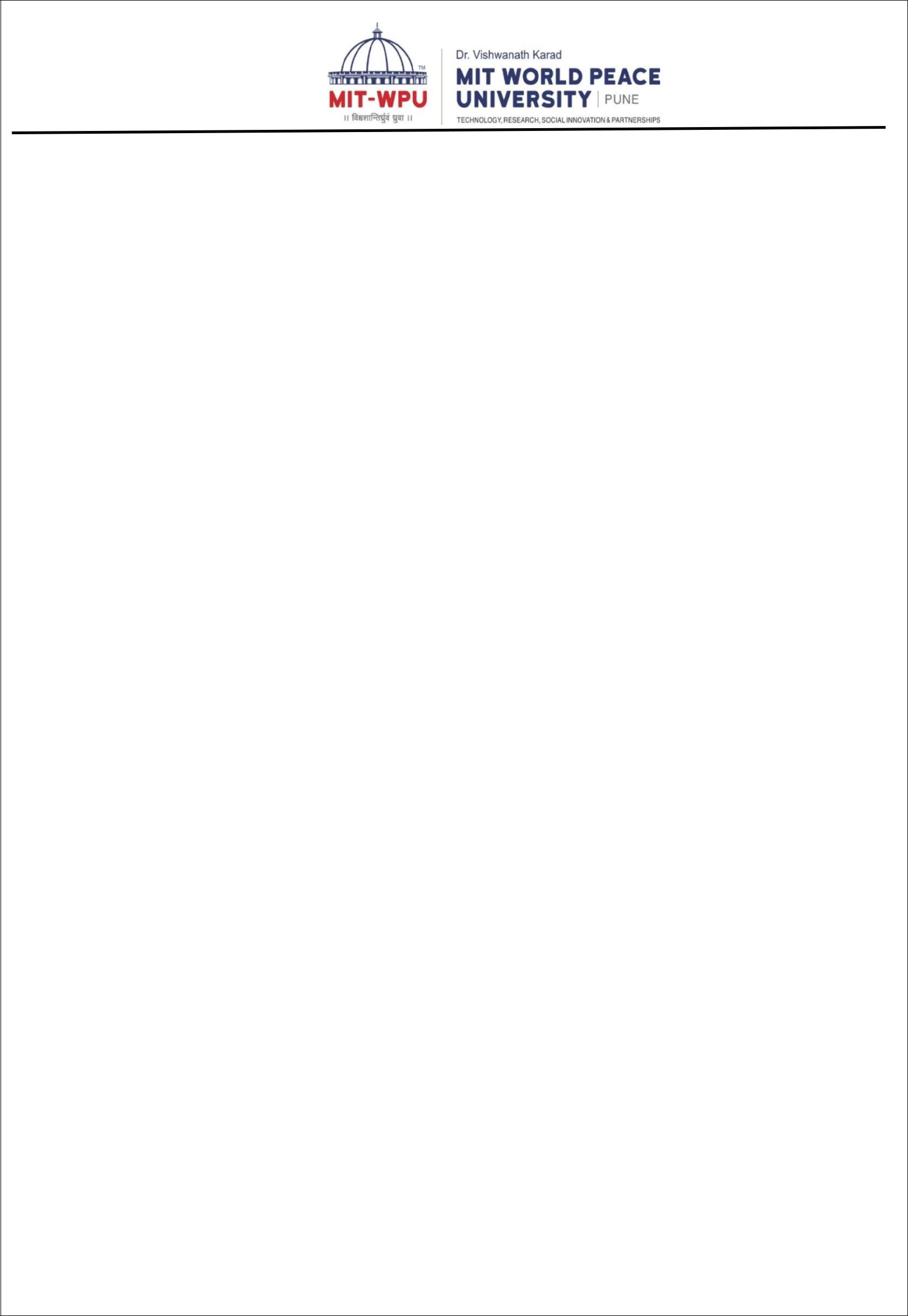
https://surprise.readthedocs.io/en/stable/

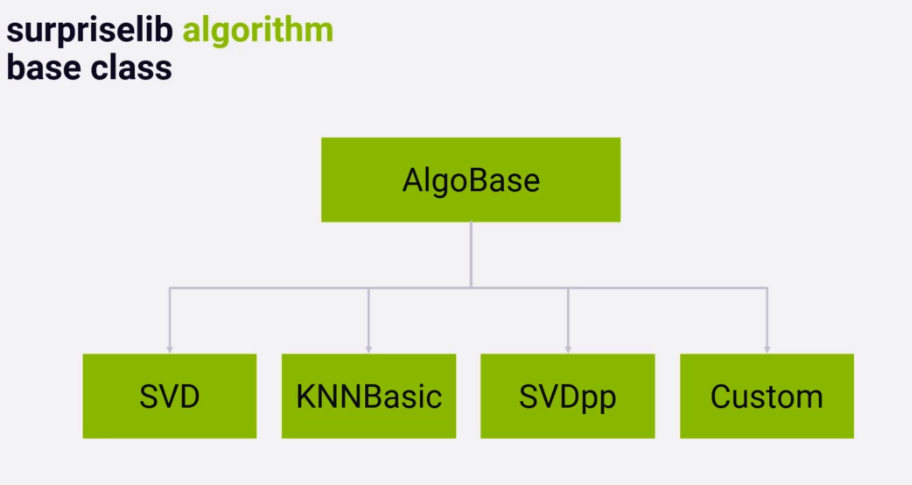
c:\>pip install surprise

* os, sys – operating system
* csv – to read dataset
* re – regular expression
* NumPy – for mathematical operations
* random – random input

Surprise

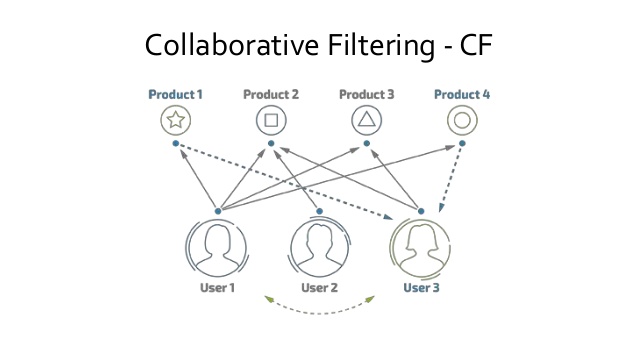
A python scikit for building and analysing recommender systems.





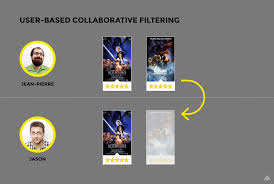
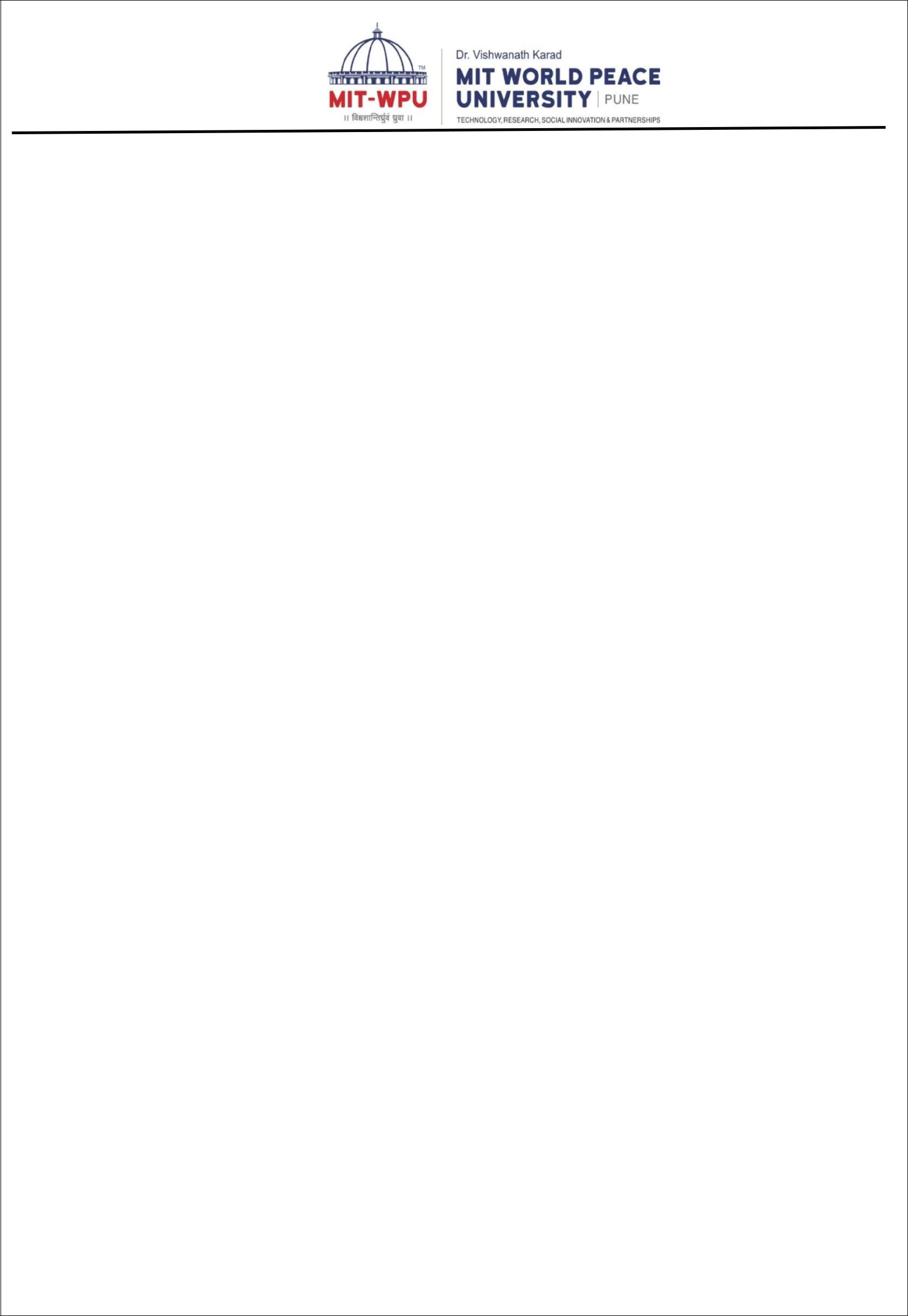
**Proposed Approach and Algorithm:** We are following these below steps to recommend movies to user:

1. **Collaborative Filtering**, on the other hand, doesn’t need anything else except users’ historical preference on a set of items. Because it’s based on historical data, the core assumption here is that the users who have agreed in the past tend to also agree in the future.



1. **USER BASED CF**

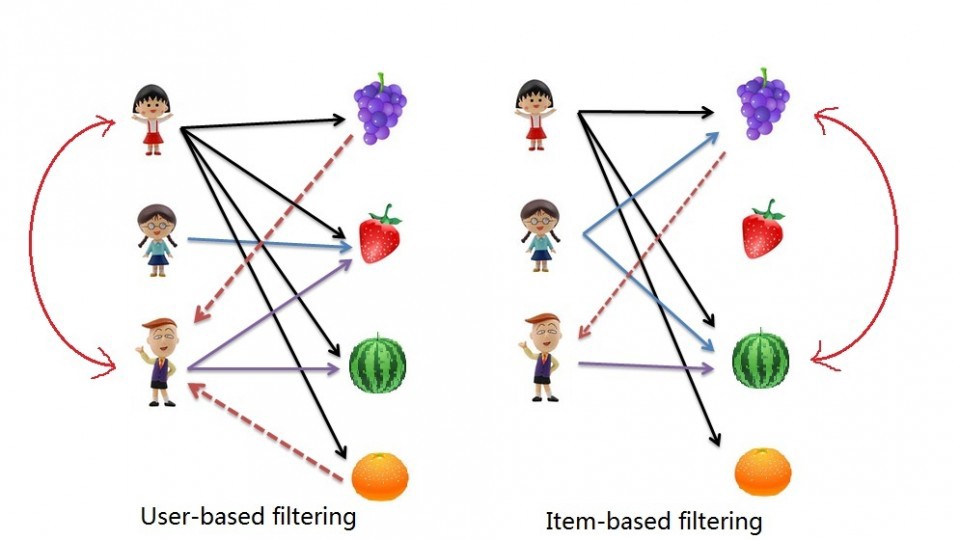
The idea behind user-based collaborative filtering is to find users with similar tastes to our target user. If Jean-Pierre and Jason have rated several films in a similar way in the past, then we consider those two as similar users and we can use the ratings of Jean-Pierre to predict the unknown ratings of Jason.



1. **ITEM BASED CF**

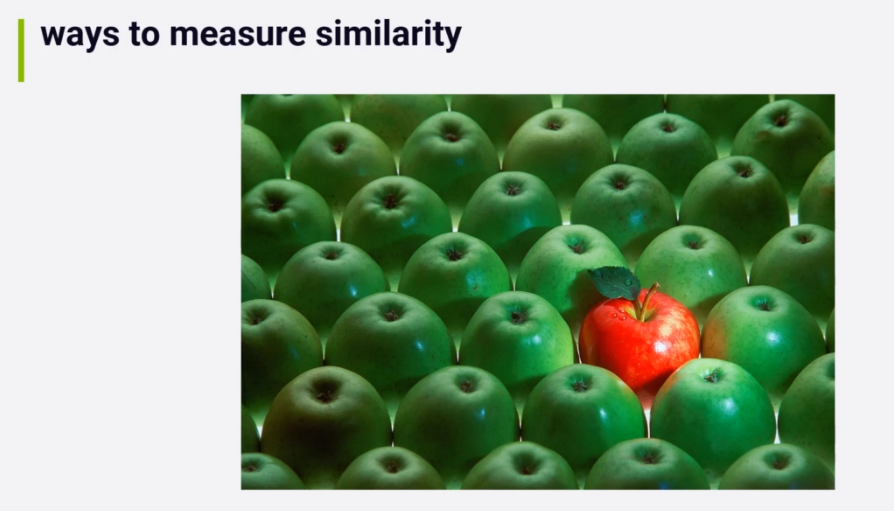
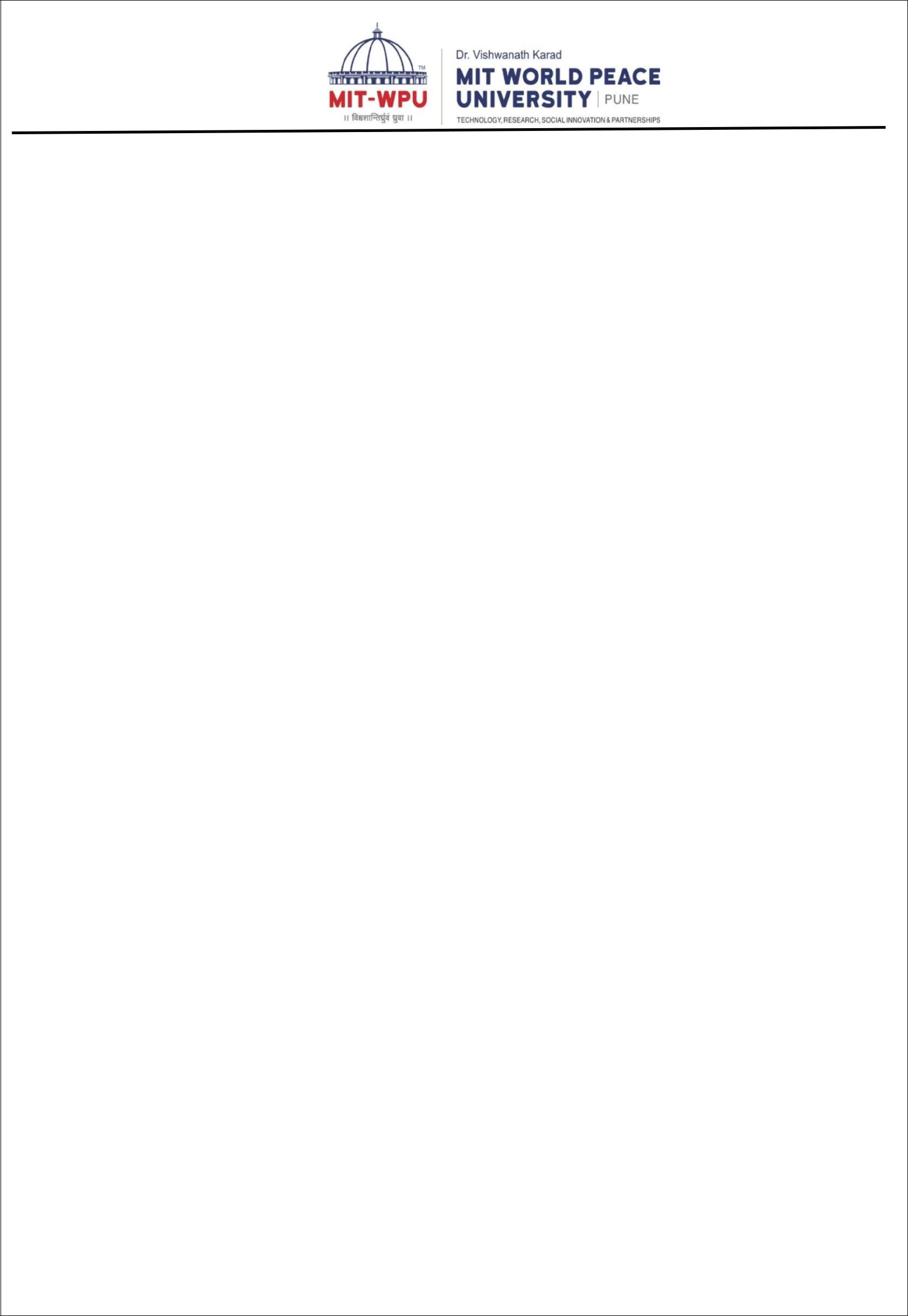
Item based is a form of collaborative filtering for recommender systems based on the similarity between items calculated using people's ratings of those items.

Instead of focusing on his friends, we could focus on what items from all the options are more similar to what we know he enjoys. This new focus is known as Item-Based Collaborative Filtering (IB-CF).



1. **NEAREST NEIGHBOUR**

Neighbourhood recommends the stuff they liked which user haven’t seen yet based on the people’s behaviour simile to you or your neighborhood.

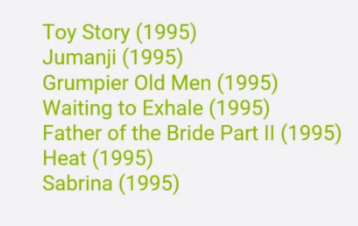


1. **Content based Filtering - KNN**

**K-Nearest-Neighbors (KNN) and Content Recs**

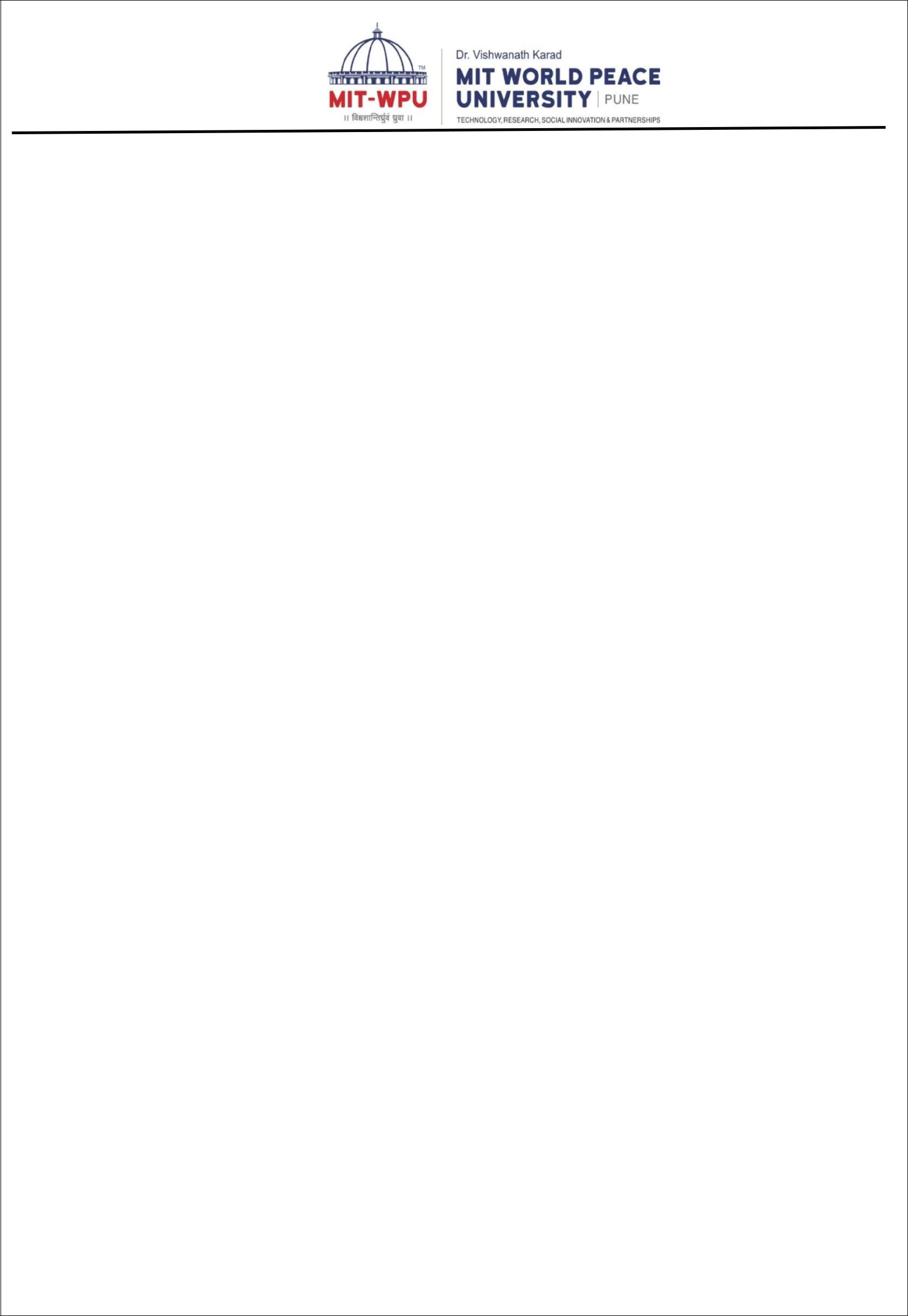
We earlier said that we also consider the release years for each movie. All movie list in the movie-lens data set contains release year at the end of the movie name.

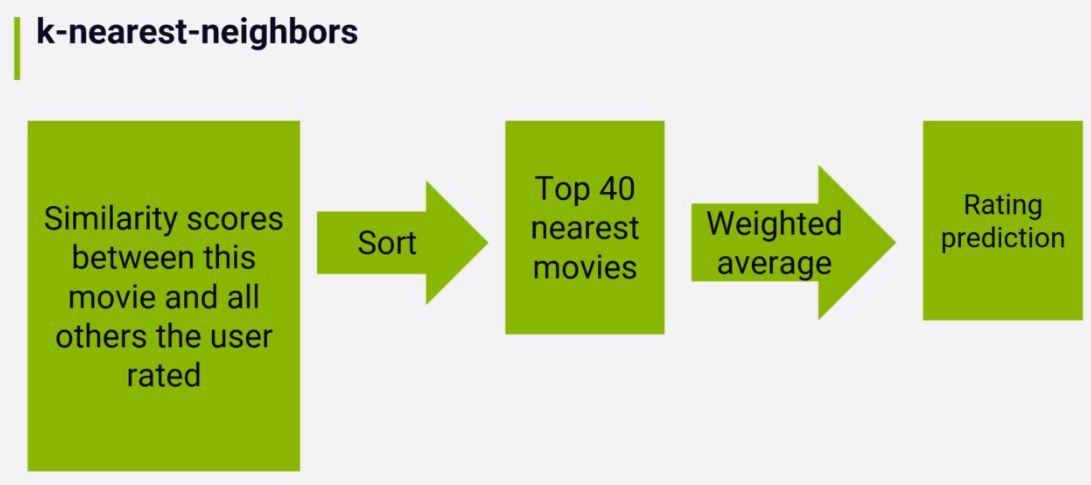
Release years



The recommendation algorithm surprise library has one job to predict rating for given user for given movie.

One of the techniques can be used is K-Nearest-Neighbors (KNN).





1. **Matrix Factorization – SVD**

SVD recommendation based of matrix factorization techniques using PCA (Principal Component Analysis)

Surprise contains a couple of SVD implementations i.e. the regular SVD and slight variant of it called SVDpp (SVD++) for providing recommendations.

The difference in SVD++ is to use a loss function while running stochastic gradient descent.

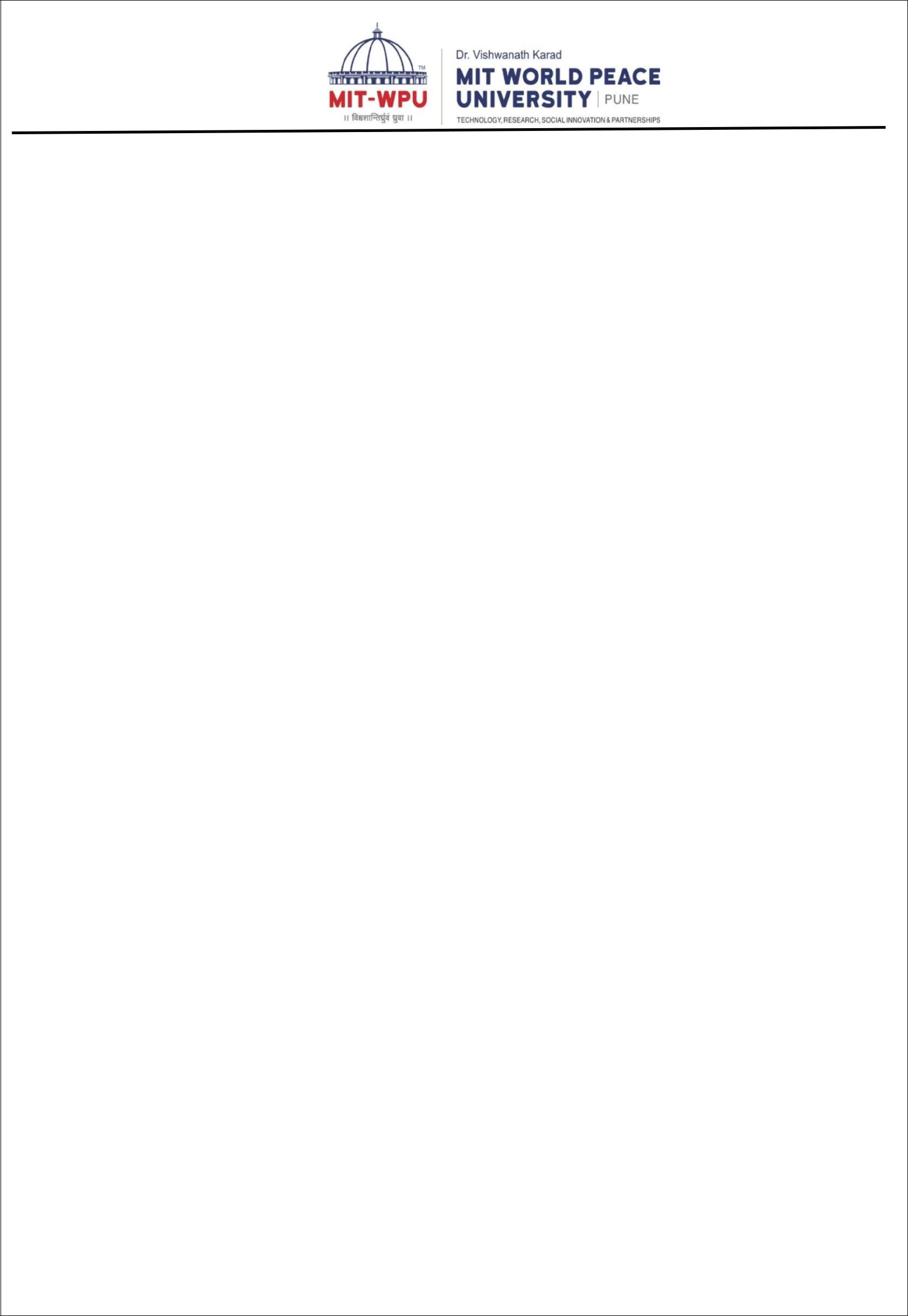
In SVD++ this loss function takes into account the idea that nearly rating an item at all is some sort of implicit interest in the item no matter what the rating was. The implementation is taken cared by the surprise library.

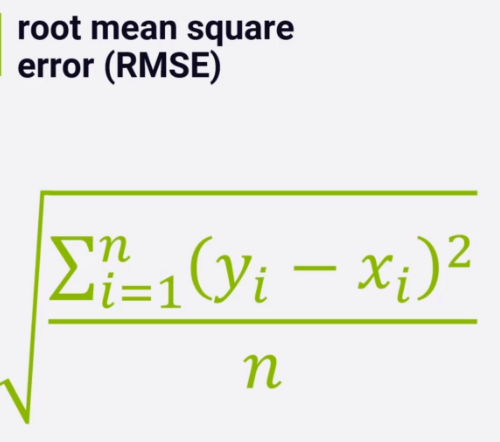
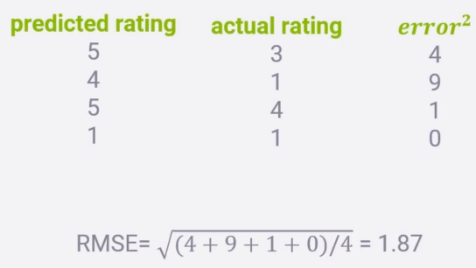
**Model Performance Measurement Approach:**

1. **Root Mean Square Error – RMSE**

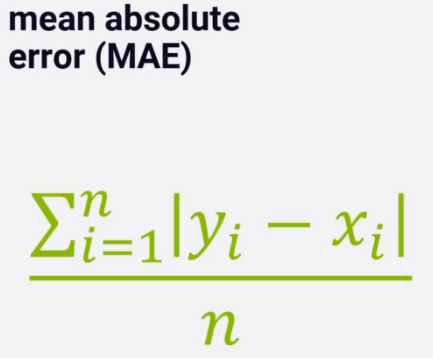
This is more popular metric due to some reasons.

We take the square errors of the differences between predicted rating y and actual rating x, sum them and divide the sum by n to get the mean and finally take the square root of the mean which is RMSE.



1. **Mean Absolute Error – MAE**



**n - n ratings in the test set we want to evaluate.**

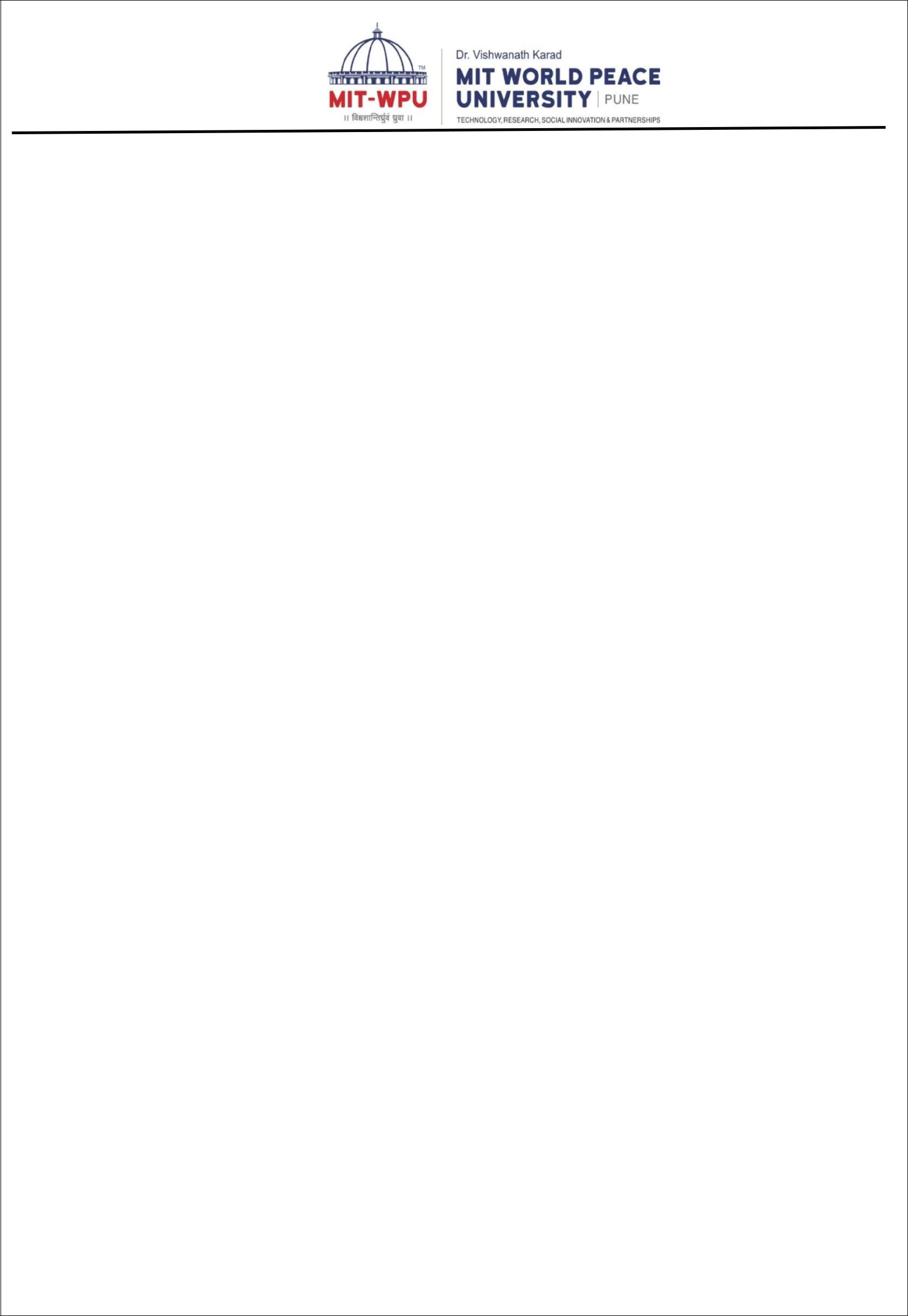
**Y - for each rating our system predicts the rating y.**

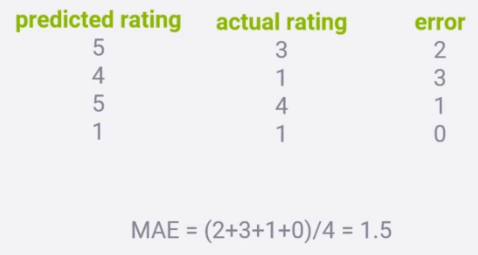
**x - the rating user actually gave is x.**

Just taking absolute value of the difference between the two y and x to measure the error for that rating prediction. It is just the difference between predicted rating and actual rating.

We sum that error result across all n ratings in our test set divided by n to get the average or mean. So, MAE is exactly this.

Error is bad so; we want the lowest MAE score we can get and not the highest.





1. **TOP-N Hit Rate (Evaluating top-N recommenders)**

User actually rated movies will be termed as Hit.

Add up all hits in our top-n recommendations for every user in our test set divide by the numbers of users and that’s out Hit rate.



1. **Leave-one-out cross validation**

Here we cannot use our regular train-test-split or k-fold cross validation approach which we have used to measure accuracy because we are not measuring the accuracy on individual rating instead, we are measuring the accuracy on top-n list of ratings for individual users.

Here we compute the top-n recommendations for each user on training data and intentionally remove one of those items from the user’s training data.

Then we test our recommender systems ability to recommend that item that was left out in the top-n results we create for that user in the testing phase. So, we measure our ability to recommend an item in the top-n list for each user that was left out from training data.

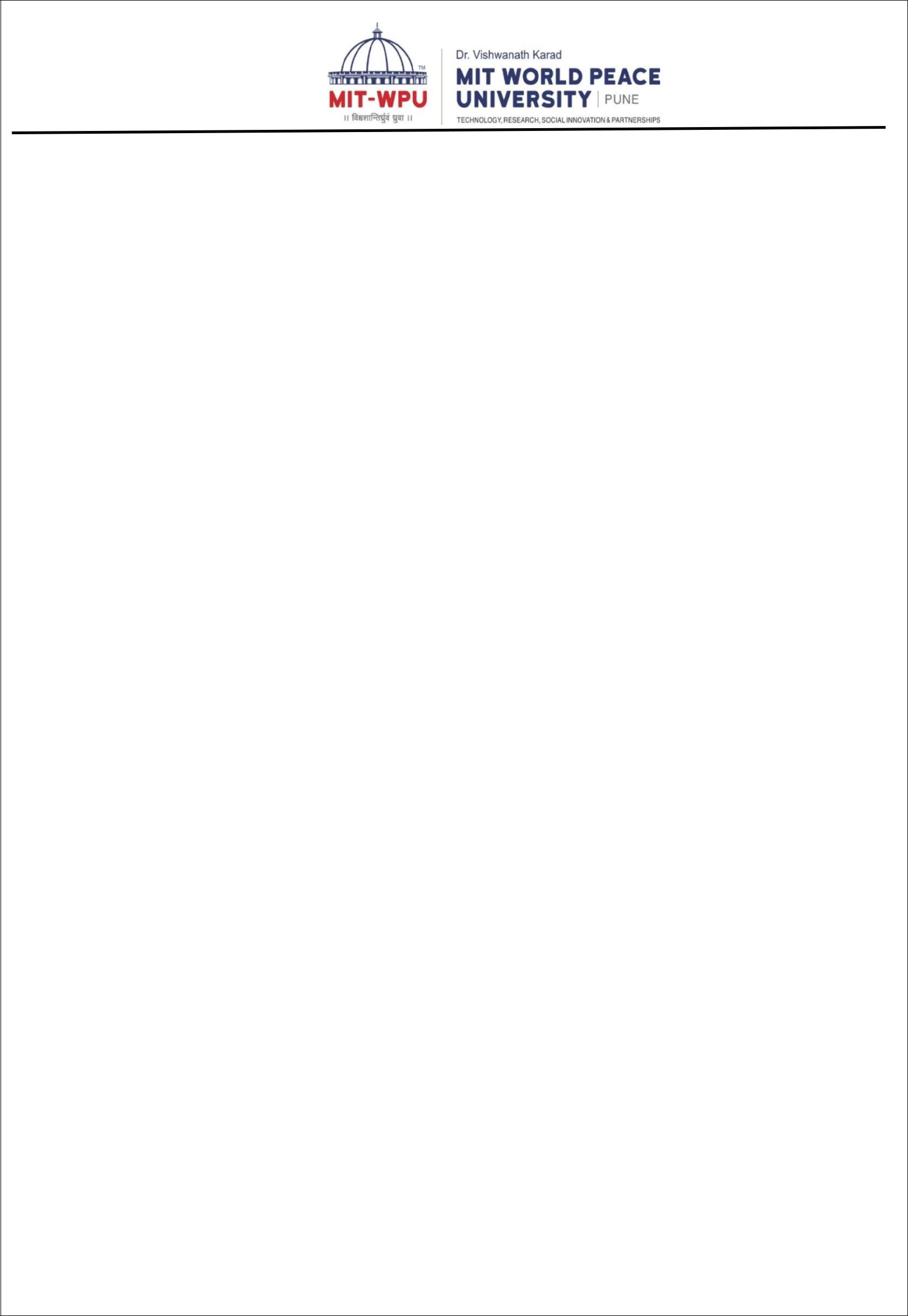
That’s why it is called leave-one-out.

**Dataset / Source:** https://grouplens.org/datasets/movielens/

**Structured/Unstructured data:** Structured Data as Tabular data in csv format

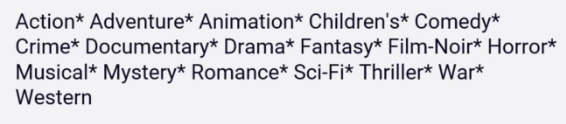
**Dataset Description:** The dataset (ml-latest-small) describes 5-star rating and free-text tagging activity from MovieLens, a movie recommendation service. These data were created by 671 users between January 09, 1995 and October 16, 2016. This dataset was generated on October 17, 2016.

This is a development dataset. As such, it may change over time.

**Initial Data Exploratory:** We played with the data by loading rating and movies data, by analyzing the user ids, movie ids, ratings and also analyzed movies with the genres as part of data analysis.

Data set contains:

* 100004 ratings
* 1296 tag applications across 9125 movies.
* All selected users had rated at least 20 movies.
* Each user is represented by an id
* The data are contained in the files ‘movies.csv’, ‘ratings.csv’
* Only movies with at least one rating or tag are included in the dataset.
* All ratings are contained in the file `ratings.csv`.
* Movie information is contained in the file `movies.csv`.
* Genres are a pipe-separated list, and are selected from the following:



**Assumptions:** As per our problem domain for testing we use with small dataset for 1 lakh records. In real life we can go with 20 million records as per the present dataset available on internet.

We tested the dataset with different approaches and best on the accuracy we can go with the final prediction.

**Pre-processing:** As part of the pre-processing of data two important steps are done:

1. **Candidate Ranking:**

Many candidates will appear more than once and need to combine together in some way. May be boosting their score in the process since they keep coming up repeatedly. After that we sort the resulting recommendations candidates by score, then there are top N list of recommendations.

Many more approaches are existing for just learning the rank:

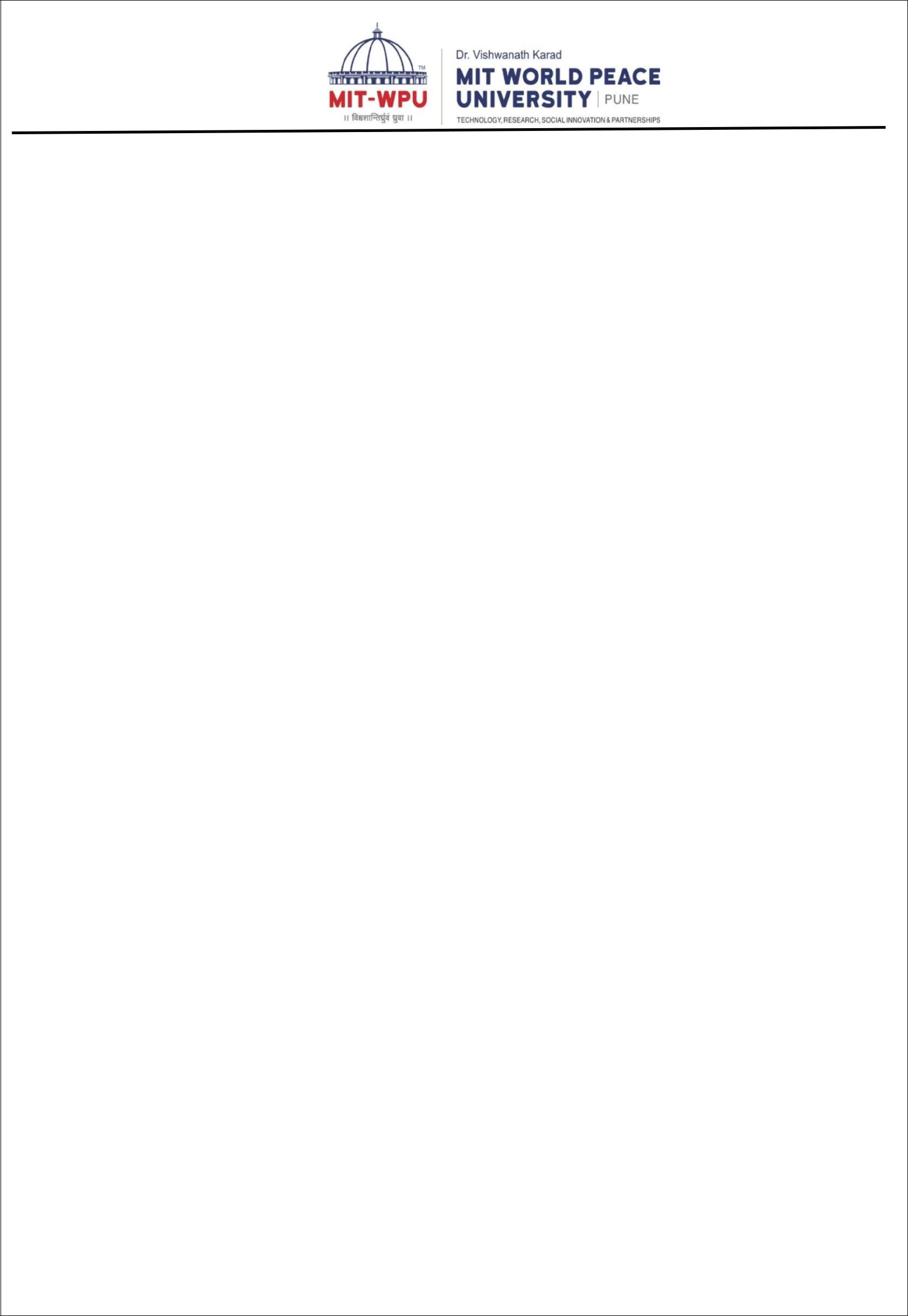
Ex. Machine Learning employed to find the optimal ranking of the candidates.

1. **Filtering:**

Some filtering will be required before presenting the final sorted list of recommendations candidates to the user. This filtering stage is where we might eliminate recommendations for items the user has already rated, since we don’t want to recommend things the user has already seen. We might also apply stop list here to remove items that are potentially offensive to the user.

Or remove items that are below some minimum quality score or minimum rating threshold. It is also where we apply the N in top N recommenders and cut things of if we have more results than we need.

**Feature Engineering:** There are 2 features of this Movie Lens dataset which is the key to approach the results. They are:

1. **User Ids**

Movie Lens users were selected at random for inclusion. Their ids have been anonymized.

1. **Movie Ids**

Only movies with at least one rating or tag are included in the dataset. These movie ids are consistent with those used on the MovieLens web site.

Ratings Data File Structure (ratings.csv)

All ratings are contained in the file ‘ratings.csv’. Each line of this file after the header row represents one rating of one movie by one user, and has the following format:

userId, movieId, rating, timestamp

**Evaluate algorithm:** After implementing different approaches the best recommendations were given by these 2 algorithms:

1. **Content based Recommendations:**

As per our subject for movie recommendations we can provide recommendations based on movie genre like science fiction, romance, horror, children, sports etc.

So, if we a user likes science fiction movie it’s reasonable to recommend other science fiction movies to that user.

We also can use movie release date for recommendations. So, we can recommend movies of other science fiction movies again by filtering them to get movies released in the same month or year.

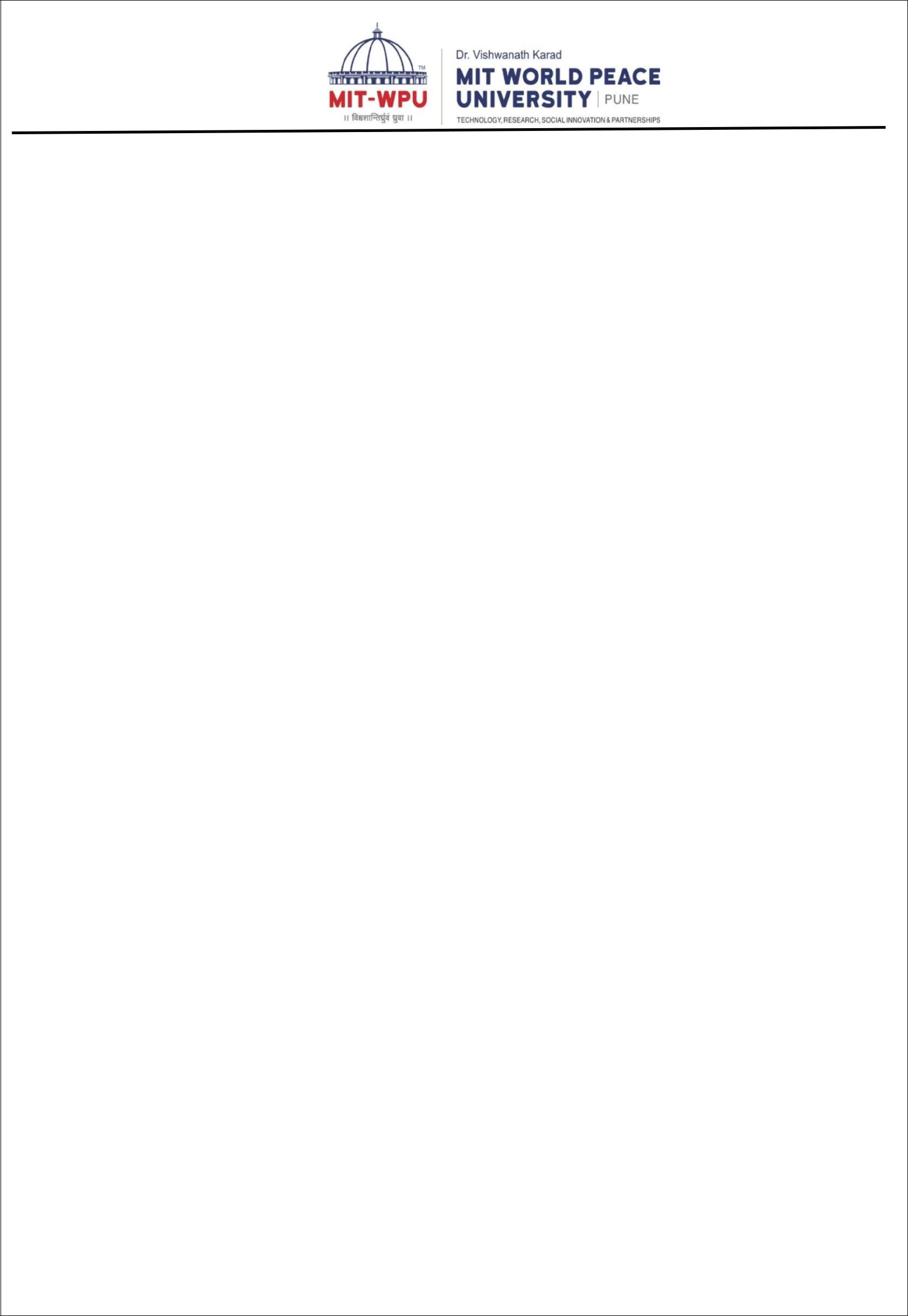
We can also tie each movie on movie lens to other data sets like IMDB to get more details about movie like director, producer, actors etc.

Below is an example of movie genre,

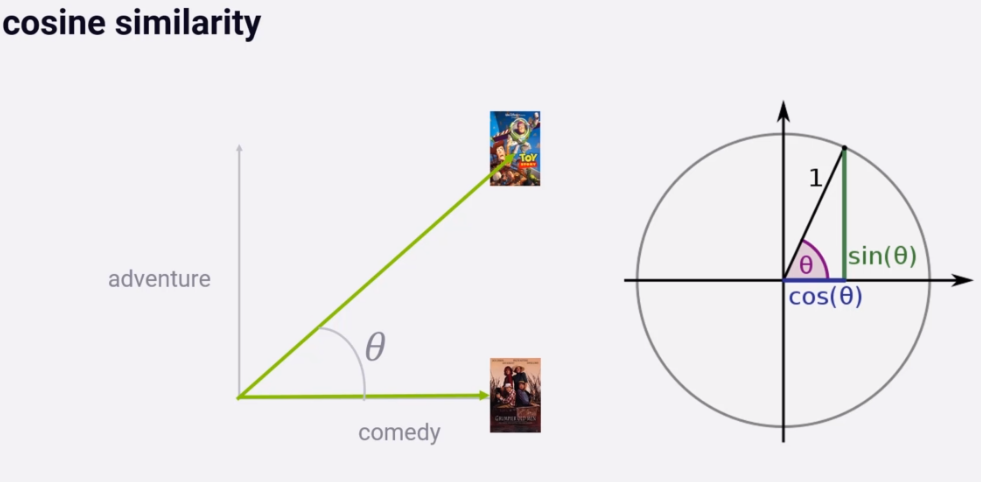


In all there are 18 different possible genres for every movie.

So, we can imagine some sort of similarity measure that looks at how many genres that a given pair of movies has in common.

This can mathematically be done using,

**Cosine Similarity Metric**



1. **SVD (Singular Value Decomposition)**

SVD makes use of PCA matrix factorization on the both the users and items and giving us back the matrices we need that are factors of the ratings matrix we want.



Here we want the complete matrix without missing values and for those missing values we can put in mean values in simple.

But there is one more way to do it,

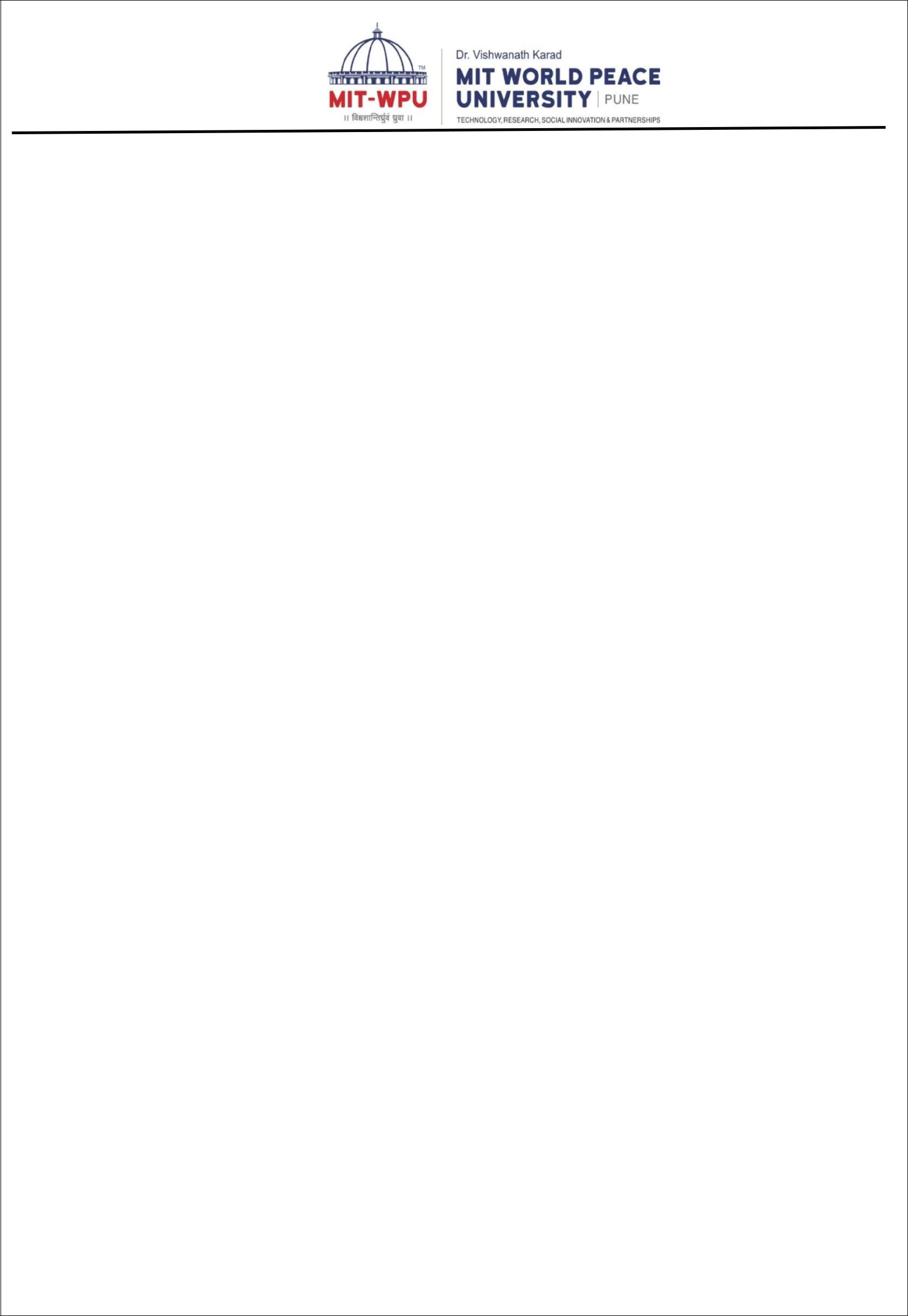
Here each rating can be described as dot product of some row in the matrix U and some column in the matrix M T-transpose.

So, let’s assume we have some rating in column U and M-t, we can treat this as minimization problem where we try to find the values of those complete rows and columns that best minimize the errors of the known rating in R.

For this we can use many ML techniques out of them we can use SGD (Stochastic Gradient Descent). Basically, just keeps iterating in some given learning rate until it arrives at a minimum error value. This can be used to learn best values of those factored matrices when we have missing data.

This is just one way to do it; Apache Spark uses another technique called Alternating Least Squared (ALS).

It’s not real SVD since we can’t do real SVD with missing data, it’s an SVD inspired algorithm.



**Tuning:**

The O/p contains below accuracy where SVD++ have better accuracy than SVD.

Then we work on improving SVD by tuning the hyper-parameters for SVD.

We can go further and tune SVD results by tuning parameters i.e. hyper-parameter tuning.

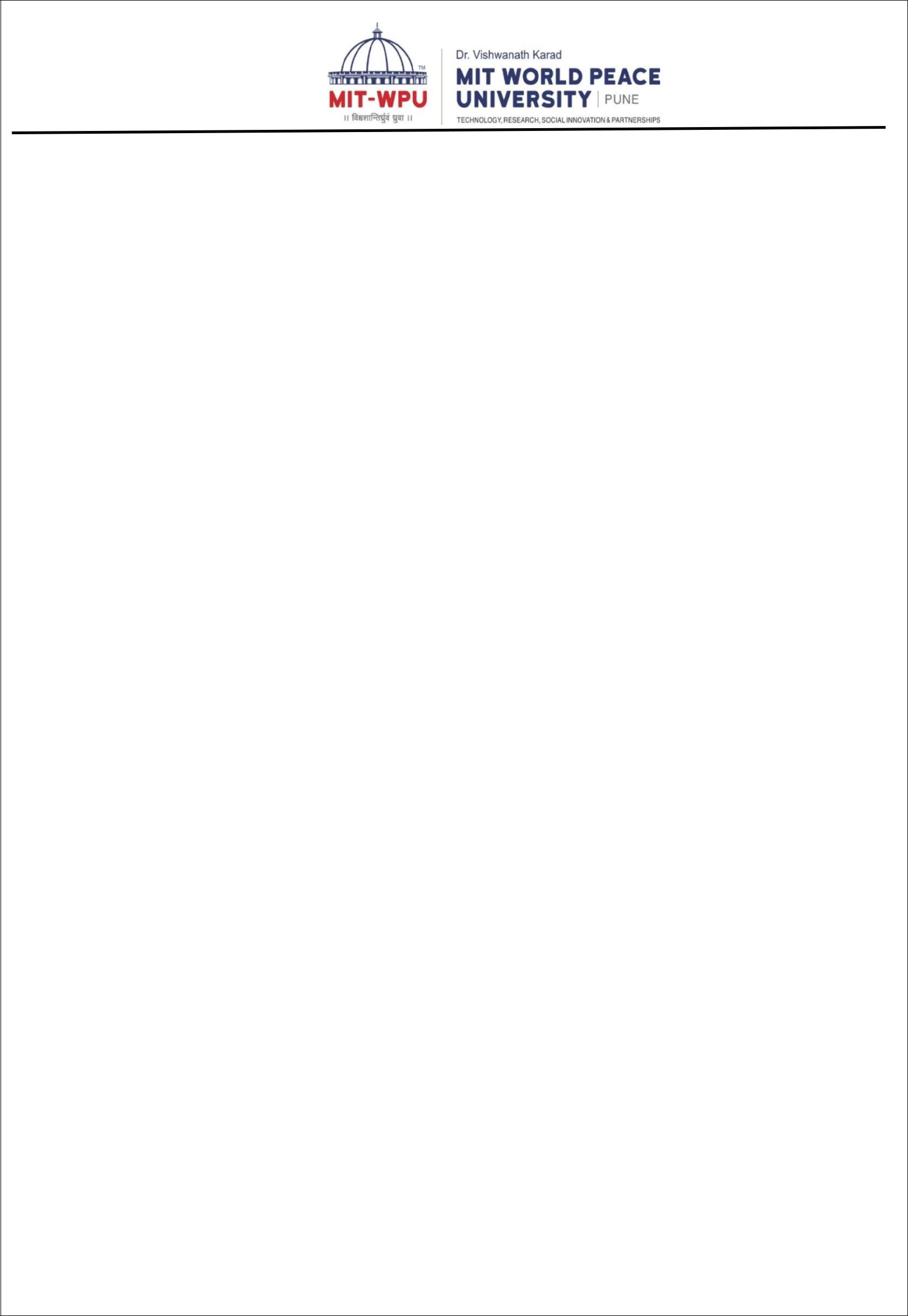
Here the code is almost similar except GridSearchCV object which takes SVD and param\_grid as an argument with measures RMSE and MAE and CV=3.

We settled on 20 epochs and learning rate lr\_all of 0.005 and 50 factors.

The tuned result gives RMSE of 0.9002 over unturned result with RMSE of 0.9033.

The difference is small but it results into very different top N movie recommendations result.

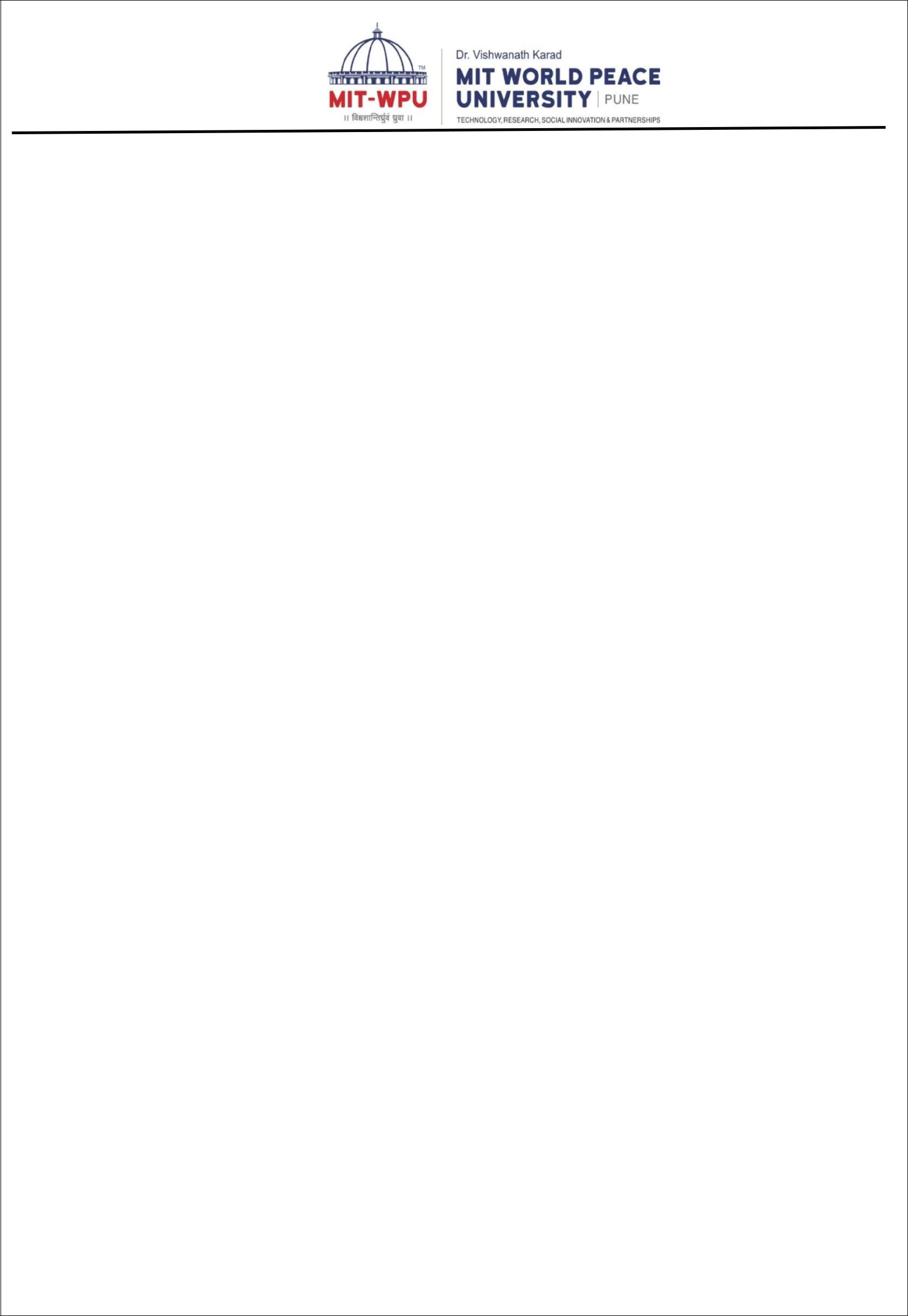
We can consider both these results are good.



**Final Model:** We used Content based filtering as the final model with KNN as an algorithm for recommendation.

**Conclusion:** After implementing all algorithms, we got the overall performance that Content Based Technique and SVD are the strong performers and we can choose either of them based on the dataset.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_End of Project Report Template \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Project Evaluation / Assessment (% weightage of marks)

Name of the Student:

**ABHISHEK KASHYAP - 8**

**AJAY PRAKASH OSTAWAL - 15**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Roll No | Quality of the Problem Definition | Project Report / Documentation | Data Exploratory Analysis | Visualization | Implementation | Testing | Presentation / Explanation | Q&A |
|  | 10 Marks | 20 Marks | 10 Marks | 10 Marks | 20 Marks | 10 Marks | 10 Marks | 10 Marks |
| 8 |  |  |  |  |  |  |  |  |
| 15 |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

Signature of the Guide Signature of the Program Head