MOVIE RECOMMENDATION SYSTEM

NIRMA UNIVERSITY
INSTITUTE OF TECHNOLOGY
NAAC ACCREDITED 'A' GRADE

KISHAN MISTRI (14BIT029)
Guided By:
PROF. SAPAN MANKAD

Objective

- This system will make predictions based on user's past activity with some of the examples and generate series of movie, which would be calculated mathematically in order to engage user with system.
- Movie recommendation system application will provide ease of navigation in logical manner for swift way of experience for user.
- Provide quality recommendations in real time from user experience.

Scope

- Movie recommendation web application is designed in python web framework Django, where necessary pages are designed. Here large dataset is already fitted into our database, which will have large pool of choices for user.
- Newly arrival movies will be automatically added through API used during development.

Tools & Technologies

Anaconda Django PostgreSQL TMDB API

Datasets

MovieLens

➔ It is a data set that provides 1,00,00,054 user ratings.
➔ 95,580 tags applied to 10,681 movies by 71,567 users.

Netflix dataset

➔ 4,80,189 users ➔ 17,770 movies ➔ 100 M+ ratings

Algorithms Compared

➔ SVD ➔ SVD++ ➔ Co-Clustering CF

Primary Database Schemas

```
majorproject-# \d movietitles
                Table "public.movietitles"
  Column   |  Type   | Collation | Nullable | Default
-----------+---------+-----------+----------+---------
 movieid   | integer |           | not null |
 yor       | integer |           |          |
 moviename | text    |           |          |
Indexes:
    "movietitles_pkey" PRIMARY KEY, btree (movieid)
```

```
majorproject-# \d rating
                Table "public.rating"
 Column  |  Type   | Collation | Nullable | Default
---------+---------+-----------+----------+---------
 movieid | integer |           |          |
 custid  | integer |           |          |
 rating  | real    |           |          |
 dor     | text    |           |          |
```

Comparison of models

- All comparisons have their limits in terms of space or time. But here it will need time parameter further divide into parts:
  Training & Predicting time.
- Here we find tendencies based algorithm more relevant but it's a list used algorithms among systems.
- Widely used algorithms always consists of CF algorithms which is more logical as prime step.
- Probabilistic algorithms gives more accuracy, but consumes more time to train

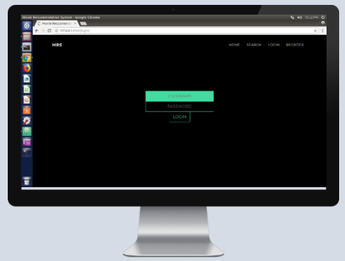
Algorithms	Training	Prediction
User based	-	O(mn)
Item based	$O(mn^2)$	O(n)
Similarity fusion	$O(n^2m+m^2n)$	O(mn)
LSI/SVD	$O((m+n)^3)$	O(1)
NSVD2	O(mnk)	O(1)
Cluster based Smoothing	$O(mn^2k)$	O(mn)
Tendencies-based	O(mn)	O(1)

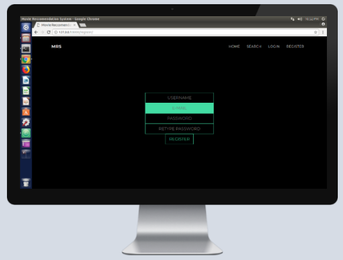
Own Algorithm Function

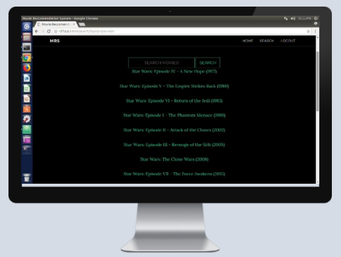
In implementation of algorithms, all items x Users matrix is checked with every other user and find the similar ones and keep record of top 500 neighbors.
Now these neighbors will calculate missing values of individual user same as normal CF, but it will be assigned weight based on similarity to user and then final missing value is filled with weighted value.

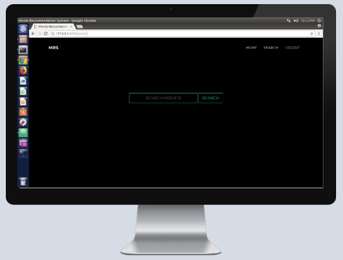
Accuracy gained using:	Train-RMSE	Test-RMSE
SVD:	0.8355	0.9546
SVD ++ :	0.8900	0.9356
Weighted Co-Clustering:	0.9278	0.9567

Implementation Screenshots


Login Page


Register Page


Recommendations Page


Search Page

Future Enhancements

- Cloud infrastructure will help to boost execution in fraction of seconds.
- Facebook, Instagram or any other social media platform linked with project help more accurate result and notice implicit activity.
- Implicit response is becoming more popular on every recommender platform where large amount of data is involve and can give insight to virtual preference of user by creating identity for predictions.