

MOVIE RECOMMENDATION SYSTEM

IT7611-CREATIVE AND INNOVATIVE PROJECT

A Project Report

Submitted by

SRIMATHI R

GOPI A

KANNAN B S



DEPARTMENT OF INFORMATION TECHNOLOGY

MADRAS INSTITUTE OF TECHNOLOGY CAMPUS

ANNA UNIVERSITY:CHENNAI 600 044

MAY 2021

ANNA UNIVERSITY: CHENNAI 600044

BONAFIDE CERTIFICATE

Certified that this creative and innovative project report
“MOVIE RECOMMENDATION SYSTEM” is a bonafide work of
**“SRIMATHI (2018506125),GOPI A (2018506033), KANNAN BS
(2018506047)”** who carried out this project work under my
supervision.

SIGNATURE

Dr. M.R SUMALATHA

Dr.P.LAKSHMI HARIKA

SUPERVISOR

Department of Information Technology

MIT Campus, Anna University

Chennai- 600 044

ACKNOWLEDGEMENT

It is essential to mention the names of the people, whose guidance and encouragement helped us to complete this project.

We express our gratitude to our project guide, **Dr. M.R Sumalatha**, Assistant Professor (Sl grade) and **Dr.P.LakshmiHarika**, Teaching Fellow, Department of Information Technology, Anna University, for guiding and assisting us throughout this project.

Our sincere thanks to **Dr. Dhananjay Kumar**, Head of the Department of Information Technology, MIT Campus, for catering to all our needs and supporting us throughout this project.

We express our gratitude to our respected Dean of MIT Campus, **Dr. T. Thyagarajan**, for providing excellent computing facilities to complete this project.

SRIMATHI R 2018506125

GOPI A 2018506033

KANNAN B S 2018506047

TABLE OF CONTENTS

CHAPTERNO	TITLE	PAGENO
	ABSTRACT	6
	LIST OF FIGURES	7
	LIST OF TABLES	8
	LIST OF ABBREVIATIONS	9
1	INTRODUCTION	10
	1.1 OVERVIEW	10
	1.2 CHALLENGES	10
	1.3 OBJECTIVE	12
	1.4 PROBLEM STATEMENT	12
	1.5 SCOPE OF THE PROJECT	12
	1.6 CONTRIBUTION	13
	1.7 MOVIE RECOMMENDATION SYSTEM	13
	1.8 APPLICATION OF MRS	14
	1.9 ORGANISATION OF THE REPORT	15
2	LITERATURE SURVEY	16
3	SYSTEM DESIGN AND ARCHITECTURE	19
	3.1 INTRODUCTION	19
	3.1 SYSTEM ARCHITECTURE	19
	3.2 SYSTEM DESIGN	20
	3.4 SOFTWARES USED	23

	3.5 SUMMARY	24
4	IMPLEMENTATION	24
	4.1 DATASET	24
	4.2 PREPROCESSING	26
	4.3 AUTOENCODERS	27
	4.3.1 ALGORITHM	27
	4.3.2 STACKED AUTOENCODERS MODEL	28
	4.3.3 TRAINING SAE	31
	4.3.4 TESTING SAE	33
	4.4 SVD MODEL	34
	4.5 CNN MODEL	36
	4.6 HYBRID MODEL	37
	4.7 FRONT-END FRAMEWORK	37
5	EVALUATION AND RESULTS	40
	5.1 PERFORMANCE ANALYSIS	40
	5.2 NOVELTY OF THE SYSTEM	45
6	CONCLUSION AND FUTURE WORK	46
	6.1 CONCLUSION	46
	6.2 FUTURE WORK	46
	REFERENCES	47
	APPENDIX-1	48
	APPENDIX-2	51

ABSTRACT

Recommendation systems are an important part of suggesting items in streaming services. For streaming movie services like Netflix, recommendation systems are essential for helping users find new movies to enjoy. We propose a **deep learning approach** based on **Deep autoencoders** and **Convolutional Neural Networks (CNN)** to produce a **collaborative filtering** system which predicts movie ratings for a user based on a large database of ratings from other users. We use concept of deep learning to predict users' ratings on new movies. The model uses model based Collaborative Filtering, **Single Value Decomposition(SVD)** a matrix factorisation technique, which reduces the number of features of a dataset by reducing the space dimension from N-dimension to K-dimension (where $K < N$). Input is Movies and Ratings table which contains rating of users on various movies and the output will be the Top Recommended movies for the user predicted by the System.

LIST OF FIGURES

FIG NO	TITLE	PAGE NO
FIG 3.1	SYSTEM ARCHITECTURE	20
FIG 3.2	AUTOENCODERS ARCHITECTURE	21
FIG 5.2	SAE EPOCH VS TRAINING LOSS PLOT	41
FIG 5.3	AUTOENCODER EPOCH OUTPUT	41
FIG 5.5	K CROSS FOLD VALIDATION	42
FIG 5.7	CNN LOSS PLOT	43
FIG 5.8	GENRE DISTRIBUTION	44
FIG 5.9	MODEL VS LOSS COMPARISON	44

LIST OF TABLES

TABLE NO	TITLE	PAGE NO
5.1	SAE TRAINING LOSS COMPARISON	40
5.4	SVD K FOLD CROSS VALIDATON	42
5.6	CNN LOSS TABLE	43

LIST OF ABBREVIATONS

ABBREVIATION	FULL FORM
CF	Collaborative Filtering
MRS	Movie Recommendation System
CNN	Convolutional Neural Network
SVD	Single Value Decomposition
SAE	Stacked Auto Encoders
MSE	Mean Squared Error

CHAPTER 1

INTRODUCTION

1.1. Overview:

This recommendation system recommends different movies to users. System which is based on a collaborative approach computes the connection between different clients and relying upon their ratings, prescribes movies to others who have similar tastes,. It is a web application that allows users to rate movies as well as recommends them appropriate movies based on other's ratings. Input is Movies and Ratings table which contains rating of users on various movies and the output will be the Top Recommended movies for the user predicted by the System.

1.2. Challenges:

1) Cold start

This problem arises when new users /items are added to the system, a new item can't recommend to users initially when it is introduced to the recommendation system without any rating or reviews and hence it is hard to predict the choice or interest of users which leads to less accurate recommendations.

A new user or item added based problem is difficult to handle as it is impossible to obtain a similar user without knowing previous interest or preferences.

2)Sparsity

It happens when most of the users do not give ratings or reviews to the items they purchased and hence the rating model becomes very sparse.It leads to data

sparsity problems. It decreases the possibilities of finding a set of users with similar ratings or interest.

3)Synonymy

Synonymy arises when a single item is represented with two or more different names or listings of items having similar meanings, the recommendation system can't recognize whether the terms shows various items or the same item.

4)Privacy

An individual needs to feed his personal information to the recommendation system for more beneficial services but it causes the issues of data privacy and security, many users feel hesitation to feed their personal data into recommendation systems that suffer from data privacy issues.

The recommendation system is bound to have the personal information of users and use it to the fullest in order to provide personalized recommendation services. So, the recommendation systems must ensure trust among their users.

5)Scalability

A huge changing data is generated by user-item interactions in the form of ratings and reviews and consequently, scalability is a big concern for these datasets. Recommendation systems interpret results on large datasets inefficiently, some advanced large-scaled methods are required for this issue.

1.3. Problem Statement:

Given a set of users with their previous ratings for a set of movies, we predict the rating user would give to movies which they have not rated.

1.4. Objective:

The Movie Recommendation System provides a mechanism to help users categorize users with similar interests. It involves a number of factors to create personalized lists of useful and interesting content specific to each user/individual. Recommendation systems are Deep Learning based algorithms that skim through all possible options and create a customized list of items that are interesting and relevant to an individual. These results are based on their what other people with similar demographics are watching, and how likely are you to watch those movies. This is achieved by applying item-based collaborative filtering to Autoencoders and Convolution Neural Network.

1.5. Scope:

Recommendation systems are important for **movie streaming services** like Netflix, Amazon Prime in helping users to discover new content to enjoy.

Number of choices for movies on internet is very high .It's tedious to **refine most wanted data** by self while searching. The scope of this proposal system makes it easy to work with such numerous data.

People have problem selecting movie due to lack of time and search issues. The system **saves User's time**.

Many mobile phone and limited processing power computers can't handle recommender system due to its **extremely large dataset**. The solution opted is use of **web services**.

1.6 CONTRIBUTIONS

Our contributions can be summarised as follows:

The general sign language recognition models uses Collaborative Filtering (CF) and Content Based Filtering (CBF) for recommendation systems. But this model uses Autoencoders to produce CF and Convolution Neural Networks(CNN) .This has greatly improved the results and the words recognised are accurate.

1.7 MOVIE RECOMMENDATION SYSTEM

A recommendation system is a type of information filtering system which assume the priorities of a user, and make recommendations on the basis of user's priorities The popularity of recommendations systems have gradually increased and are recently implemented in almost all online platforms that people use. The content of such system differs from films, podcasts, books and videos, to colleagues and stories on social media, to commodities on e-commerce websites, to people on commercial websites. Every user has different likes and dislikes. In addition, even the taste of a single customer can differ depending on a large number of aspects, such as mood, season, or type of activity the user is performing.

Our proposed Movie Recommendation System is based on **Collaborative Filtering** Approach (Cluster alike users together and use data about the group to make recommendations to the customer). We use Deep Learning **Unsupervised Approach (Autoencoders)** to produce a collaborative filtering system which predicts movie ratings for a user based on a large database of ratings from other users. We compare our approach to standard collaborative filtering techniques: **Matrix-factorization (SVD) model** which reduces the number of features of a dataset by reducing the space dimension from N-dimension to K-dimension (where $K < N$) and **Neural Networks based CF system**.

Input is Movies and Ratings table which contains rating of users on various movies and the output will be the **Top Recommended movies for the user** predicted by the System.

1.8 APPLICATIONS OF MOVIE RECOMMENDATION SYSTEM

1. Helps user discover new movies based on their previous favourites.
2. Makes it easier for user by reducing their effort by refining the data ,by recommending movies predicted by the system.
3. Saves time for users by avoiding the search time.

1.9 ORGANIZATION OF THE REPORT

The rest of the thesis is as follows:

Chapter 2 analyses various methods and notions from a number of international conference papers and journal papers.

Chapter 3 describes System architecture explaining the parts present in the architecture. -

Chapter 4 describes the implementation.

Chapter 5 describes the results and its discussions.

Chapter 6 presents the conclusion of the project along 4 with the future enhancements that can be applied to the existing work. Finally, it is followed by a Reference Section containing a catalogue of the papers listed in the thesis along with the authors and the year of publication.

CHAPTER 2

LITERATURE SURVEY

[1] Recommender system using fuzzy(2015) by Hirdesh Shivhare et. Al Provides a combinatorial approach by combining fuzzy c means clustering technique and genetic algorithm based weighted similarity measure to develop a recommender system (RS). The proposed FCMGENSM recommender system provides better similarity metrics and quality than the ones provided by the existing GENSM recommender system .They provide better similarity metrics and quality.They have more computational time which is the limitation of system.

[2] An Improved Collaborative MRS using computational intelligence by Zan Wang et. al.(2014) proposed a hybrid model-based movie recommendation system which utilizes the improved K-means clustering coupled with genetic algorithms (GA) to partition transformed user space.In this way, the original user space becomes much denser and reliable, and used for neighborhood selection instead of searching in the whole user space. By this proposed method it will capable of generating effective estimation of movie ratings for new users via traditional movie recommendation systems Hybrid approach reduces computational complexity by data reduction technique.

[3] Reviewing cluster based collaborative filtering approach by F.Darvishi-mirshekarlou et. al.(2013) gave the review about the Recommender systems using collaborative filtering. These users have the same preferences and are interested in it in the past. Scalability is the major challenge of collaborative filtering. With regard to increasing customers and products gradually, the time consumed for finding nearest neighbour of target user or item increases, and consequently more response time is required. It reduces response time and increase stability by clustering algorithms.

[4] Maximum entropy approach by D. Y. Pavlov and D. M. Pennock proposed a approach Clustering of items based on user access path in order to reduce the apriori probability. This helps in addressing sparsity and dimensionality reduction.

[5] Hybrid personalized Recommender by Subhash K. Shindeet proposed a novel modified fuzzy C- means clustering algorithm which is used for hybrid personalized recommender system. It works in two phases. In the first phase opinions from users are collected in form of user item rating matrix. In second phase recommendations are generated online for active users using similarity measures .It proposed coefficient parameter for similarity computation, gave effective and quality recommendation.

[6] Recommendation system using fuzzy C-means Clustering by Kwoting Fanget.al. proposes a recommender system with two approaches on user access behaviour by fuzzy clustering method. It shows similar preference to the given users and recommends what they have liked. The recall value was used to evaluate recommended accuracy with two algorithms (item-to-item) and (user-to-user). Gives valuable information to administrator manage website efficient and users access the interested pages quickly

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

3.1 INTRODUCTION:

Deep artificial neural networks(Deep Learning) is able to learn complex decision boundaries for classification or complex non-linear regressions.

By stacking large numbers of hidden layers in these networks,deep neural networks can learn complex functions by learning to extract many low level features from the data and composing them in useful non-linear combinations.

It is a web application that allows users to rate movies as well as recommends them appropriate movies based on other's ratings. Input is Movies and Ratings table which contains rating of users on various movies and the output will be the Top Recommended movies for the user predicted by the System.

3.1 SYSTEM ARCHITECTURE:

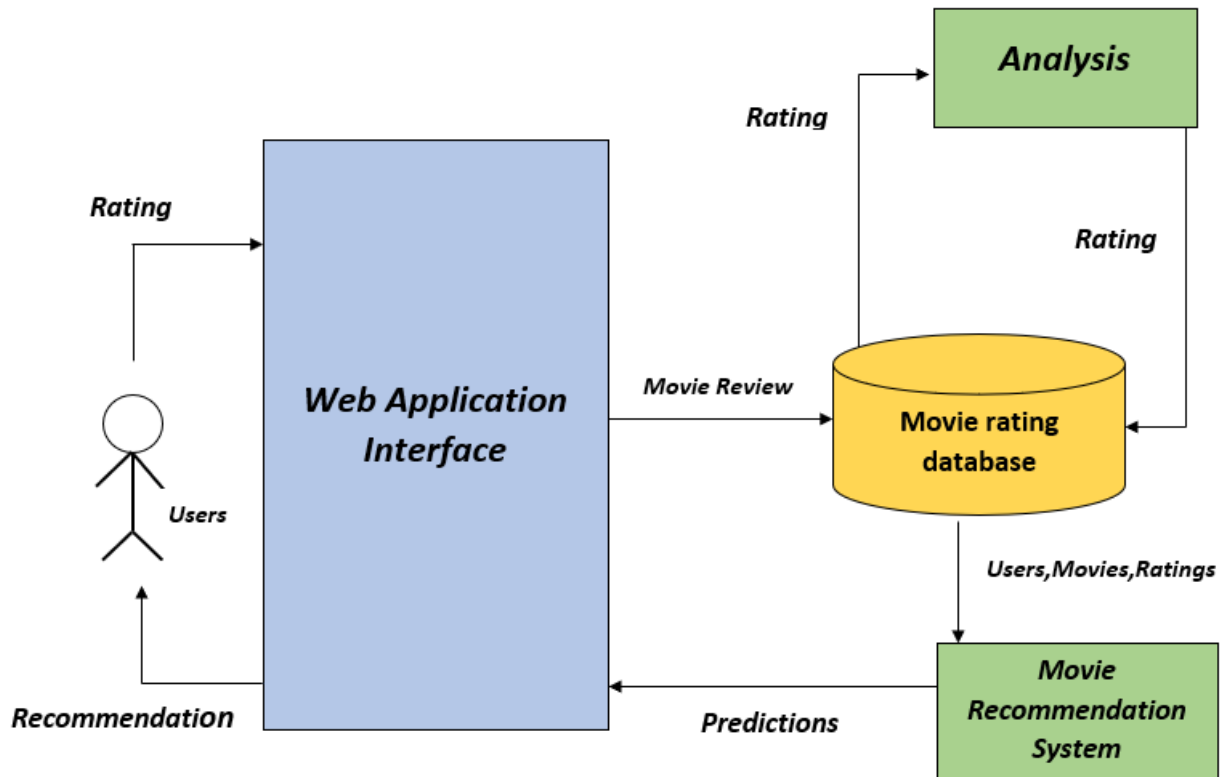


FIG 3.1 -MOVIE RECOMMENDATION SYSTEM ARCHITECTURE

3.2. SYSTEM DESIGN

OUR PROPOSED RECOMMENDATION SYSTEM

Deep artificial neural networks(Deep Learning) is able to learn complex decision boundaries for classification or complex non-linear regressions.

By stacking large numbers of hidden layers in these networks, deep neural networks can learn complex functions by learning to extract many low level features from the data and composing them in useful non-linear combinations.

3.2.1. NETWORK ARCHITECTURE:

The inputs to our network architecture **are two n-dimensional vectors**, where n is the number of movies in movie database.

One vector encodes a particular user profile, with each dimension indicating the rating the user gave for a particular film (or a zero to indicate that no rating has been given). The other vector is a one-hot encoding of a particular movie (i.e., a vector with a single “hot” dimension set to 1, with all other values set to zero).

These two vectors request that the network predict a rating for a particular user for a specific movie.

Hidden Layers. We start with a number of the standard fully-connected layers.

Output: Top Recommended movies for the user predicted by the System.

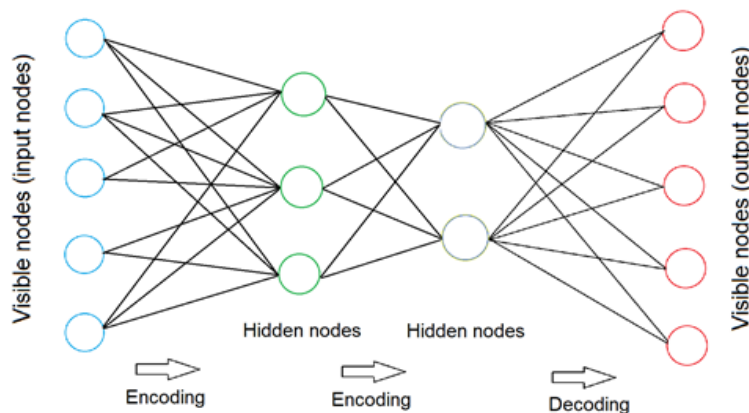


FIG 3.2- AUTOENCODERS ARCHITECTURE

3.2.2 DEEP LEARNING ARCHITECTURE:

We first consider the dimensions of the input and output of the neural network. The loss of the network on that training example must be computed with respect to the single withheld rating.

We choose to use **mean squared error (MSE)** with respect to known ratings as our loss function. Compared to the mean absolute error, mean squared error more heavily penalizes predictions which are further off. This is good in the context of recommender system because predicting a high rating for an item the user did not enjoy significantly impacts the quality of the recommendations. On the other hand, smaller errors in prediction likely result in recommendations that are still useful—the regression is not exactly correct, but at least the highest predicted rating are likely to be relevant to the user.

3.2.3 AUTOENCODERS:

One of the existing deep learning models is the Deep Neural Network (DNN) model. DNN is a Multi-Layer Perceptron (MLP) model with many hidden layers. The uniqueness of DNN is due to its larger number of hidden units and better parameter initialization techniques.

An autoencoder with three fully-connected hidden layers. Through an unsupervised learning algorithm, for linear reconstructions the autoencoder attempts to learn a function to minimize the mean square difference.

3.2.4 MULTI-LAYER PERCEPTRON:

This architecture takes an input and connects it to some number of fully connected hidden layers which include a “bottleneck.” This bottleneck is a hidden layer which has a much smaller dimensionality than the input. The output of the network is then re-expanded to have the same dimensionality as the input.

The network is then trained to learn the identity function, with the idea that in order for the network to compute the identity function through the bottleneck, it must learn a dense representation of the input.

3.2.5 INTITUTION

Many recommendation models have been proposed during the last few years. However, they all have their limitations in dealing with **data sparsity** and **cold-start issues**.

- The **data sparsity** occurs when the recommendation performance drops significantly if the interactions between users and items are very sparse.
- The **cold-start issues** occur when the model can't recommend new users and new items.

To solve these problems, recent approaches have exploited side information about users or items.

3.3 SOFTWARES USED

- Visual Studio Code:
- MongoDB compass:
- Spyder-IDE

Backend Technologies:

- Python Version (2,7,2.8)
- Anaconda (Spyder Ide)
- pytorch

Requirements:

- OPERATING SYSTEM: windows 10
- PROCESSOR INTEL: CORE i3
- DISK STORAGE :3Gb

3.4 SUMMARY:

MRS using AutoEncoders and hybrid model when compared with the standard collaborative techniques, the system outperforms the traditional methods and produces promising outcomes.

CHAPTER 4

IMPLEMENTATION DETAILS

4.1 DATASET

The data used here is MovieLens 100K. It has 100,000 movie ratings. Each user has an ID, and each movie has an ID.

This is split into 80,000 ratings for the training set `u_train.data` (with fields of user ID, movie ID, and that user's rating for that movie), and 20,000 for the testing set `u_test.data`.

This data set consists of:

- 100,000 ratings (1-5) from 943 users on 1682 movies.
- Each user has rated at least 20 movies.
- `movie.csv` that contains movie information(movieId,title,genres).
- `rating.csv` that contains ratings of movies by users:
(userId,movieId,Rating,timestamp)
- `User.csv` that contains user information(userId,Gender,Age,Occupation,ZipCode)

4.2 PREPROCESSING

The user and movie fields are non-sequential integers representing some unique ID for that entity. They need to be sequential starting at zero to use for modeling .

LabelEncoder().fit_transform(y)

We use scikit-learn's LabelEncoder class to transform the fields.

The `sklearn.preprocessing.LabelEncoder().fit_transform(y)` fits the label encoder, or assigns labels, for a single column and transforms the values in that column to the correct labels.

pandas.DataFrame.nunique()

It returns number of unique elements over the axis. The variables are created with the total number of unique users and movies in the data, also the min and max ratings present in the data.

```
19
20 user_enc = LabelEncoder()
21 ratings['user'] = user_enc.fit_transform(ratings['userId'].values)
22 n_users = ratings['user'].nunique()
23 item_enc = LabelEncoder()
24 ratings['movie'] = item_enc.fit_transform(ratings['movieId'].values)
25 n_movies = ratings['movie'].nunique()
26 ratings['rating'] = ratings['rating'].values.astype(np.float32)
27 min_rating = min(ratings['rating'])
28 max_rating = max(ratings['rating'])
```

FIG -4.1 PREPROCESSING

4.3 AUTOENCODERS

4.3.1 ALGORITHM:

STEP 1: We start with an array where the lines (the observations) correspond to the users and the columns (the features) correspond to the movies. Each cell (u, i) contains the rating (from 1 to 5, 0 if no rating) of the movie i by the user u .

STEP 2: The first user goes into the network. The input vector $x = (0, r_2, r_n)$ contains all its ratings for all the movies.

STEP 3: The input vector x is encoded into a vector z of lower dimensions by a mapping function f (sigmoid function): $z = f(Wx + b)$ where W is the vector of input weights and b the bias.

STEP 4: z is then decoded into the output vector y of same dimensions as x , aiming to replicate the input vector x .

STEP 5: The reconstruction error $d(x, y) = \|x - y\|$ is computed. The goal is to minimize it.

STEP 6: Back-Propagation: from right to left, the error is back-propagated. The weights are updated according to how much they are responsible for the error. The learning rate decides by how much we update the weights.

STEP 7: Repeat Steps 1 to 6 and update the weights after each observation (Reinforcement Learning). Or: Repeat Steps 1 to 6 but update the weights only after a batch of observations (Batch Learning).

STEP 8: When the whole training set passed through the ANN, that makes an epoch. Redo more epochs.

4.3.2 Stacked AutoEncoders(SAE) Implementation:

Data structure creation

We convert Dataframe to Numpy array because we will use Pytorch tensor which requires array as input.

```
"""## Convert data into Torch tensors"""  
  
training_set = torch.FloatTensor(training_set)  
test_set = torch.FloatTensor(test_set)
```

FIG 4.2 PYTORCH

To prepare the training and test data, we need to create training and test sets in array format with each row representing a user and each cell in the row representing the rating for each movie. We convert the training set and test set to numpy arrays. The total number of users as row numbers and the total number of movies as column numbers is obtained as below.

```
training_set = pd.read_csv('ml-100k/u1.base', delimiter = '\t')  
training_set = np.array(training_set, dtype = 'int')  
test_set = pd.read_csv('ml-100k/u1.test', delimiter = '\t')  
test_set = np.array(test_set, dtype = 'int')  
  
"""## number of users and movies"""  
  
nb_users = int(max(max(training_set[:, 0], ), max(test_set[:, 0])))  
nb_movies = int(max(max(training_set[:, 1], ), max(test_set[:, 1])))
```

FIG- 4.3 TRAIN AND TEST SET

Data Conversion

We create a function for data conversion which returns a list of lists. Each child list represents one user's ratings for all movies. If the user did not rate a movie, initialize the rating with 0. With the above conversion function, we convert the training set and test set.

```
def convert(data):
    new_data = []
    for id_users in range(1, nb_users + 1):
        id_movies = data[:, 1] [data[:, 0] == id_users]
        id_ratings = data[:, 2] [data[:, 0] == id_users]
        ratings = np.zeros(nb_movies)
        ratings[id_movies - 1] = id_ratings
        new_data.append(list(ratings))
    return new_data
training_set = convert(training_set)
test_set = convert(test_set)
```

FIG -4.4 NUMPY TO TENSOR

We convert the **list of list** type into **Tensor** because we use Pytorch to build the auto-encoder.

Model Creation:

We will inherit a parent class called ***Module*** from Pytorch. Inheritance allows to create a child class to build SAE.

ENCODING :

Inherit all classes and functions from the parent class *Module*. Create the **1st full connection layer *fc1*** using inherited class *nn.Linear()*, which connects between first input vector features and the first encoded vector.

The first argument for *nn.Linear()* is the number of features, which is the number of movies, *nb_movies*. The 2nd argument is the number of nodes in the first hidden layer. We choose 20, the first encoded vector is a vector of 20 elements. These 20 features represent features of movies that are liked by similar users. The 2nd hidden layer *fc2* with 20 features as input and 10 encoded features as output.

DECODER:

The decoding layers *fc3* and *fc4* which will be symmetrical to the encoding. Remember, auto-encoder aims to reconstruct the input vector, and the output vector needs to have the same dimension as the input vector.

The activation function is specified for the model, **Sigmoid function** .

```
class SAE(nn.Module):
    def __init__(self, ):
        super(SAE, self).__init__()
        self.fc1 = nn.Linear(nb_movies, 20)
        self.fc2 = nn.Linear(20, 10)
        self.fc3 = nn.Linear(10, 20)
        self.fc4 = nn.Linear(20, nb_movies)
        self.activation = nn.Sigmoid()
```

FIG -4.5 ACTIVATION FUNCTION

Forward Function

Activation functions is applied for encoding and decoding layers. It returns a vector of the predicted ratings which will be compared to the real ratings. The input vector x argument, will be encoded twice and decoded twice successively, producing the reconstructed vector. The first activation function **activates the 20 neurons of the first hidden layer $fc1$ which accepts the input vector x . Similarly, $fc2$ and $fc3$ are activated.** We do not apply activation function in $fc4$ when reconstructing the decoded vector at the output layer. The output is vector of predicted ratings.

```
def forward(self, x):  
    x = self.activation(self.fc1(x))  
    x = self.activation(self.fc2(x))  
    x = self.activation(self.fc3(x))  
    x = self.fc4(x)  
    return x
```

FIG-4.6 ACTIVATION FORWARD FUNCTION

4.3.3 Training SAE:

We use the **Mean Squared Error** from `nn.MSELoss()` for loss function, RMSprop from `optim.RMSprop()` for the optimizer. We choose a **learning rate of 0.01**, and **weight decay of 0.5**. (weight decay is used to reduce learning rate after every few epochs, to regulate the convergence). We train the model on **200 epochs**.

```
sae = SAE()  
criterion = nn.MSELoss()  
optimizer = optim.RMSprop(sae.parameters(), lr = 0.01, weight_decay = 0.5)
```

FIG-4.7 LOSS FUNCTION

We create 2 for loops, one for epoch iteration, and one for observation iteration. Inside the epoch loop, initialize the *train_loss* and number of users who rated at least 1 movie. To optimize the computation, the model will not be trained on users who did not rate any movies.

Each observation is fed one by one to predict the ratings for each user using the SAE class with *sae* object in each iteration. The loss of each observation is calculated, and the optimizer is applied to optimize the loss. The loss is accumulated after each epoch.

A batch dimension is added for the input vector from the training set for PyTorch network. We use *Variable()* and *unsqueeze()* functions to put the batch dimension at index 0. Clone the input to create the target using *clone()* function.

```
"""## Training the SAE"""
nb_epoch = 200
for epoch in range(1, nb_epoch + 1):
    train_loss = 0
    s = 0.
    for id_user in range(nb_users):
        input = Variable(training_set[id_user]).unsqueeze(0)
        target = input.clone()
        if torch.sum(target.data > 0) > 0:
            output = sae(input)
            target.requires_grad = False
            output[target == 0] = 0
            loss = criterion(output, target)
            mean_corrector = nb_movies/float(torch.sum(target.data > 0) + 1e-10)
            loss.backward()
            train_loss += np.sqrt(loss.data*mean_corrector)
            s += 1.
        optimizer.step()
    train_loss=train_loss/s
    print('epoch: ' +str(epoch)+'loss: ' + str(train_loss/s))
```

FIG-4.8 TRAINING SAE

We set *target.require_grad False*, since *target* copied the same *require_grad* field from *input*. We only need to compute the gradient with respect to *input*, not to *target*. We can efficiently compute the loss using **criteron object**. *backward()* method is used to decide which direction to update the weights, increasing or decreasing. We use *optimizer.step()* to denote number of weights to be updated .

4.3.4 Testing the SAE

We use the *training_set as input* to make a prediction and use *test_set as target* for loss computation. We only compute the test loss for movies that are rated by the users in *test_set*.

```
"""## Testing the SAE"""
test_loss = 0
s = 0.
for id_user in range(nb_users):
    input = Variable(training_set[id_user]).unsqueeze(0)
    target = Variable(test_set[id_user]).unsqueeze(0)
    if torch.sum(target.data > 0) > 0:
        output = sae(input)
        target.require_grad = False
        output[target == 0] = 0
        loss = criterion(output, target)
        mean_corrector = nb_movies/float(torch.sum(target.data > 0) + 1e-10)
        test_loss += np.sqrt(loss.data*mean_corrector)
        s += 1.
test_loss=test_loss/s
print('test Loss: '+str(test_loss/s))
```

FIG-4.9 TESTING SAE

4.4 SVD MODEL:

The Singular Value Decomposition (SVD) is a popular method in linear algebra for matrix factorization in machine learning for dimensionality reduction. It shrinks the space dimension from N-dimension to K-dimension (where $K < N$) and reduces the number of features. SVD constructs a matrix with the row of users and columns of items and the elements are given by the users' ratings. Singular value decomposition decomposes a matrix into three other matrices and extracts the factors from the factorization of a high-level (user-item-rating) matrix.

$$A = USV^T$$

Matrix U: singular matrix of (user*latent factors)

Matrix S: diagonal matrix (shows the strength of each latent factor)

Matrix U: singular matrix of (item*latent factors)

To train recommender systems with Surprise, we need to create a Dataset object. A Surprise Dataset object is a dataset that contains the following fields in this order:

1. The user IDs
2. The movie IDs
3. The corresponding rating

We call the pivot function to create a pivot table where users take on different rows, Movies different columns, and respective ratings values within that table with a shape (m*n).

The `cross_validate()` function runs a cross-validation procedure according to the `cv` argument, and computes accuracy measures. We are using a classical 5-fold cross-validation.

`fit()` method which will train the algorithm on the trainset, and the `test()` method which will return the predictions made from the testset

The SVD can be calculated by calling the `svd()` function. The function takes a matrix and returns the `U`, `Sigma` and `VT` elements. The `Sigma` diagonal matrix is returned as a vector of singular values. The `V` matrix is returned in a transposed form, The original matrix can be reconstructed from the `U`, `Sigma`, and `VT` elements. The `U`, `s`, and `V` elements returned from the `svd()` cannot be multiplied directly.

The `s` vector must be converted into a diagonal matrix using the `diag()` function. By default, this function will create a square matrix that is $n \times n$, relative to our original matrix. This causes a problem as the size of the matrices do not fit the rules of matrix multiplication, where the number of columns in a matrix must match the number of rows in the subsequent matrix.

A user implicitly tells us about her preferences by choosing to voice her opinion and vote a rating. This reduces the ratings matrix into a binary matrix, where “1” stands for “rated”, and “0” for “not rated” This binary data is not as vast and independent as other sources of implicit feedback We have found that incorporating this kind of implicit data which inherently exist in every rating based recommender system significantly improves prediction accuracy. SVD++ factors in this implicit feedback and gives better accuracy .

4.5 CONVOLUTIONAL NEURAL NETWORKS

The user/movie fields are currently non-sequential integers representing Unique ID for that entity. They need to be sequential starting at zero to use for modeling. We use scikit-learn's LabelEncoder class to transform the fields.

We'll create variables with the total number of unique users and movies in the data, as well as the min and max ratings present in the data.

We turn users and movies into separate arrays in the training and test data. This is because in Keras they'll each be defined as distinct inputs, and the way Keras works is each input needs to be fed in as its own array.

We're going to use **embeddings** (from `keras.layers.Embeddings`) to represent each user and each movie in the data. These embeddings will be vectors (of size `n_factors`). We compute the dot product between a user vector and a movie vector to get a predicted rating. In an embedding model the embeddings are the weights that are learned during training.

1. **Input:** Input for both movies and users
2. **Embedding Layers:** Embeddings for movies and users
3. **Dot:** combines embeddings using a dot product

Training the model

We have two input layers (one for the Moovies and one for the users), we need to specify an array of training data as our x data. This model is for 10 epochs.

4.6 HYBRID MODEL

We stack the individual models **Content based Filtering** model and Single value decomposition model discussed above.

Hybrid =Content Based Filtering + SVD

ALGORITHM:

1. Run Content based filtering and determine the movies which we want to recommend to the user.
2. Filter and sort the recommendations of CF using SVD predicted ratings.

4.7 FRONT -END FRAMEWORK:

4.7.1 REGISTRATION ALGORITHM:

REGISTRATION USING NODE.JS AND MONGO DB

STEP 1:The User Interface contains a Navbar with fields "Login" and "Register".

STEP 2.If you are already a registered user, you can login else you have register first inorder to login.

STEP 3.Clicking on "Register" tab in the navigation bar will redirect you to "Register.ejs" page

STEP 4:Register page contains fields like

"UserID","Gender","Age","Occupation","ZipCode","Password"

STEP 5:After entering all the values with Unique UserID your details will be successfully registered in the "users" collections in "MovieRec" Database of MongoDB.

4.7.2 LOGIN ALGORITHM:

STEP 1:The login page "Login.ejs" contains fields "UserID" and "Password".

STEP 2:By clicking on login after entering the user credentials, it will check the database to ensure whether he is a legitimate user or not, if yes,it will redirect the user to the Movie Recommendation Home page where the actual recommendation is performed.

STEP 3:If the user data is not present in the database, the user needs to Re-enter his credentials .

4.7.3 RENDERING RECOMMENDED MOVIES-ALGORITHM:

STEP 1:Once the user logs in, the respective user data is fetched from the Mongoddb database and it is sent to the machine learning model which performs all the necessary operations required to predict the movies, the user may like to watch.

STEP 2:If he/she is a naive user, the model will recommend the user with highly rated movies.

STEP 3:Node js child process is used to send the data from node js to python script.The model will perform collaborative filtering and send all the predicted movies along with respective data in the form of json to the Node js child process where the python script is actually called.

STEP 4:All the necessary preprocessing is done in order to display the predicted movies in the client side.

STEP 5:After that, the processed data is sent to home page which is written in ejs. Once the data is received by the home.ejs page, it will render it to the client side with drop down list box to rate movies.

4.7.4 USER RATINGS ALGORITHM:

STEP 1:When the User rates a particular movie, the correspong record is fetched from the Mongoddb database using "Ratings.find()" method.

STEP 2:After fetching the record, the rating of that particular movie will be updated as rated by the user using "Ratings.updateOne()" function.

CHAPTER 5

EVALUATION AND RESULTS

5.1 PERFORMANCE ANALYSIS:

5.1.1 AUTOENCODERS:

In the model training, we got a loss of 1.77 at epoch 1, a loss of 0.934 at epoch 100, and a loss of 0.914 at epoch 200. In the model training, we got a loss of 0.95 . The optimal result obtained is using a 3 layers deep neural network and 200 epochs.

EPOCH	MSE LOSS
1-50	1.0345
51-100	0.9453
101-150	0.9301
151-200	0.9123
Test Loss	0.9681

TABLE -5.1 SAE TRAINING LOSS COMPARISON



FIG -5.2 EPOCH VS TRAINING LOSS PLOT

```

epoch: 1loss: tensor(1.7723)
epoch: 2loss: tensor(1.0968)
epoch: 3loss: tensor(1.0532)
epoch: 4loss: tensor(1.0384)
epoch: 5loss: tensor(1.0310)
epoch: 6loss: tensor(1.0266)
epoch: 7loss: tensor(1.0240)
epoch: 8loss: tensor(1.0217)
epoch: 9loss: tensor(1.0208)
epoch: 10loss: tensor(1.0196)
epoch: 11loss: tensor(1.0191)
epoch: 12loss: tensor(1.0184)
epoch: 13loss: tensor(1.0181)
epoch: 14loss: tensor(1.0175)
epoch: 15loss: tensor(1.0172)
epoch: 16loss: tensor(1.0169)
epoch: 17loss: tensor(1.0167)
epoch: 18loss: tensor(1.0166)
epoch: 19loss: tensor(1.0164)
epoch: 183loss: tensor(0.9180)
epoch: 184loss: tensor(0.9176)
epoch: 185loss: tensor(0.9175)
epoch: 186loss: tensor(0.9173)
epoch: 187loss: tensor(0.9172)
epoch: 188loss: tensor(0.9172)
epoch: 189loss: tensor(0.9169)
epoch: 190loss: tensor(0.9169)
epoch: 191loss: tensor(0.9163)
epoch: 192loss: tensor(0.9164)
epoch: 193loss: tensor(0.9159)
epoch: 194loss: tensor(0.9190)
epoch: 195loss: tensor(0.9166)
epoch: 196loss: tensor(0.9155)
epoch: 197loss: tensor(0.9153)
epoch: 198loss: tensor(0.9152)
epoch: 199loss: tensor(0.9146)
epoch: 200loss: tensor(0.9145)

```

FIG-5.3 AUTOENCODER EPOCH OUTPUT

5.1.2 SVD MODEL

The Singular Value Decomposition used for recommendation system gave a accuracy of 94.5 % and RMSE 0.9406. We obtain the following result by using classical 5-fold cross-validation. SVD++ factors in this implicit feedback and gives significantly produces good accuracy. We used n_latent factors here as 160.

	FOLD 1	FOLD 2	FOLD 3	FOLD 4	FOLD 5	MEAN	STD
RMSE(TestSet)	0.8733	0.8761	0.8745	0.8734	0.8738	0.8742	0.0010
MAE(TestSet)	0.6858	0.6874	0.6861	0.6857	0.6867	0.6863	0.0006
Fit time	84.71	92.67	85.18	86.43	83.75	86.55	3.18
Test time	2.74	54.68	3.48	3.27	3.38	13.51	20.59

TABLE -5.4 K -CROSS FOLD VALIDATION

```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).
```

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.8733	0.8761	0.8745	0.8734	0.8738	0.8742	0.0010
MAE (testset)	0.6858	0.6874	0.6861	0.6857	0.6867	0.6863	0.0006
Fit time	84.71	92.67	85.18	86.43	83.75	86.55	3.18
Test time	2.74	54.68	3.48	3.27	3.38	13.51	20.59

```
SVDpp : Test Set  
RMSE: 0.9406  
Out[3]: 0.9405979804106971
```

FIG -5.5 K-CROSS FOLD OUTPUT

5.1.3 CONVOLUTIONAL NEURAL NETWORKS

The Convolutional Neural Network used for recommendation system gave a validation The number of layers as well as the number of nodes in the intermediate layers was varied to determine the model with the highest accuracy. The optimized results was obtained for a CNN with three layers consisting of 50 respectively and with two dense layers and latent factors as 50 for embedding .The Accuracy of this model was less compared to other two models.

EPOCH(1418/1418)	STEP LOSS	VAL LOSS
1	0.6557	0.6203
2	0.613	0.6185
3	0.6105	0.6134
4	0.6074	0.61345
5	0.6089	0.6199

TABLE -5.6 CNN LOSS TABLE

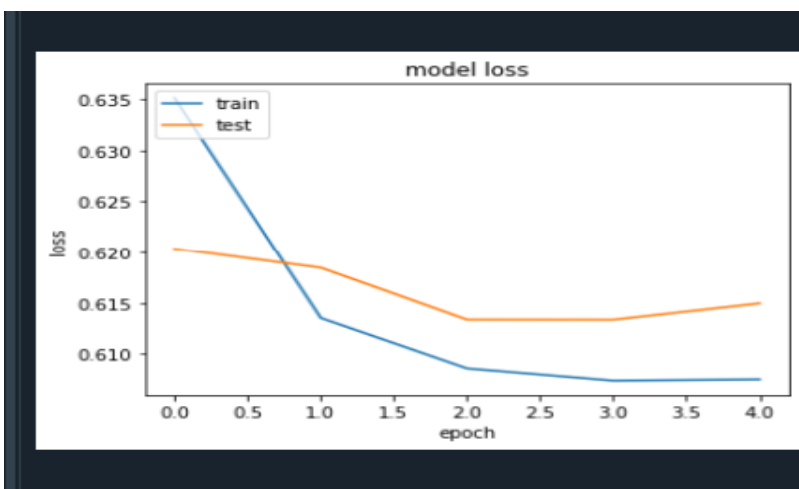


FIG -5.7 CNN LOSS PLOT

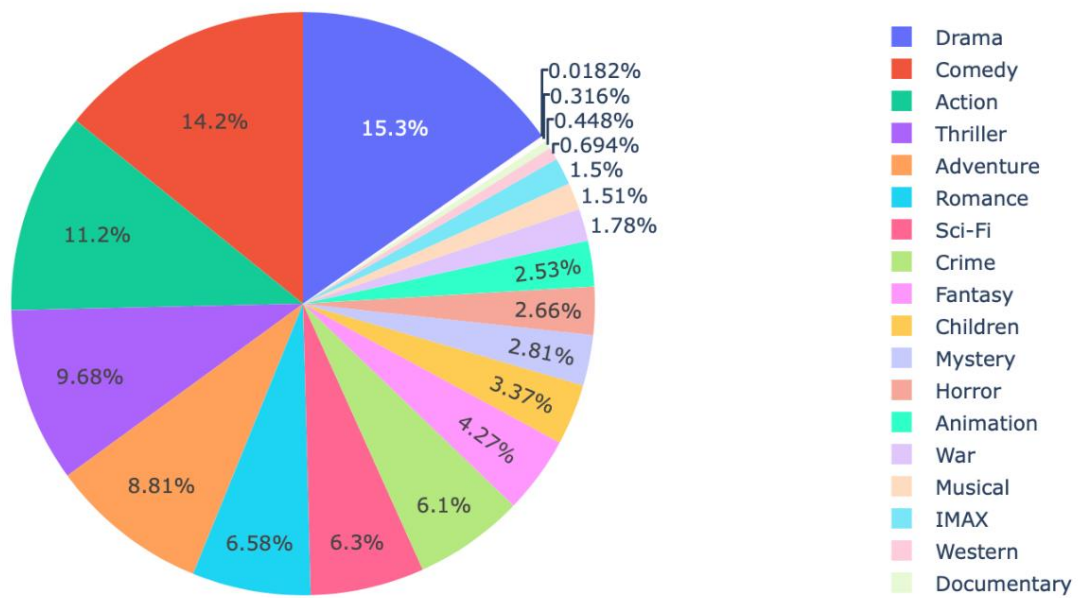


FIG -5.8 GENRE DISTRIBUTION



FIG -5.9 LOSS COMPARISON

5.2 NOVELTY OF THE SYSTEM:

The content based model approach don't suffer from Cold Start problem ,we only need basic information on a user (a single movie) to provide similar recommendations based on the items.We provide recommendations based on popular genres and also the genres based on the movie user has watched.

Drawback is that it tends to **recommend the same type of items to the user**. In order to be able to recommend a different type of item, the user would have to have rated or have shown interest in the new type of item. This is a problem that Collaborative Filtering methods don't have, since the match here is done between neighbouring users with similar tastes, but different items rated.

We have used the **HYBRID MODEL** which stacks the individual model (**Content based filtering model and SVD model**)which outperforms the individual models.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

Recommender systems are a powerful new technology for extracting additional value for a business from its user databases. These systems help users find items they want to buy from a business. Recommender systems benefit users by enabling them to find items they like. Conversely, they help the business by generating more sales. Recommender systems are rapidly becoming a crucial tool in E-commerce on the Web. Recommender systems are being stressed by the huge volume of user data in existing corporate databases, and will be stressed even more by the increasing volume of user data available on the Web. We have made a Neural network model which has least root mean squared error. This model gives best prediction ratings of the users which had not given ratings to the movies. We have used regularization function to minimize the errors. Also, our system gave best movie recommendations. Our approach is better at test time with net test loss=0.96.

6.2 FUTURE WORK:

The offered technique is an incentive to recommend films to users using autoencoders. There are a lot of approach to develop the work done in this task. The content based strategy can be extended to incorporate more criteria to help sort the movies.. Furthermore, movies discharged inside a similar timeframe could likewise get a lift in probability for suggestion. We could create hybrid strategies that attempt to join the upsides of both content based techniques and collaborative filtering into one recommendation framework. We are keen in exploring denoising and variational Autoencoders for Recommendation System.

REFERENCES

- [1] *Recommender system using fuzzy(2015)* by Hirdesh Shivhare et. Al ,2015 IEEE International Conference on Computer (Communication and Control (IC4-2015))
- [2] *An Improved Collaborative MRS using computational intelligence* by Zan Wang et. al.(2014)
- [3] *Reviewing cluster based collaborative filtering approach* by F.Darvishi-mirshekarlou et. al.(2013)
- [4] *Maximum entropy approach* by D. Y. Pavlov and D. M. in Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, Vancouver, British Columbia, Canada]
- [5] *Hybrid personalized Recommender* by Subhash K. Shindeet Shinde Research Scholar, SRTMU, Nanded, India Uday V. Kulkarni Professor, Computer Engineering, SGGSIET, Nanded, India Published in International Journal of Artificial Intelligence and Expert Systems (IJAE)
- [6] *Recommendation system using fuzzy C-means Clustering* by Kwoting Fanget.al.
- [7] K. Kazuya and M. Yutaka, “*The application of deep learning in recommender system,*” Proceedings of the 28th Annual Conference of the Japanese Society for Artificial Intelligence, vol. 28, pp. 1–4, 2014. (in Japanese)
- [8] *Cold Start, Warm Start and Everything in Between: An Autoencoder based Approach to Recommendation* by Anant Jain and Angshul Majumdar, IIIT Delhi, New Delhi, India
- [9] *Expanded Autoencoder Recommendation Framework and its Application in Movie Recommendation* by Baolin Yi, Xiaoxuan Shen, Zhaoli Zhang and Jiangbo Shu, Hai Liu, Member, IEEE, national Engineering Research Center for e-Learning, Central China Normal University, Wuhan Hubei, China

APPENDIX-1

AUTOENCODERS

Auto-encoder is a type of directed neural network, which aims to generate outputs as identical as inputs. An Autoencoder is an ANN used to learn a representation (encoding) for a set of input data, usually to achieve dimensionality reduction. Inputs are encoded and decoded through hidden layers, producing outputs which are then compared to the inputs. Afterward, back-propagation is performed to update weights. This iteration is repeated for training purposes

Autoencoder is a feedforward neural network having an input layer, one hidden layer and an output layer. The output layer has the same number of neurons as the input layer for the purpose of reconstructing its own inputs. This makes an Autoencoder a form of unsupervised learning, no labelled data are necessary — only a set of input data instead of input-output pairs

PYTORCH TENSOR

PyTorch supports dynamic computation graphs that allow you to change how the network behaves on the fly, unlike static graphs that are used in frameworks such as Tensorflow. Numpy, with strong GPU acceleration.

PyTorch is a Python-based scientific computing package that uses the power of graphics processing units. It is also one of the preferred deep learning research platforms built to provide maximum flexibility and speed. It is known for providing

two of the most high-level features; namely, tensor computations with strong GPU acceleration support and building deep neural networks.

FRONT-END FRAME WORKS AND DATABASE:

Node.js

Node.js is a free, open-sourced, cross-platform JavaScript run-time environment that lets developers write command line tools and server-side scripts outside of a browser.

Express.js:

Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node based Web applications. Following are some of the core features of Express framework –

- Allows to set up middlewares to respond to HTTP Requests.
- Defines a routing table which is used to perform different actions based on HTTP Method and URL.
- Allows to dynamically render HTML Pages based on passing arguments to templates.

MongoDB:

MongoDB is a NoSQL database which stores the data in form of key-value pairs. It is an **Open Source, Document Database** which provides high performance and scalability along with data modelling and data management of huge sets of data in an enterprise application.

MongoDB also provides the feature of Auto-Scaling. Since, MongoDB is a cross platform database and can be installed across different platforms like Windows, Linux etc.

Visual studio code is a very powerful and easy-to-use **code** editor. It comes with broad programming language support, is highly customizable with various extensions and it is for free - a great package for beginners and more advanced programmers.

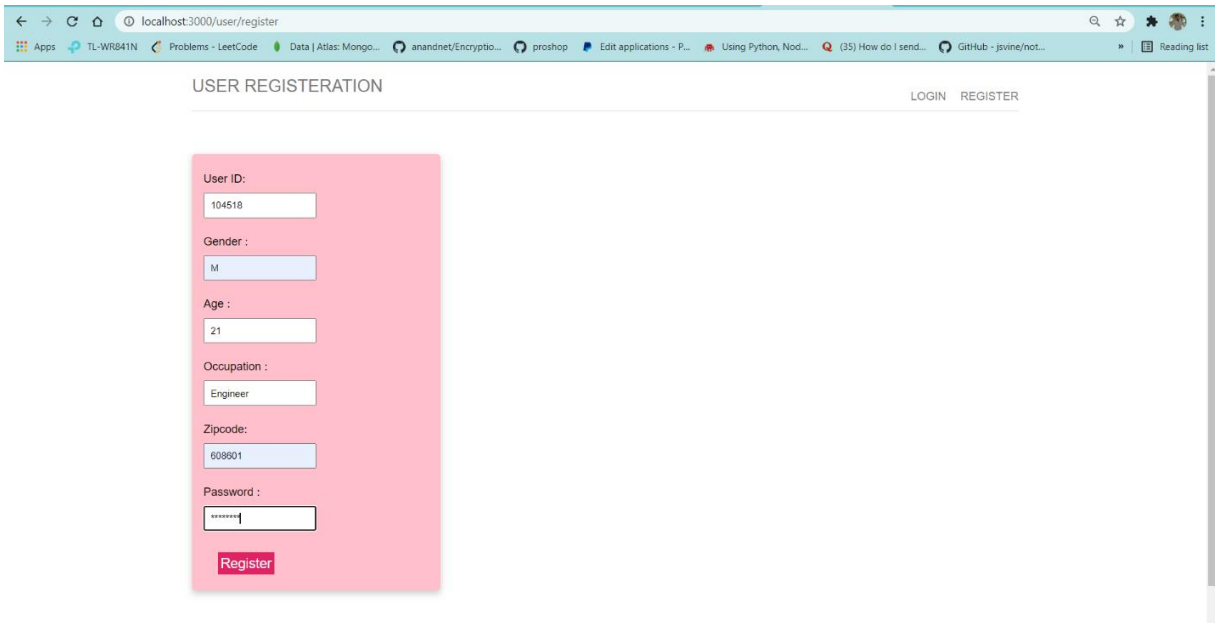
MongoDB compass allows you to analyse and understand the contents of your data without formal knowledge of ***MongoDB*** query syntax

Spyder IDE- **Spyder** is an open-source cross-platform integrated development environment (IDE) for scientific programming in the Python language. Spyder integrates with a number of prominent packages in the scientific Python including NumPy, SciPy, Matplotlib, pandas, IPython, SymPy and Cython, as well as other open-source software.

APPENDIX – II

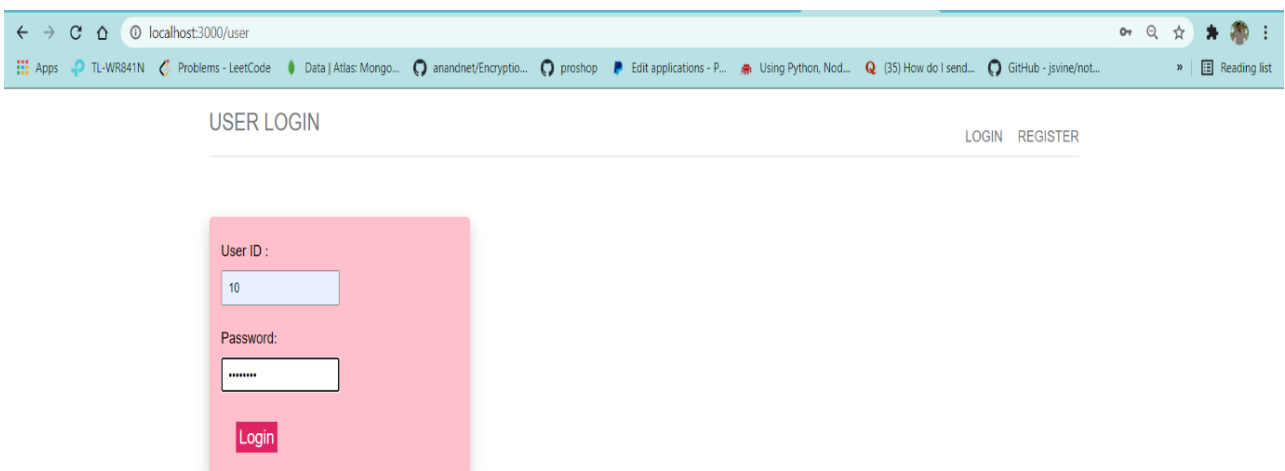
SCREENSHOTS

FRONT-END FRAMEWORK



The screenshot shows a web browser at localhost:3000/user/register. The page has a header with 'USER REGISTRATION' and links for 'LOGIN' and 'REGISTER'. A pink registration form is centered, containing fields for 'User ID' (104518), 'Gender' (M), 'Age' (21), 'Occupation' (Engineer), 'Zipcode' (608601), and 'Password' (masked with asterisks). A red 'Register' button is at the bottom of the form.

FIG -1A REGISTRATION PAGE (USING NODE.JS)



The screenshot shows a web browser at localhost:3000/user. The page has a header with 'USER LOGIN' and links for 'LOGIN' and 'REGISTER'. A pink login form is centered, containing fields for 'User ID' (10) and 'Password' (masked with asterisks). A red 'Login' button is at the bottom of the form.

Copyright ©2021

FIG -2A LOGIN PAGE

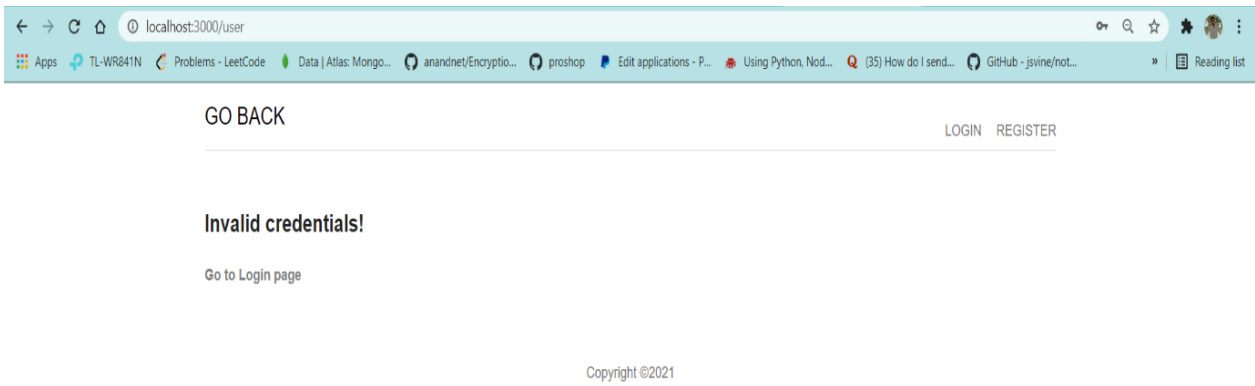


FIG -3A CREDENTIALS ERROR OUTPUT

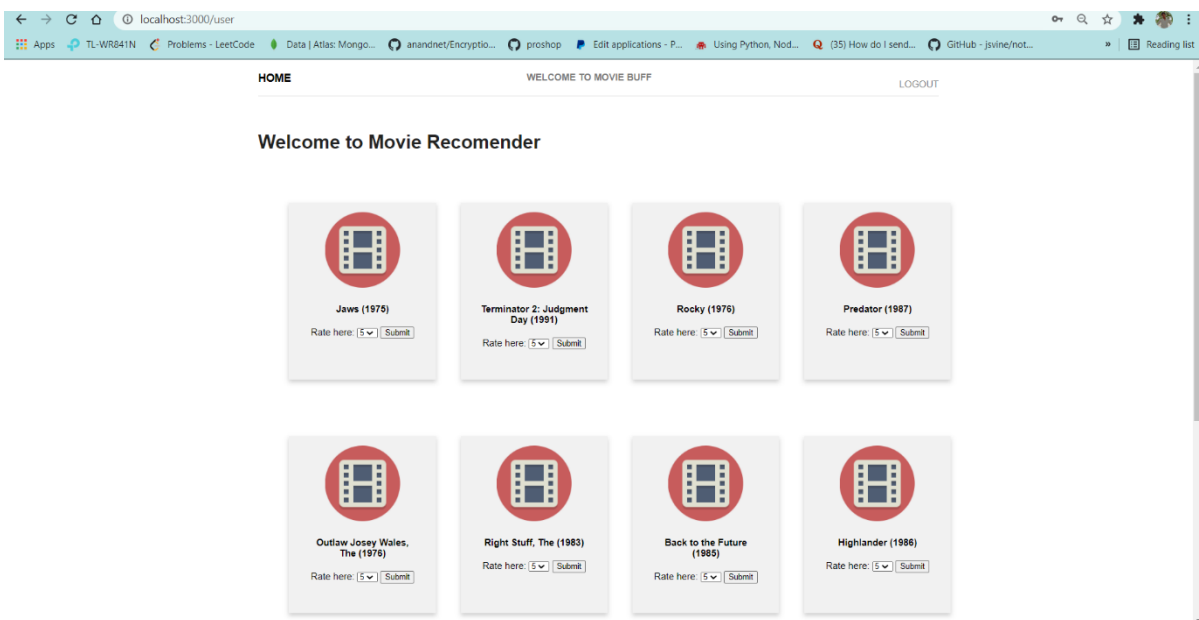


FIG-4A MOVIE RECOMMENDATION SYSTEM OUTPUT

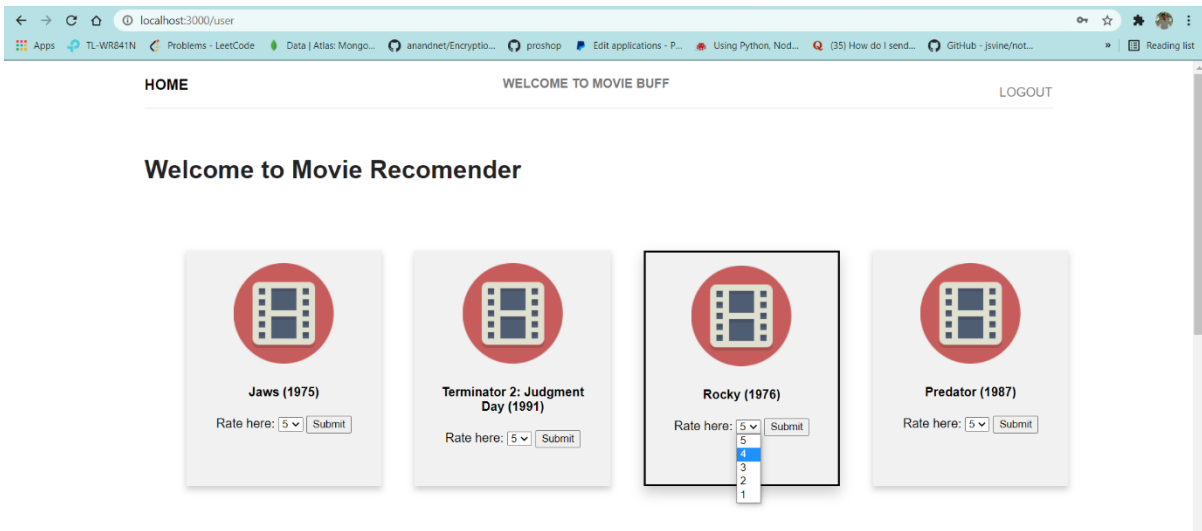


FIG -5A RATE MOVIE OUTPUT

```

PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
]
POST /user 200 35985.674 ms - 10543
PS C:\Users\Gopi\Desktop\movierec project\nodelogin> nodemon app
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
GET /user 200 572.664 ms - 1440
Rendering homepage.....
10
POST /user 200 40930.735 ms - 10462
1885
3
User Rating Updated Successfully!
POST /user/1885/10 - - ms - -
585
4
User Rating Updated Successfully!
POST /user/585/10 - - ms - -
[]

```

FIG-6A TERMINAL OUTPUT

MongoDB Compass - localhost:27017/MovieRec

Connect View Help

Local

DBS COLLECTIONS

HOST: localhost:27017

CLUSTER: Standalone

EDITION: MongoDB 4.4.4 Community

Filter your data

MovielRec

collections

ratings

users

admin

config

local

CREATE COLLECTION

Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
movies	9,000	92.1 B	809.4 KB	1	140.0 KB	
ratings	5,001,057	60.8 B	289.8 MB	1	48.2 MB	
users	6,040	97.3 B	574.2 KB	1	80.0 KB	

FIG -8A MONGODB COLLECTION

BACK END OUTPUT:

```
In [24]: result_df.head()
Out[24]:
```

	Movie	User input	Target Rating
0	GoldenEye (1995)	3	3.92619
1	Dracula: Dead and Loving It (1995)	5	4.51335
2	Nixon (1995)	5	3.88791
3	Sense and Sensibility (1995)	3	3.46745
4	Money Train (1995)	4	3.43806

```
In [25]: result_df[result_df['Target Rating'] > 0]
...: prediction = sae(test_set)

In [26]: prediction
Out[26]:
```

```
tensor([[4.0769, 3.7017, 3.3186, ..., 2.1564, 3.5031, 3.2625],
        [4.2193, 3.8831, 3.5522, ..., 2.2466, 3.6463, 3.3997],
        [4.7945, 4.6448, 4.5761, ..., 2.6271, 4.2475, 3.9781],
        ...,
        [4.1017, 3.7727, 3.4814, ..., 2.1943, 3.5599, 3.3207],
        [4.1017, 3.7727, 3.4814, ..., 2.1943, 3.5599, 3.3207],
        [4.1017, 3.7727, 3.4814, ..., 2.1943, 3.5599, 3.3207]],
        grad_fn=<AddmmBackward>)
```

FIG-9A AUTOENCODERS OUTPUT:

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
User_Input (InputLayer)	[(None, 1)]	0	
input_1 (InputLayer)	[(None, 1)]	0	
embedding (Embedding)	(None, 1, 50)	30500	User_Input[0][0]
embedding_1 (Embedding)	(None, 1, 50)	486200	input_1[0][0]
reshape (Reshape)	(None, 50)	0	embedding[0][0]
reshape_1 (Reshape)	(None, 50)	0	embedding_1[0][0]
User_Vector (Flatten)	(None, 50)	0	reshape[0][0]
Movie_Vector (Flatten)	(None, 50)	0	reshape_1[0][0]
concatenate (Concatenate)	(None, 100)	0	User_Vector[0][0] Movie_Vector[0][0]
dropout (Dropout)	(None, 100)	0	concatenate[0][0]
dense (Dense)	(None, 100)	10100	dropout[0][0]
activation (Activation)	(None, 100)	0	dense[0][0]
dropout_1 (Dropout)	(None, 100)	0	activation[0][0]

reshape_1 (Reshape)	(None, 50)	0	embedding_1[0][0]
User_Vector (Flatten)	(None, 50)	0	reshape[0][0]
Movie_Vector (Flatten)	(None, 50)	0	reshape_1[0][0]
concatenate (Concatenate)	(None, 100)	0	User_Vector[0][0] Movie_Vector[0][0]
dropout (Dropout)	(None, 100)	0	concatenate[0][0]
dense (Dense)	(None, 100)	10100	dropout[0][0]
activation (Activation)	(None, 100)	0	dense[0][0]
dropout_1 (Dropout)	(None, 100)	0	activation[0][0]
dense_1 (Dense)	(None, 1)	101	dropout_1[0][0]
activation_1 (Activation)	(None, 1)	0	dense_1[0][0]
lambda (Lambda)	(None, 1)	0	activation_1[0][0]
=====			
Total params: 526,901			
Trainable params: 526,901			
Non-trainable params: 0			

FIG-10 CNN MODEL SUMMARY

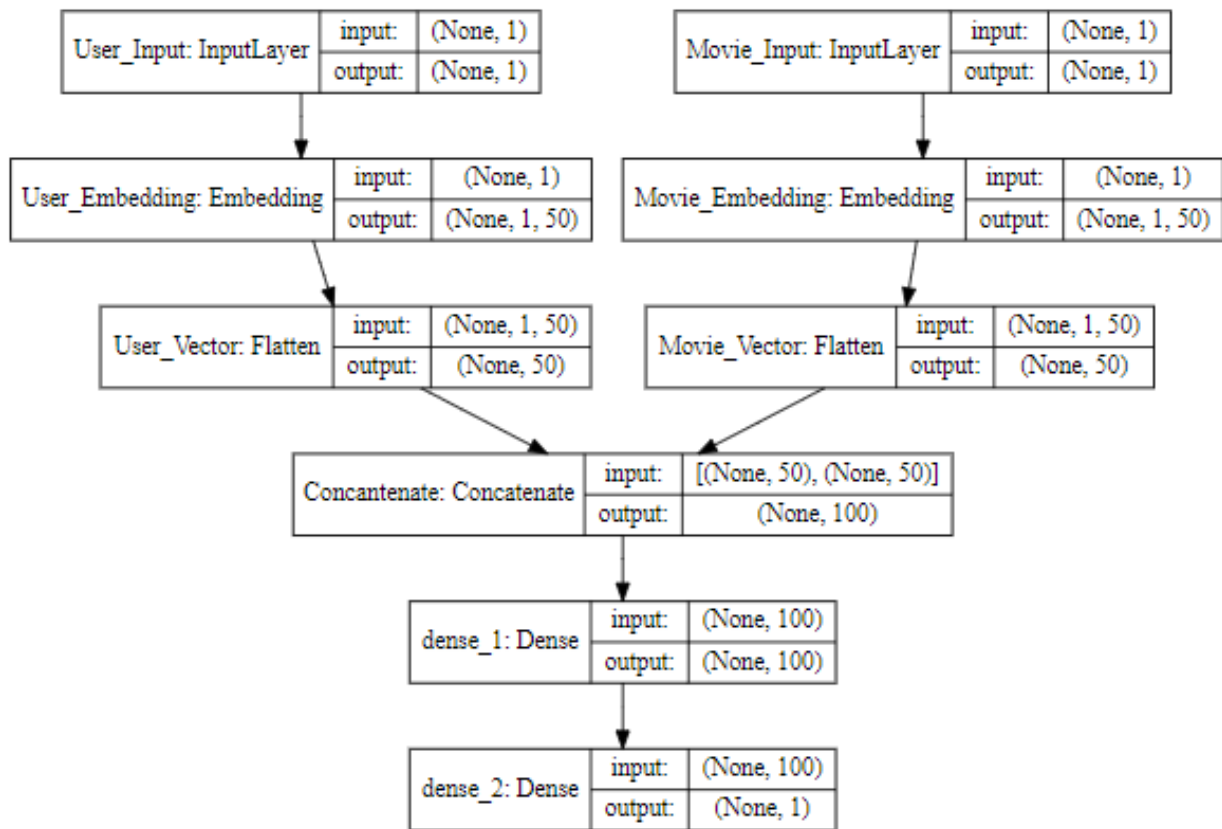


FIG 11 A-CNN MODEL DIAGRAM

```

Epoch 1/5
1418/1418 [=====] - 48s 28ms/step - loss:
0.6557 - val_loss: 0.6203
Epoch 2/5
1418/1418 [=====] - 17s 12ms/step - loss:
0.6163 - val_loss: 0.6185
Epoch 3/5
1418/1418 [=====] - 17s 12ms/step - loss:
0.6105 - val_loss: 0.6134
Epoch 4/5
1418/1418 [=====] - 20s 14ms/step - loss:
0.6074 - val_loss: 0.61345
Epoch 5/5
1418/1418 [=====] - 19s 14ms/step - loss:
0.6089 - val_loss: 0.6149

```

FIG-12 A CNN TRAINING OUTPUT


```
genre_based_popularity('Animation')[['title', 'popularity']]
```

	title	popularity
32753	Minions	547.488298
25016	Big Hero 6	213.849907
47929	Captain Underpants: The First Epic Movie	88.561239
5522	Spirited Away	41.048867
48707	Despicable Me 3	36.631519
45789	A Silent Voice	34.852055
44363	Your Name.	34.461252
50057	The Emoji Movie	33.694599
4793	Monsters, Inc.	26.419962
39518	Zootopia	26.024868
6275	Finding Nemo	25.497794
21452	Despicable Me 2	24.823550

FIG -13A GENRE POPULARITY

```
Showing recommendations for user: 469
=====
Movies with high ratings from user
-----
Willy Wonka & the Chocolate Factory (1971) : Children|Comedy|Fantasy|
Musical
Paths of Glory (1957) : Drama|War
Rosencrantz and Guildenstern Are Dead (1990) : Comedy|Drama
Ronin (1998) : Action|Crime|Thriller
Deliverance (1972) : Adventure|Drama|Thriller
-----
Top 10 movie recommendations
-----
Forrest Gump (1994) : Comedy|Drama|Romance|War
Secrets & Lies (1996) : Drama
Boogie Nights (1997) : Drama
Truman Show, The (1998) : Comedy|Drama|Sci-Fi
American History X (1998) : Crime|Drama
Fight Club (1999) : Action|Crime|Drama|Thriller
Boondock Saints, The (2000) : Action|Crime|Drama|Thriller
Amelie (Fabuleux destin d'Amélie Poulain, Le) (2001) : Comedy|Romance
Lord of the Rings: The Fellowship of the Ring, The (2001) : Adventure|
Fantasy
Beautiful Mind, A (2001) : Drama|Romance
```

FIG -14A CNN MODEL RATING PREDICTIO

```

Number of users = 6040 | Number of movies = 3706
6040
The sparsity level of MovieLens dataset is 95.5%
Size of sigma: 50
Shape of sigma: (50, 50)
[[ 147.18581225    0.          0.          ...    0.
   0.          0.          ]
 [  0.          147.62154312    0.          ...    0.
   0.          0.          ]
 [  0.          0.          148.58855276 ...    0.
   0.          0.          ]
 ...
 [  0.          0.          0.          ... 574.46932602
   0.          0.          ]
 [  0.          0.          0.          ...    0.
 670.41536276    0.          ]
 [  0.          0.          0.          ...    0.
   0.          1544.10679346]]
Shape of U: (6040, 50)
All user predicted rating : (6040, 3706)
Shape of Vt: (50, 3706)

```

FIG -15A SVD MODEL OUTPUT

```

Out[7]:
   movieId  ... svd_rating
520      608  ...   3.693186
706      924  ...   3.453491
2798     3740 ...   3.382859
1730     2324 ...   3.352101
7180     7226 ...   3.344175
2743     3681 ...   3.324915
224       260 ...   3.280118
1218     1617 ...   3.262298
613       778 ...   3.245400
2259     2997 ...   3.242278
709       928 ...   3.153711

[11 rows x 4 columns]

```

FIG -16A HYBRID MODEL OUTPUT