UNIVERSITY OF SIEGEN

FACULTY OF SCIENCE AND TECHNOLOGY

A PROJECT REPORT ON

CASE STUDY ON FEATURE EXTRACTION FOR RECOMMENDER SYSTEMS

Submitted by
Sagar Sirbi I 1607563
Master Mechatronics

Under the guidance of
Prof. Dr-Ing. Joeran Beel
Mr. Lukas Wegmeth (Ph.D)

INTELLIGENT SYSTEMS GROUP (ISG)
University of Siegen
Dept. of Electrical Engineering and Computer Science
North Rhine-Westphalia (NRW), Siegen, Germany

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

With the explosive growth of online information, users are generally provided with endless movies, products and services. As a result, recommender systems have come to rescue and possess effective strategies to overcome such information load. The influence of deep learning has been revolutionizing over the decade in the area of recommender systems and brings in a number of opportunities to improve the performance of the recommender. Moreover, the influence of feature extraction techniques in recommender systems is not very well documented. Hence, this project focuses on the survey of state-of-the-art feature extraction techniques currently implemented in recommender systems. More concretely, we implement feature extraction techniques using an autoencoder for a deep neural network predictive model and a hybrid approach for feature extraction using a machine learning predictive model. We evaluate these models against the baseline, derived from traditional structured dataset, being fed into the model. Furthermore, we demonstrate our models performance on two different datasets of varying sparsity, namely MovieLens 100k and MovieLens1M. Lastly, we evaluate our models performance using Mean Squared Error and Mean Absolute Error.

1. INTRODUCTION

In a world of rapid technological breakthroughs and Internet usage, an expanding number of devices, and exponentially growing data, people have a difficult time finding the information they need, and recommendation systems seek to solve this problem [1]. Currently, recommender systems are becoming increasingly vital in enabling people to deal with increasing information overload and make the best decisions. Generally, recommender systems utilize the user’s preference from their past feedback on items to provide personalized recommendations. Hence, user’s preferences are classified into two broad categories: implicit and explicit feedback. User's true preferences are expressed by explicit feedback such as ratings to products or services in some scenarios. Moreover, users also express their feedback implicitly such as browsing history, clicking / not clicking on websites. There are different definitions for recommender systems; for the purposes of this project, we will make the general assertion that they deliver a rating to the user-item pair. There are three types of recommender system techniques: content-based, collaborative filtering-based, and hybrid. Collaborative Filtering tends to determine similar users like and recommendations to be provided considering the cluster of similar users and providing

recommendations based on the preference of the cluster. The content based filtering method uses additional information about items or users. In this method, items features are considered to make recommendations of items to a user, based on the user's likes and also by their implicit / explicit feedback. Hybrid methods seek to find the best of both worlds through a combination of content based and collaborative filtering methods. Furthermore, based on the existence of two-way interaction between the rating information and auxiliary information, we further divide hybrid methods in two sub-categories: loosely coupled and tightly coupled methods. The auxiliary information is once processed, and then features are provided to the collaborative filtering method in a loosely coupled methods. In contrast, tightly coupled methods automatically learn the features using auxiliary information and balance the influence of auxiliary information and rating.

In recent years, the development and improvement of recommendation systems has become a demand, with the most popular of them being used in various domains such as ecommerce, businesses, financial services, and social networking. However, most of the approaches are incapable of capturing non-triviality hidden in the interactions between user-item matrix and content information in collaborative and content-based filtering methods. Moreover, as a result of advancements in graphics processing unit hardware, speeding the neural network training and advances in software algorithms, deep learning methods are becoming increasingly popular demonstrating its cutting-edge performance in revolutionizing recommender systems, as well as the ability to learn more complex abstractions as effective and to capture complex relationships within data. Despite the effectiveness of deep learning methods, the performance of the algorithm is limited due to the high sparsity of data. Therefore, there is a need for alternative methods to improve the recommendation performance.

Feature extraction seeks to create a feature vector by optimally adapting the input data. This procedure is extremely vital for the success of the ML application because another primary goal is to extract important details from the input data concisely while also removing noise and redundancy, reducing the computation time and initial prediction error to improve the accuracy of the ML model. However, the actual influence on performance of such features in recommender systems is not very well documented. Hence, the aforementioned details form the basis for the motivation of the project. The scope of our study is to collect scientific evidence of the effect of feature extraction in recommender systems and to build a feature extraction pipeline for recommender

systems using Deep Neural Networks where the goal is prediction of numerical ratings and not for the case where output is ranking of items for the user. The main contributions of this project are as follows:

- We implement a machine learning algorithm to build a predictive model for recommendation tasks. Further, we use a hybrid approach of feature extraction based on clustering in the recommendation task.
- We propose a discriminative model *of Deep Autoencoders for Feature Learning in Recommender Systems* based on incorporation of features from embeddings and autoencoders in a deep neural network to predict user's ratings for a recommendation task.
- Lastly, we compare and evaluate models' performances incorporating a feature extraction approach against the benchmark models.

We organize the paper into six sections. Section 1 introduces the study and explains the motivation for this study. Section 2 looks at the technological background necessary for the proposed project and solutions. Section 3 covers the literature survey performed in regard to the proposed solution prior to its usage and implementation in the project. Section 4 explains in detail the methodology adopted in feature extraction and the predictive model used in our approaches. Section 5 provides insights on the experimental setup necessary for efficient performance of proposed approaches. Section 6 covers the results on the proposed approaches and lastly section 7 provides the conclusion and future scope of the project.

2. TECHNOLOGICAL BACKGROUND

This section addresses the technological background reference for this project necessary for the reader.

2.1. Machine Learning

Machine learning is a branch of AI that involves the development of algorithms and statistical models that allow machines to improve their performance on a task over time. It entails feeding data to a computer system and allowing it to gain knowledge from that data without being explicitly programmed for this purpose. Machine Learning is further categorized into the following:

- *Supervised Learning:* In supervised learning, the algorithm is trained on a labeled dataset, where the desired output is already known.

- *Unsupervised Learning:* In unsupervised learning, the algorithm tries to find patterns in an unlabeled dataset, and it is often used for clustering and dimensionality reduction.
- *Semi-supervised Learning:* In semi-supervised learning, the algorithm is trained on a combination of labeled and unlabeled data.
- *Reinforcement Learning:* Reinforcement Learning is a type of machine learning that focuses on training an agent to make decisions in an environment by performing certain actions and receiving rewards or penalties.

Machine learning is widely used in various fields, such as computer vision, natural language processing, and robotics, to mention a few. Despite its many benefits, machine learning also has some limitations. One of the main challenges is that the accuracy of the predictions made by machine learning algorithms depends on the quality of the data used to train them.

To summarize, machine learning is a rapidly expanding field with the potential to revolutionize a wide range of industries and fields. Machine learning has the capability to provide valued insights and enhance decision-making practices in a broad array of applications due to its ability to analyze and comprehend large amounts of data. However, it is critical to understand its drawbacks and to develop and implement machine learning algorithms responsibly.

2.2. OPTICS

Ordering Points To Identify Clustering Structure (OPTICS) belongs to the category of density based clustering algorithm and has the objective of identifying clusters of similar data in a dataset. Unlike other clustering algorithms such as K-Means that uses predefined distance metrics and predefined number of cluster, OPTICS dynamically identifies clusters on the basis of the locality of data points.

OPTICS begins with creating an ordered list of points in as dataset, called ‘ordering’ particularly used to identify the clusters of a dataset. Ordering is created beginning with an arbitrary data point and then visiting the neighborhood points based on the reachability distance. Reachability distance refers to how close the point is w.r.t. to the other data points and their local density.

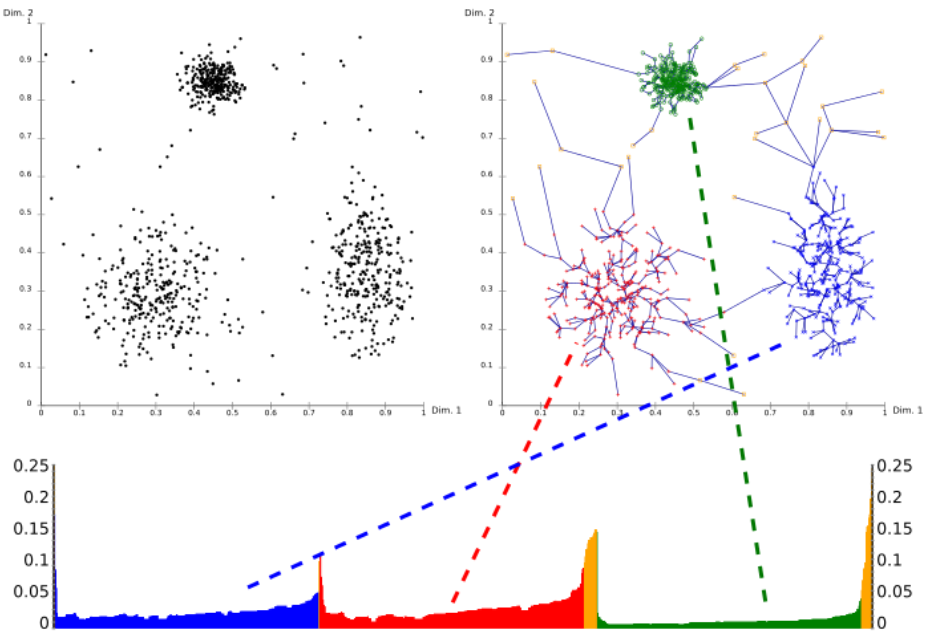

Figure 1: Illustration of cluster-ordering

The key advantages of OPTICS is its ability to identify clusters of varying density and shapes and overcomes the shortcomings of DBSCAN in detecting clusters of varying density. However, OPTICS is highly scalable and well suited for large datasets. The challenge of using the OPTICS algorithm is in the interpretation of results. The results of OPTICS are based on the ordering and are complex in nature, unlike other clustering algorithms providing clear labels for each data point.

In conclusion, OPTICS is a powerful algorithm for many data analysis and modeling tasks. Moreover, due to its flexibility, it is used for a wide range of applications. However, the results of the algorithm are complex in nature, there are available resources and tools to help users make sense of the results and apply the algorithm effectively.

2.3. K-Means

K-means clustering is an unsupervised machine learning algorithm for grouping similar points into a cluster. The user predefines a number and accordingly the algorithm divides the dataset into k number of clusters. The data points in a cluster are then used to calculate a centroid or mean, serving as a representative for the cluster.

The K-means clustering begins with random selection of k data points as centroids and then assigning each data point to the nearest centroid based on Euclidean distance. The algorithm is implemented in the following steps:

1. Randomly select k data points as cluster centers called ***centroids.***

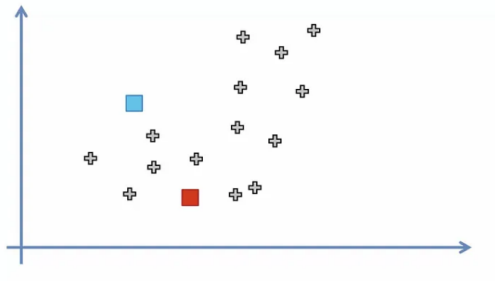

Figure 2: k data points as cluster centers

2. Calculate the Euclidean distance of each data point with regard to the centroids and assign the data points to the respective clusters.

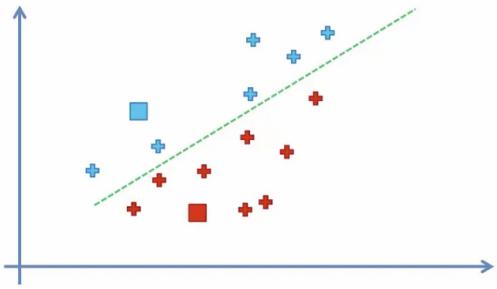

Figure 3: Assign data points to the respective clusters

3. Determine a new cluster centroid using the average of the assigned points in a cluster.

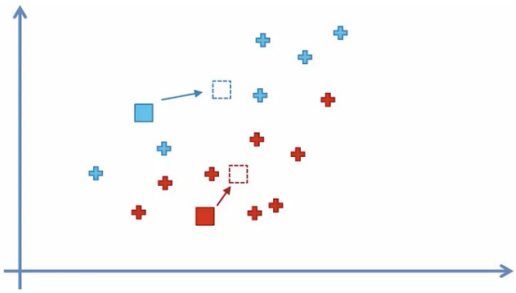

Figure 4: Determine new cluster centroid

4. Repeat steps 2 and 3 until none of the cluster assignments change.

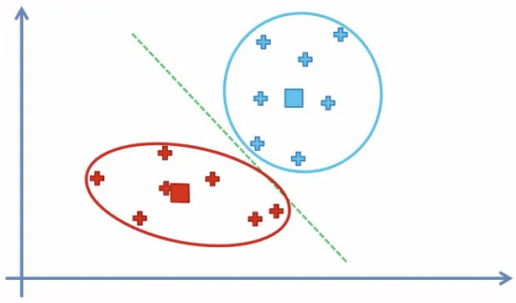

Figure 5: Illustration of cluster formation

The advantage of K-means clustering is its simplicity and ease of use. One of the prime factors to consider during K-means is the number of clusters (K) to be formed. A small value of K may not capture the complexity of data, and a large value of K may result in over-partitioning the data, hence it is important to consider an optimal value for K. There exists various methods to determine the optimal value of K, including the Elbow method and Silhouette Score. The algorithm may not perform well with non-globular shaped clusters and also with high dimensional data.

2.4. K-Prototypes

K-Prototype clustering, an extension of the K-means clustering algorithm is a combination of K-means and K-modes clustering algorithm, used for clustering mixed data i.e. data with both numerical and categorical variables.

K-prototype clustering starts by randomly selecting K centroids in the dataset, with each centroid consisting of a numeric vector and a categorical vector. The algorithm then assigns each point in the dataset to the nearest centroid, based on Euclidean distance for the numerical variable and based on hamming distance for the categorical variables. The centroids are further updated based on the mode for the categorical variables and mean for the numerical variables for data points assigned to the centroid. The process is repeated until convergence, where the centroid no longer changes.

The advantage of the K-prototypes clustering algorithm over traditional K-means clusters is it can handle mixed data, often encountered in real world applications. However, choosing a value for K in K-prototypes is a challenging task, as there are a number of parameters to consider, including numerical variables, number of categorical variables and number of clusters.

2.5. Deep Neural Network

Deep Neural Network (DNN) is a kind of Artificial Neural Network (ANN) consisting of multiple hidden layers between input and output layers. The term 'Deep' refers to the number of hidden layers in the neural network. Hidden layers serve the purpose of learning and extracting features from the raw input data, thus making it possible to solve complex tasks such as image classification, object detection, speech recognition and many more.

Deep Neural Network is a supervised learning approach, given the input output pairs and adjusting its parameters and thus minimizing the difference between predicted outputs and actual values. The training process involves two steps as follows:

- *Forward propagation:* Forward propagation is a key part of the training process in neural networks and helps models to learn patterns in data. Forward propagation is a process of calculating outputs for each node in a neural network, given the input values, multiplying the inputs by the weights, adding biases, passing the sum through an activation function to produce the node's output. This output acts as input to the next layer.
- *Backward propagation:* Backward propagation is a process of updating weights and biases in a neural network during training, as a result of reducing the differences between the actual values and the predicted values. This works by computing the gradient for the loss function with respect to the weights and biases and using optimization algorithms to update the parameters in the reverse direction of the gradient.

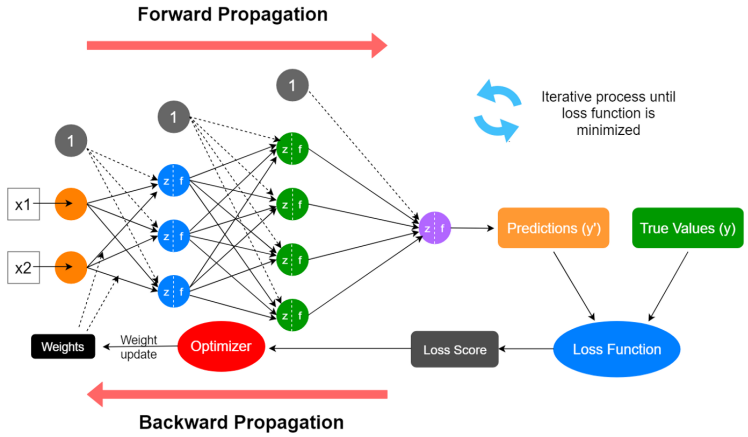

Figure 6: Forward and Backward propagation in a Deep Neural Network

2.6 Multilayer perceptron in Deep Neural Networks:

The Multilayer Perceptron (MLP) neural network architecture consists of an input layer, one or more hidden layers and an output layer. However, the nodes in a MLP are organized in layers and each node in a layer is connected to all the other nodes in the previous layer. Hence, these connections are represented by weights describing the relation between the node's strength. In the training time, weights are being adjusted to minimize the difference between the actual and predicted values.

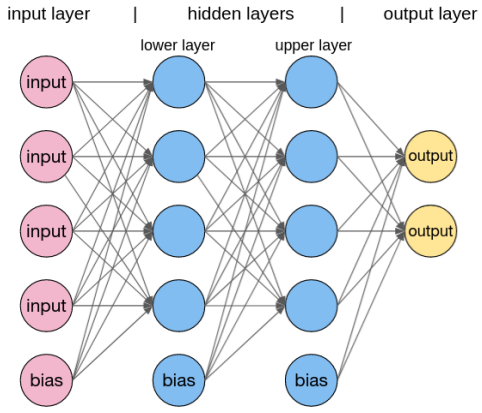

Figure 7: Multilayer perceptron neural network architecture

MLP holds the capability to learn complex representations by the addition of multiple hidden layers to the network, allowing the model to establish complex relationships between the input and output data. This also increases the risk of overfitting, resulting in poor performance on unseen data. To address this issue, batch normalization can be used to improve the training process, reducing the variance of outputs.

2.7 Autoencoders

An autoencoder is an artificial neural network that can encode and decode data, and it is made up of two components: an encoder and a decoder. The encoder takes the input data and compresses it into a lower-dimensional representation, called the encoding, by passing it through one or more hidden layers. The output of the final hidden layer is the encoding, which represents a condensed version of the original input data. The decoder takes the encoding and reconstructs the original input data by passing it through one or more hidden layers that increase the dimensionality of the data. The output of the final hidden layer is the reconstructed data, which should resemble the original input data as closely as possible. During training, the

autoencoder is trained to minimize the difference between the input data and the reconstructed output by adjusting the weights of the network using an optimization algorithm like stochastic gradient descent. The objective is to learn an encoding that captures the most important features of the input data while reducing the loss of information during the encoding and decoding process.

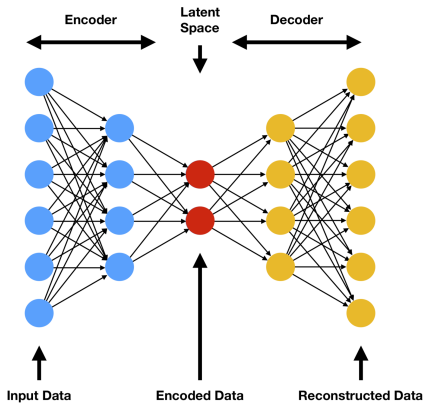

Figure 8: Autoencoder architecture

Data compression is one of the most common applications for autoencoders. Autoencoders can be used to reduce data storage requirements by learning to compress input data into a lower-dimensional representation. This is especially useful in applications with limited storage space.

3. LITERATURE SURVEY

3.1. Feature Representation Learning with Multilayer perceptron:

Multilayer Perceptron (MLP) also known as artificial neural networks can be used for feature learning as they are very straightforward and highly efficient, although it might not be as expressive as CNN's, RNN's and autoencoders.

Wide and Deep Learning belongs to the class of MLP and is used to solve problems related to both regression and classification. In wide and deep learning, the wide component can be regarded as a generalized linear model consisting of a single perceptron. The deep learning component consists of a multilayer perceptron. A blend of these two learning techniques allows the recommender systems to capture both generalization and memorization. Wide Learning possess capability to catch the direct features based on historical data. This is achieved by memorization in a wide model.

Meanwhile, generalization achieved by the deep learning component produces more general and abstract representations. As a result, this model can improve both the accuracy and the diversity of recommendation.

The wide learning is formally defined as $\boldsymbol{Y = W^{T}_{wide}\{x, \phi(x)\} + b}$ where, W^{T}_{wide} , b denotes model parameters and $\{x, \phi(x)\}$ denotes concatenated feature set consisting of raw input (x) and cross product transformation capturing the correlation between features ($\phi(x)$). While, deep neural component layer is defined as $\boldsymbol{\alpha^{(l+1)} = f(W^{(l)}_{deep} a^{(l)} + b^{(l)})}$ where, l denotes l[th] layer, $f(.)$ denotes activation function and $W^{(l)}_{deep}$ denotes weight and bias. The wide and deep learning model is achieved by culmination of the above two components defined as shown below

$$P(\hat{r}_{ui} = 1|x) = \sigma(W^{T}_{wide}\{x, \emptyset(x)\} + W^{T}_{deep} a^{(l_f)} + bias)$$

where $\sigma(.)$ is the sigmoid function, $\hat{r}_{ui}$ refers to binary rating label, $a^{(l_f)}$ is the final activation. The model is optimized using stochastic back propagation and, as s result, a recommendation list is generated on the basis of predicted scores.

Chen et al. [9] extended the wide and deep learning model, devising a locally connected wide and deep learning model focusing on large scale industrial level recommendation task. In this model, the deep neural network is replaced by a locally connected network, hence, decreasing the running time by one order of magnitude. An important step in wide and deep learning model is selection of features for wide and deep parts. Hence, the model requires to be capable of determining the features to be memorized and generalized. Moreover, the cross product transformation has to be designed manually. The above-mentioned steps greatly influence the performance of the model. The latter deep factorization-based model can help with feature engineering.

3.2 Feature Representation Learning with Convolutional Neural Networks (CNN):

CNNs are a type of deep learning algorithm used in computer vision and image processing tasks. They are capable of processing images, audio, text and consists of multiple layers, including convolutional, pooling and fully connected networks.

CNNs for image feature extraction: Wang et al.[10] examined the performance of visual features to Point-of-Interest (POI) recommendation, and proposed a visual enhanced POI recommender system (VPOI). VPOI extracts image features with the aid of CNNs. Recommendation model is based on two Probabilistic Matrix Factorization (PMF) exploring the interactions between visual content and latent user factor and visual content and latent location factor. On the other hand, the visual features extracted by CNN joint with text representations are fed as input into Bayesian Probabilistic Matrix Factorization (BPMF), Matrix Factorization (MF) to test their performance. We find that the visual information improves the performance to some degree, and is not very significant. Meanwhile, a personalized tag recommendation model based on CNNs was proposed by Nguyen et al.[11. This model uses a max pooling layer and a convolutional layer to get visual features of the patch of images. Personalized recommendation is generated by injecting user information. However, the model is optimized adopting the BPR objective to maximize the differences between relevant and irrelevant tags. A comparative deep learning model was developed using CNN for image recommendation. This network consists of an MLP for user preference modeling and two convolutional neural networks used for image representation learning. It compares a positive image liked by the user to a negative image disliked by the user against a user. This model takes into assumption that, the distance between the user and the positive image is closer than the distance between the user and the negative image. The metric for computing the distance is Euclidean distance.

CNN for audio and video feature extraction: Lee et al.[13] proposed on extracting audio features utilizing the prominent CNNs pretrained neural network ResNet. Meanwhile, Van et al. [13] proposed extracting features from music signals using CNNs. This content based model allows operations at multiple timescales, utilizing the convolutional kernels and pooling layers. However, this model can alleviate the cold start problem for music recommendation

CNNs for text feature extraction: DeepCoNN [14] uses review texts to model user behaviors and item properties using CNNs. This model solves the sparsity problem and improves model interpretability by combining CNNs with rich semantic representations of review texts. Further, this model employs a word embedding technique to map the review texts into a lower-dimensional latent space while retaining the word sequence information. The convolution layers take extracted review representations with different kernels, a max pooling layer, and a fully-connected layer in that order. The output of the

item and user network are concatenated and provided as the input to the prediction layer, where the factorization machine captures their interactions for rating prediction. ConvMF employs CNNs to learn high level item representations, whereas Collaborative Deep Learning (CDL) employs autoencoders to learn item feature representations. ConvMF has an edge over CDL in that CNNs can capture more precise contextual information on items through word embedding and convolutional kernels.

3.3. Feature Representation Learning using Autoencoders:

Autoencoders have the capability of powerful feature representation learning. Hence, autoencoders find applications in recommender systems to extract latent factors for the items and users content features. However, autoencoders are also deployed in user level as well as item level as stand-alone recommender systems. Autoencoder result in power ensembles as a result of stacking. Gaussian noise is added to stacked autoencoders, forming variations of stacked autoencoders and aid to generalize better. This helps beyond simply learning identity function and discovering more robust features. AutoSVD++, a variant of stacked autoencoders, learns item features making use of contractive autoencoder[122] integrating into the classic recommendation model, SVD++. AutoSVD++ posses the advantages of an efficient training algorithm designed to reduce training time. HRCD [15, 16] is a combination of autoencoder and timeSVD++. It is a time-aware model utilizing SDAE to solve cold start problem and learn raw features item representations. Deep Collaborative Filtering Approach treats dense and categorical features in a powerful network with leveraging side information resulting in outperformance of the model. The method incorporates a user and item autoencoder to learn compressed representations for user and item autoencoders. These compressed representations are used alongside latent features learned as a result of matrix factorization of the rating matrix. Side information[27] can be integrated to a collaborative filtering by extending construction of embeddings to neural networks. COFILS, Collaborative Filtering to Supervised Learning [28] achieves outperformance by collaborative filtering by using values of compressed representation of users and items. In comparison to CDL, marginalized Denoising Autoencoder based Collaborative Filtering model (mDA-CF) is an efficient model saving computational efforts in searching for a sufficient corrupted version of input by marginalizing the corrupted input, making mDA-CF more scalable than CDL. Moreover, CDL considers only the effectiveness of item features, whereas mDA-CF embeds content information of items and users.

The current approaches for feature learning using autoencoders consists of several drawbacks. Poor prediction accuracy is achieved using a stand-alone autoencoder for recommender systems. As a result, they have not been effectively adopted for collaborative filtering approaches. Deep collaborative filtering methods suffer from feature engineering from side information tending to be domain specific. Likewise, COFILS suffers from rich representation learned through embeddings. In our approach, we incorporate activations of the bottleneck layers for the user and item autoencoder along with embeddings as input in the deep neural network instead of using an AutoRec. In comparison to DCF our approach does not incorporate side information and only makes use of user and item identifiers only. Hence, the proposed approach possesses the following advantages: (1) Model is domain agnostic and no domain specific information is engineered (2) Model is scalable and can take the advantage of side information. The semantic similarity between our approach and COFILS consists of latent features learned as features by the autoencoders in a supervised learning model, a deep neural network in our approach and random forest in A-COFILS.

4. METHODOLOGY

4.1. Deep Autoencoder for feature extraction with embeddings:

The methodology for this approach consists of two major parts, namely extraction of features from an autoencoder and making predictions in a deep neural network incorporating features. Feature extraction is primarily achieved through training a user and item autoencoder. Prediction in a neural network is achieved by a combination of features extracted from a user and item encoder alongside the embeddings corresponding to the user and item identifiers.

4.1.1. Item Autoencoder and User Autoencoder for Feature Extraction:

The first step to the solution involves building an autoencoder for feature extraction. Hence, a rating matrix is created utilizing the pivot operation on a list of user, item, rating tuples in a Movielens-1M dataset. In case of the user autoencoder, a user-item matrix is created and vice versa for an item autoencoder. Given the table R corresponds to the ratings matrix and R^{m*n} corresponds to the user-item ratings matrix, $U = \{U_1, U_2, U_3, ... U_m\}$ is a list of users and $I = I\{_1, I_2, I_3, ... I_n\}$ is a list of items. R^T is the transpose of R denoting the item-user matrix. The input to the user autoencoder is a list of individual rows of user-item matrix, $r^{(u)}_u = \{r_{u1}, r_{u2}, r_{u3}, ... r_{un}\}$, as a result has input identifiers as input and item identifiers as features for the user autoencoder[1].

Symbol	Meaning
$U, I; r_{i,j,t}$	Number of users and items; rating of an item j by user i at time t
$R \in R^{m*n}$	User-item rating matrix
$r_u^{(u)}$	Row of ratings to items by user u
R_{uy}	Rating of item y by y user u
$r_i^{(i)}$	Row of ratings given by users to item i
R_{ix}	Rating to item x by a user i
W_H , b_H	Weight matrix and bias of the affine portion of the layer
h_i	Hidden nodes in the ith layer
Re*LU*(x)	Keeps the positive values as-is and transforms the non-0 values to 0
p	Dropout probability
concat($v1$, $v2$)	Combine vectors v1 and v2 into a single vector
n_batch	Number of batches
$\sigma(t)$	$\sigma(t)= \frac{1}{1+e^{-x}}$
x_t	One-hot vector for the levels in the categorical variable
$V * x_t$	Embedding for the categorical variable
K_U	User factor
K_I	Item factor
n_emb	Number of embeddings
$f(r: \theta)$	Reconstruction of input $r \in R^d$

Table 1: Notation in deep learning model approach

The user autoencoder takes $r^{(u)}_u$ as the values of the input and the autoencoder tries to recreate the input values at the output, passing through a low dimensional latent space. The autoencoder is capable of learning the identify function, feeding the input values through various order feature interactions accumulated in the innermost layers. Formally, the expression for the equation is as follows:

$$min \sum_{r \epsilon S} ||(r - f(r, \theta))||^2$$

The user autoencoder architecture is as shown below

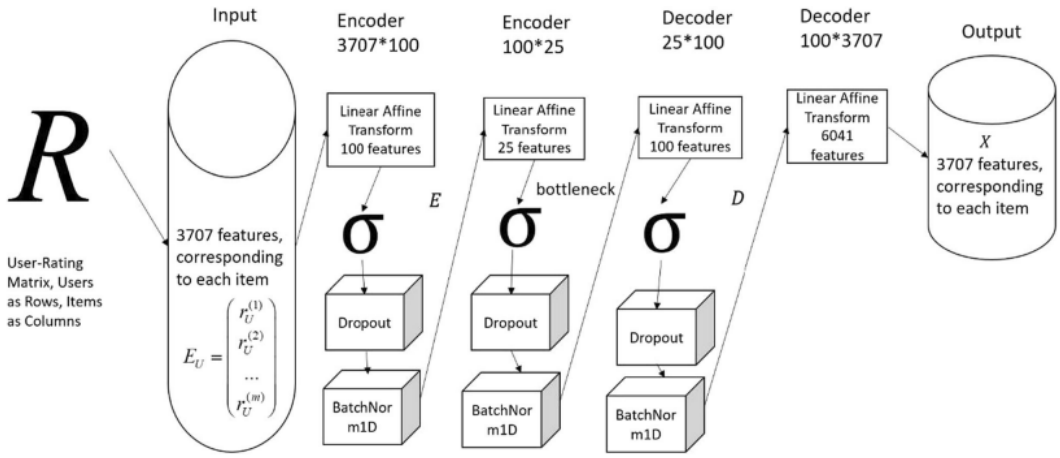

Figure 9: User autoencoder architecture

We observe that the encoder follows an affine transformation followed by a non-linear transformation. Further, a low dimensional affine transformation is performed alongside non-linear sigmoid transformation. The decoder is the exact mirror image of the encoder in the opposite direction, as a result the number of layers in the autoencoder increases. The equations for the user autoencoder are given by the equations 1 to 3.

$$E = \sigma(W_2(\sigma(W_1 R^l + b_1)) + b_2) \quad (1)$$

$$D = \sigma(W_3.E + b_3) \quad (2)$$

$$f(r, \theta) = W_4.D + b_4 \quad (3)$$

The parameters (Theta) are learned with backpropagation. Batch normalization is added after each layer to ensure efficient autoencoder training. However, the regularization is achieved by the weight decay parameter in the Adam optimizer and dropouts in various layers of encoder and decoder. As a result, the approach generalizes [1].

The item autoencoder architecture is very similar to the user autoencoder, but only varies in the values of the inputs. The item autoencoder architecture is as shown below

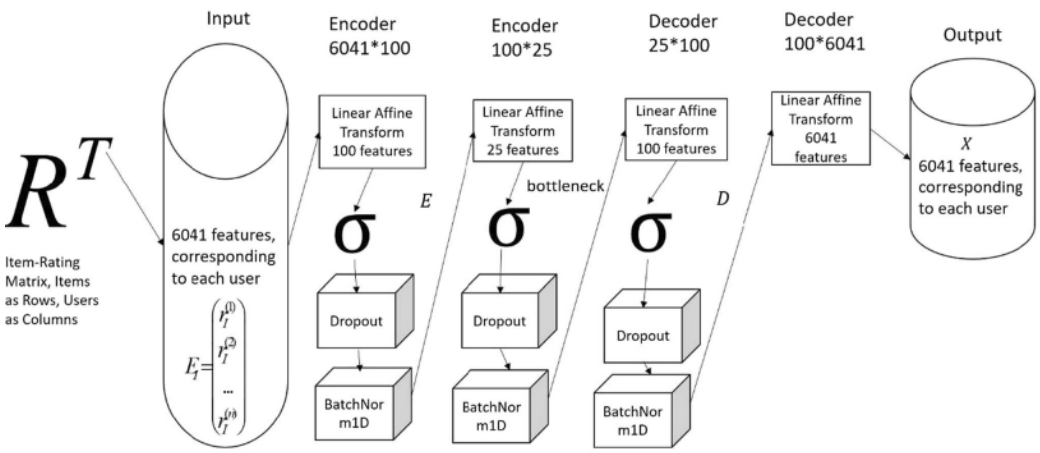

Figure 10: Item autoencoder architecture

The equations for the item autoencoder are similar to the equations of the user autoencoder, given by equations 4 to 6.

$$E = \sigma(W_2(\sigma(W_1 R^U + b_1)) + b_2) \quad (4)$$

$$D = \sigma(W_3.E + b_3) \quad (5)$$

$$f(r, \theta) = W_4.D + b_4 \quad (6)$$

4.1.2. Deep Neural Network with user and item autoencoder features and embeddings as input:

The second step of the solution involves making ratings prediction by the user, involving the autoencoder features and embeddings for the users and items. Embeddings are considered to be capable of learning a rich representation in multiple dimensions for the identifier. Embeddings are alternative representations for dummy variable coding and one-hot encoding in conventional machine learning. However, the weights of the embeddings are learned the traditional way as any other features, as in autoencoder derived features in our proposed solution.

The solution primarily consists of two hidden layers, that are affine transformations followed by ReLU transformation. The dropouts introduced in the layers regularize and aid in generalization. Dropouts help to ensure faster convergence in the intermediate layers. The equations for the hidden layers are as shown in equations 7 to 9 . The

hidden layers perform an affine and nonlinear ReLU activation in sequential manner. The output of the innermost layer is fed into a sigmoid activation function, illustrated in equation 4. The output of the neural network is a real number in the range of 0 and 1, further transformed in the ratings scale. The overall list of operations is illustrated below:

$$X_1 = dropout_1(ReLU(W_1 * X_0 + b_1)) \quad (7)$$

$$X_2 = dropout_2(ReLU(W_2 * X_0 + b_2)) \quad (8)$$

$$X_3 = dropout_3(ReLU(W_3 * X_0 + b_3)) \quad (9)$$

$$\sigma_{output} = \sigma(X_3) \quad (10)$$

$$Output = rating_{min} + {}_{output} * (rating_{max} - rating_{min}) \quad (11)$$

$$\sigma_{output} = \sigma(dropout_3(ReLU(W_3 * dropout_2 \quad (12)$$

$$(ReLU * (W_2 * (dropout_1(ReLU(W_1 * X_0 + b_1))) + b_2)) + b_3))$$

The first step of the proposed model will create a user autoencoder consisting of 4000 features, followed by an item autoencoder with 6000 features. In the second step, the activations from the bottleneck layers of the user autoencoder and item autoencoder are extracted, consisting of 25 features each for the user and items. The third step involves utilizing a deep neural network alongside 25 features from user and item autoencoder with user and item embeddings. The output of a deep neural network is the ranting prediction for the user.

Figure 11 represents the overall solution architecture

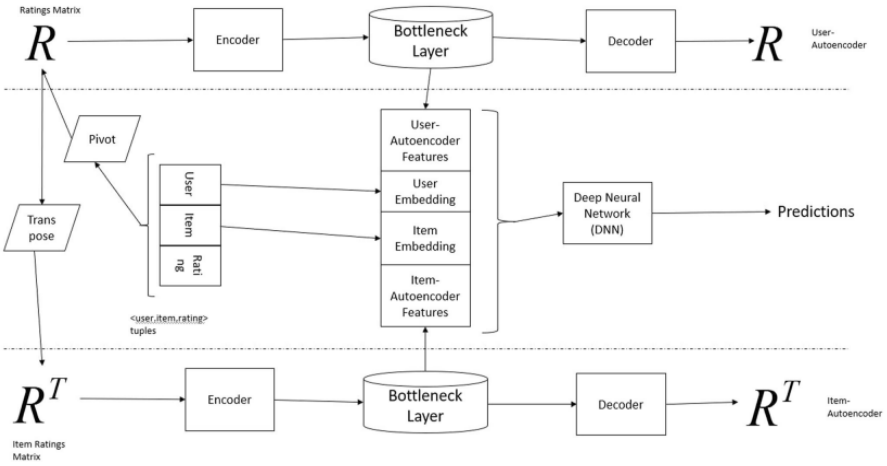

Figure 11: Proposed Model overall flow

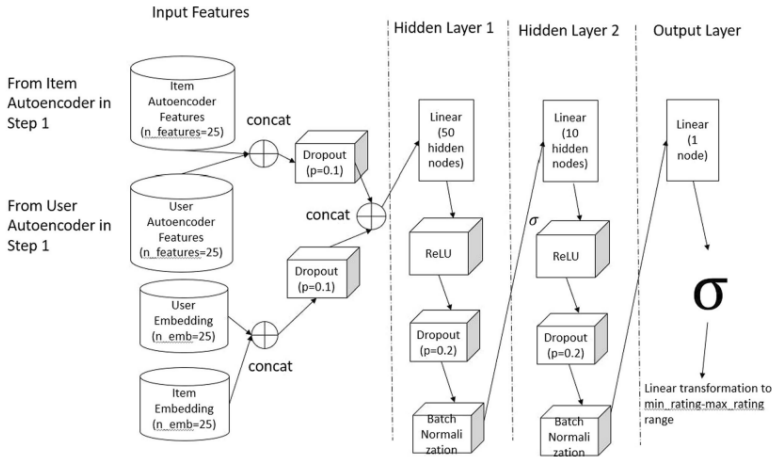

Figure 12: Proposed model architecture

4.2. Hybrid approach of feature extraction using clustering

The methodology in this approach, primarily, we focus on feature extraction on Movielens-1M dataset in order to find users in relation to the movies utilizing the clustering techniques. Eventually, the traditional techniques are customized to comprise distinct features unique to the dataset including age, gender, occupation and zip code. Further, we focus on rating predictions by the user using a machine learning algorithm. Alongside we have also considered users and items dataset specifically focussing the predictive model addressing both collaborative filtering and content based recommender systems.

4.2.1. Feature Extraction using Data Clustering

The first step to the solution involves clustering the whole dataset using a selected clustering algorithm. We assume that the number of clusters is either specified as a parameter or determined automatically by the algorithm. For each cluster, a cluster representative is selected, summarizing the cluster's characteristics. If the clustering algorithm doesn't produce any cluster representatives, then the cluster representatives are manually selected like the smallest average distance among the data points in a given cluster [3]. These cluster representatives in culmination with all the original data samples compute new features using a distance measure. As a result, the new features are equal to the number of clusters. Given the dataset of training samples $X = \{x_1, x_2, x_3, \ldots x_n\}$ with features $\mathcal{f} = \{F_1, F_2, F_3, \ldots F_m\}$ followed by distance measure $\delta(x_1, x_2)$ results in clustering (grouping) of data points. The new features, one for each cluster representative $c_l \in C$, $l = 1..k$ are added to the training samples $F_{j+1}, F_{j+2},\ldots, F_{j+k}$. The

augmented dataset $x_i = (x_{i1},..., x_{im}, x_{im+1},..., x_{im+k})$ is constituted by both the old and new features. The augmented dataset is used to further train a random forest regressor, storing the cluster representatives in order to transform new unlabeled data points and avoid retraining of the model. This results in modifying the unlabeled examples into the same feature space as training samples [3].

4.2.2. Random Forest Regressor with augmented dataset:

The second step of the solution involves determining the rating predictions by the user using the dataset augmented with new features. These new features result from combining data from various knowledge sources and data clustering. In the next step, we prepare the data in a form that can be used for analysis in a machine learning algorithm. Further, the model utilizes k-fold cross validation technique for evaluation, hence we divide the data into k-folds. A recommendation system is then built using Decision Regression Trees (Random forest Regressor) to predict ratings by the user. As a result, the model learns the decision rules by inferring the data features characteristics and making predictions for the users ratings.

5. EXPERIMENTAL SETUP

5.1. Datasets

Datasets used in this project include MovieLens 100k and MovieLens 1M dataset. These datasets were released by GroupLens research group at the University of Minnesota, containing information about the users ratings of movies belonging to MovieLens website. MovieLens 100K and MovieLens 1M are widely used in recommender systems research, including collaborative, content based and hybrid approaches. The datasets are used in developing users preference and generating recommendation lists in recommender systems. MovieLens 100k consists of 100,000 ratings of movies, scaling between 1 and 5. The dataset consists of 1682 movies and 943 users with each user rating a minimum of 20 movies. MovieLens 1M dataset consists of 1 million ratings of 3900 movies by 6040 users with a minimum of 20 movies per user. These datasets also consist of demographic information of users such as age, gender, occupation and zip code, including movies such as movie title, genre, etc.

5.2. Clustering

In our *machine learning* proposed model, we use the following clustering algorithms: (1) KModes (2) KPrototypes and (3) OPTICS clustering. The mentioned clustering algorithms require the number of cluster parameters to be set manually. Usually, the value of K is selected based on domain knowledge or using clustering evaluation metrics such as silhouette score or elbow method. In silhouette score, each point in a cluster is compared against other clusters. Higher silhouette scores indicate better clustering performance. However, the elbow method plots the within-cluster sum of squares (WCSS) as a function of the number of clusters. Then, identifying the 'elbow' in the plot determines the diminishing point in terms of WCSS reduction. The point at which an elbow is formed represents the optimal number of clusters for the data. In our proposed approach, we utilize the elbow method in order to find the optimal parameters for the number of clusters k in a clustering algorithm.

5.3. New features using distance measure :

In the proposed *machine learning* model on *Hybrid Approach of feature extraction for Recommender systems,* we augment the features of the dataset with new features. The number of new features is determined by the number of cluster parameters selected for the algorithm. For OPTICS clustering, the cluster representative is not determined by the available library, hence we compute the cluster representative manually. We determine the cluster representative by selecting the data point with minimum reachability distance in a given cluster. In KModes and KPrototypes we readily use an available library function to determine the cluster representative, and it corresponds to the centroid in a cluster. Further, the cluster representative $c_l = \{a_1, a_2, a_3, \ldots. a_{m+k}\}$ has the same shape as the shape of the input to the clustering model $X = \{x_{i1}, x_{i2}, \ldots. X_{im}, x_{im+k}\}$. Hence, we compute a Euclidean distance as shown below:

$$d_{euc}(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - a_i)^2}$$

Where a_i is the cluster representative and x_i is the data point.

5.4. Data preprocessing

Data preprocessing is considered to be an essential step in data analysis pipeline involving transformation of raw data into a format that can be easily understood by a machine learning algorithm and as a result, improving the accuracy and efficiency of

the model enabling better decision-making based on results obtained. Hence, we performed data preprocessing on both the proposed models in our project. The data preprocessing on our proposed *deep learning* model is processed as follows:

- *User and item index creation:* We build dictionaries that map distinct user and item IDs to distinct integer indices. These dictionaries are used to convert user and item IDs in raw data into integer indices that machine learning algorithms can use.
- *Encoding categorical variables:* We build dictionaries that map distinct values of categorical variables (such as gender, age, title, genre, and zip) to distinct integer indices. These dictionaries are employed to encode categorical variables as integer indices for use in analysis.
- *Creating sparse matrices:* The user and item indices are used to generate sparse matrices that represent the user-item ratings, as well as binary indicators that indicate whether each rating is in the training or validation set. This step entails grouping the data by user and item indices, and then using those groupings to generate sparse matrices.
- *Creating separate train and validation sets:* We create separate train and validation sets using sparse matrices and binary indicators. For every set, multiply the rating matrix by the binary indicator matrix, resulting in a matrix containing only the ratings from the desired set.

5.5. Neural Network Architecture:

In the proposed deep learning model, we utilize three neural networks for our proposed model as shown below:

- Autoencoder: The autoencoder in the proposed *deep learning* model consists of the following architecture:
    - *Encoder:* The encoder is sequential in nature and is made up of two linear layers. The linear layers are followed by sigmoid activation functions, a dropout layer and a batch normalization layer. The first layer takes an input of size corresponding to the number of items in the input data. As a result, the autoencoder produces a code of size hidden[0]. The second layer utilizes the first layer's output and generates a code of size hidden[1], representing the compressed input.
    - *Decoder:* The decoder is similar to the encoder and consists of one single layer, sigmoid activation, dropout layer and batch normalization layer. This layer takes a compressed input of size hidden[1] and produces a

code of size hidden[0]. Further, this layer is passed through the sigmoid activation, dropout and batch normalization resulting in a code of size as the input to the encoder, representing the reconstructed input.

- *Fully connected layer:* This layer produces a linear transformation mapping the compressed input of size hidden[1] to the original size fed to the encoder.
- User Autoencoder architecture:

```
autoencoder(
  (encoder): Sequential(
    (0): Sequential(
      (0): Linear(in_features=3707, out_features=25, bias=True)
      (1): Sigmoid()
      (2): Dropout(p=0.2, inplace=False)
      (3): BatchNorm1d(25, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
    (1): Sequential(
      (0): Linear(in_features=25, out_features=10, bias=True)
      (1): Sigmoid()
      (2): Dropout(p=0.2, inplace=False)
      (3): BatchNorm1d(10, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (decoder): Sequential(
    (0): Sequential(
      (0): Linear(in_features=10, out_features=25, bias=True)
      (1): Sigmoid()
      (2): Dropout(p=0.2, inplace=False)
      (3): BatchNorm1d(25, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (fc): Linear(in_features=25, out_features=3707, bias=True)
  (criterion): MSELoss()
)
```

- Item Autoencoder architecture:

```
autoencoder(
  (encoder): Sequential(
    (0): Sequential(
      (0): Linear(in_features=3707, out_features=25, bias=True)
      (1): Sigmoid()
```

```
      (2): Dropout(p=0.2, inplace=False)
      (3): BatchNorm1d(25, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
    (1): Sequential(
      (0): Linear(in_features=25, out_features=10, bias=True)
      (1): Sigmoid()
      (2): Dropout(p=0.2, inplace=False)
      (3): BatchNorm1d(10, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (decoder): Sequential(
    (0): Sequential(
      (0): Linear(in_features=10, out_features=25, bias=True)
      (1): Sigmoid()
      (2): Dropout(p=0.2, inplace=False)
      (3): BatchNorm1d(25, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (fc): Linear(in_features=25, out_features=3707, bias=True)
  (criterion): MSELoss()
)
```

- *Deep Neural Network:* This model architecture is popularly used for collaborative filtering, a popular technique in recommender systems. However, this model uses an embedding layer to capture the latent features of users and items based on their ID's , biases to capture the inherent rating tendencies, a sigmoid activation function followed by evaluation metric mean squared error loss function for training the model and minimizing the errors between actual and predicted ratings.

```
cf(
  (emb_user): Embedding(6041, 50)
  (emb_item): Embedding(3707, 50)
  (emb_dropout): Dropout(p=0.01, inplace=False)
  (ub): Embedding(6041, 1)
  (ib): Embedding(3707, 1)
  (sigmoid): Sigmoid()
  (criterion): MSELoss()
)
```

- *Deep Neural network:* This is a collaborative filtering model architecture that predicts ratings by users based on user and item IDs. The model consists of two embedding layers for users and items outputting 50 output dimensions. Moreover, two additional embedding layers for users and items bias outputting 1 output dimensions. Dropout layer is applied to the latter input embeddings. Furthermore, the model also consists of three fully connected layers with ReLU activation and dropout, gradually reducing the dimensionality of the input. A final layer is linear with 1 output dimension and a sigmoid activation function. The model is trained using mean square error criteria, minimizing the errors between actual and predicted ratings.

```
cf(
  (emb_user): Embedding(6041, 50)
  (emb_item): Embedding(3707, 50)
  (ub): Embedding(6041, 1)
  (ib): Embedding(3707, 1)
  (emb_dropout): Dropout(p=0.1, inplace=False)
  (lin1): Sequential(
    (0): Linear(in_features=100, out_features=25, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.2, inplace=False)
  )
  (lin_bias): Linear(in_features=3, out_features=1, bias=True)
  (lin2): Sequential(
    (0): Linear(in_features=35, out_features=10, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.3, inplace=False)
  )
  (lin3): Sequential(
    (0): Linear(in_features=36, out_features=10, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.2, inplace=False)
  )
  (lin4): Linear(in_features=10, out_features=1, bias=True)
  (sigmoid): Sigmoid()
  (criterion): MSELoss()
)
```

5.6. Evaluation Metrics

The proposed model utilizes multiple evaluation metrics to examine the accuracy of the proposed model like L1-error (also called MAE) and L2-error (also called MSE) which are calculated using the equation below:

$$\text{RMSE} = \sqrt{\sum_{i,j}^{m,n} T_{ij}(r_{ij}^{actual} - r_{ij}^{predicted})^2}$$

$$\text{MSE} = \sum_{i,i}^{m,n} T_{ij}(r_{ij}^{actual} - r_{ij}^{predicted})^2$$

$$\text{MAE} = \sum_{i,i}^{m,n} T_{ij}\,|(r_{ij}^{actual} - r_{ij}^{predicted})|$$

6. RESULTS AND EVALUATION

In the proposed machine learning model, we utilize three clustering methods in our approach and address both collaborative filtering and content based filtering. In collaborative filtering, initially, the project implements a machine learning model without clustering the data and consider model predictions for comparison and is termed as Baseline model. However, the approach implements various clustering approaches for feature extraction followed by data augmentation and model prediction. These models are termed as test models. The RMSE and MAE score for the baseline and test models are shown below:

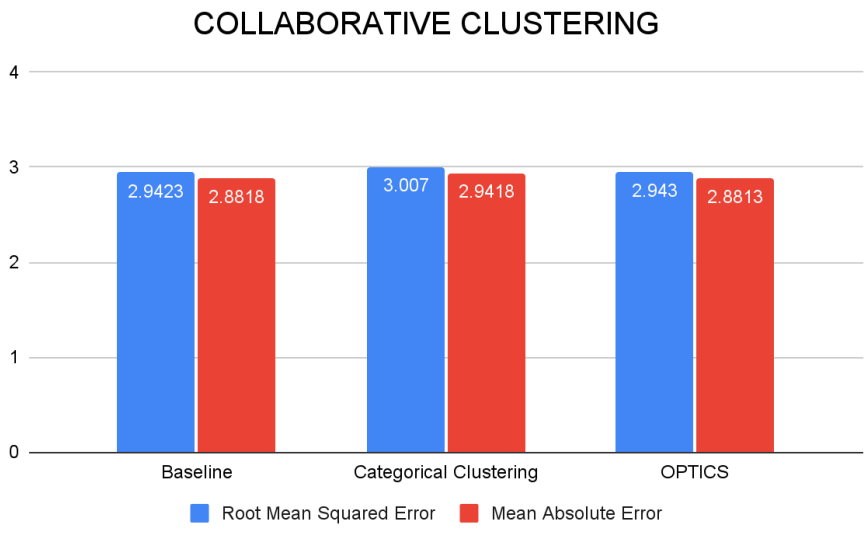

Figure 13: Collaborative filtering results

From the Fig.[1] we observe that, predictive model with optics clustering is on comparable scale with baseline model. Hence, the performance of the model doesn't improve as a result of clustering. In contrast, the performance of the predictive model with categorical clustering underperforms (drop of 2.15% in the RMSE score) in comparison to the baseline model without clustering.

In content based filtering, the approach followed for clustering and analysis is similar to collaborative filtering. The RMSE and MAE scores for the baseline and test models are shown below:

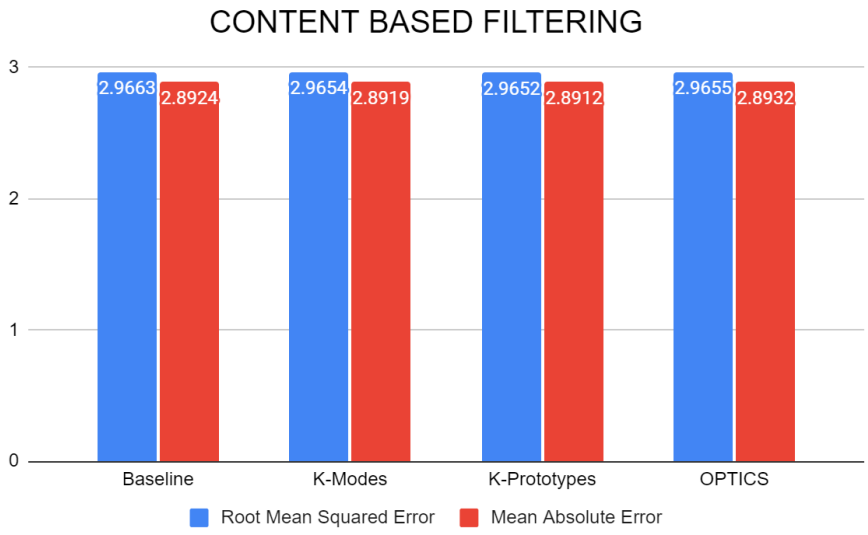

Fig 14: Content based filtering results

In Fig 14 we find that, the predictive models with K-Modes, K-Prototypes and OPTICS clustering are on comparable scale with other in terms of RMSE and MAE score. Moreover, these models are on comparable scale with the baseline models. Hence, we conclude that feature extraction technique with data clustering for rating prediction model using Movie lens 100K dataset doesn't improve the performance of the model by a significant value.

In the proposed deep learning model, the effectiveness of using autoencoders with embeddings in a deep neural network is examined against some key research papers in the area. The model utilize ten-fold cross validation technique, ensuring reliability of the proposed model results. The results show a significant improvement in the performance of the proposed model. The model utilizes MovieLens-1M dataset for

experiments and MSE, MAE and RMSE metrics for evaluation. The results are shown below:

Dataset	Reference Paper	RMSE	MAE
MovieLens-1M	I-AutoRec	0.831	NA
	mDA-CF	0.8416	NA
	SVD++	0.851	0.668

Table 2: Benchmark models for comparison

In our proposed model, we obtain a RMSE score 0.849 and MAE score 0.670. In comparison to table 2, we infer that our model has improved performance of 0.23% in terms of RMSE and -0.29% in terms of MAE. Further, we compare the proposed model results against a baseline model (Deep Neural Network with embeddings), as shown below:

Dataset	Model	Latent Features	RMSE	MAE
MovieLens -1M	DNN with embeddings	Linear	0.897	NA
	DNN with autoencoder and embeddings	Linear and Non-Linear	0.849	0.67

Table 3: Proposed model comparison

From table 3, we infer that, the proposed model achieves an improved performance of 5.35% in RMSE score. Hence, our proposed model outperforms the baseline model and the benchmark model: SVD++ in terms of RMSE score, as a result of efficient non-linear latent features captured by the user and item autoencoder in the proposed model.

7. CONCLUSION AND FUTURE SCOPE

Predictive accuracy in recommender systems plays a prime role in limiting the information overload, resulting in better choice for users and improving the profits for business and services. In our project, we propose two model approaches focusing on the feature extraction techniques in machine learning and deep learning models for recommender systems. In the first approach, we implement a machine learning

algorithm for making predictions. However, we mainly focus on feature extraction in this model and use data clustering for efficient feature extraction. As a result, we augment the dataset with new features obtained from the clustering. This results in improved performance of the model in comparison to the baseline model. In the second approach, we propose a deep neural network with latent features from autoencoders and embeddings for making predictions. The model combines the user and item identifiers as embeddings alongside the innermost activation from the autoencoder as features in a deep neural network. The deep neural network is capable of learning higher order innermost activations from a user and item autoencoder, consisting of non-linear latent features alongside linear features as a result of embeddings from user and item identifier. This results in a rich feature set exploited by the neural network, improving the performance of the model. Our approach provides an effective way to learn linear and non-linear features alongside integrating auxiliary information easily into our architecture, demonstrating the outperformance on MovieLens 1M dataset.

We find scope for future research in the proposed models. In the first model, *hybrid approach for feature extraction technique in recommender systems* we could extract more features through information retrieval techniques to further increase the performance of the model reducing the RMSE score. Moreover, we can further use principal component analysis and dimensionality reduction on the extracted features. In the second model*, Deep Autoencoders for Feature Extraction with embeddings for deep neural network,* side information of users and items could add additional value to the performance of the recommender systems. A study could incorporate the benefits of side information for users and items with few ratings versus the ones with many ratings. As a result, the auxiliary information in several cases is comparable. Further, a denoising autoencoder could also be incorporated in the model. This results in increasing the generalization capability of the model and prevents learning for only an identity function. The limitations of the proposed model include slow training of autoencoders in comparison to other available linear factorization methods. Hence, future work could include optimization of computation time and prediction time for the model.

---

https://github.com/ISG-Siegen/SA_Sagar_Sirbi

8. REFERENCES

[1] Rama, K., Kumar, P. and Bhasker, B. (2021). Deep autoencoders for feature learning with embeddings for recommendations: a novel recommender system solution. Neural Computing and Application, 33:14167–14177.
[2] Salinca, A., A hybrid approach of feature extraction for content-based recommender system.
[3] Piernik, M., Morzy, T., A study on using data clustering for feature extraction to improve the quality of classification (2021). Knowledge and Information Systems, 63:1771–1805
[4] Alhejaili, A., and Syeda, F. (2020). Latent Feature Modelling for Recommender Systems. Loughborough University.
[5] Wen, Z., Miao, K., and Shang, Y (2021). Creator2Vec: Creator Feature Embedding for Deep Learning Recommender System. IEEE International Conference on Networking, Sensing and Control (ICNSC) | 978-1-6654-4048-6
[6] Zhang, S., Ma, X., and Wang, Y (2022). An embedding and interactions learning approach for ID feature in deep recommender system. Expert Systems with Applications 210, 118425.
[7] Zhang, S., Yao, L., and Tay, Y. (2018). Deep Learning based Recommender System: A Survey and New Perspectives. ACM Computing Surveys, vol 1, No. 1, Article 1.
[8] Singh, S., Lahakare, M., and Sayar, K. and Sharma, S. (2021). RecNN: A Deep Neural Network based Recommendation System. International Conference on Artificial Intelligence and Machine Vision (AIMV).
[9] Chen, C., Zhao, P., and Li, L. (2017). Locally Connected Deep Learning Framework for Industrial-scale Recommender Systems.
[10] Wang, S., Wang, Y. and Tang, J. (2017). What Your Images Reveal: Exploiting Visual Contents for Point-of-Interest Recommendation.
[11] Nguyen, H., Wistuba, M., Grabocka, J. and Drumond, L. (2017). Personalized Deep Learning for Tag Recommendation.
[12] Lee, J., Abu-El-Haija, S., and Varadarajan, B. (2018). Collaborative Deep Metric Learning for Video Understanding.
[13] Oord, A., Dieleman, S., and Schrauwen, B. (2013). Deep content-based music recommendation.
[14] Zheng, L., Noroozi, V., and Yu, S. (2017). Joint Deep Modeling of Users and Items Using Reviews for Recommendation.

[15] Wei, J., He, J., and Chen, K. (2016). Collaborative Filtering and deep learning based hybrid recommendation for cold start problem. IEEE, 874–877.
[16] Wei, J., He, J., and Chen, K. (2017). Collaborative filtering and deep learning based recommendation system for cold start items. Expert Systems with Applications, 69, 29–39.