(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

EVALUATING THE PERFORMANCE OF RECOMMENDATION SYSTEMS USING ACCURACY, SCOPE AND SIMILARITY METRICS

A Report for the Evaluation 3 of Project 2

Submitted by

ANADI MISHRA

(1613101122)

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

Under the Supervision of
Mr. RAVINDRA KUMAR CHAHAR,
Assistant Professor, Galgotias University

APRIL / MAY- 2020

SCHOOL OF COMPUTING AND SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report "EVALUATING THE PERFORMANCE OF RECOMMENDATION SYSTEMS USING ACCURACY, SCOPE AND SIMILARITY METRICS" is the bona-fide work of "ANADI MISHRA (1613101122)" who carried out the project work under my supervision.

SIGNATURE OF HEAD
School of Computing Science & Engineering

SIGNATURE OF SUPERVISOR
School of Computing Science & Engineering

ABSTRACT:

The application of recommendation systems in e-commerce and streaming services has piqued the interest of many scholars, leading to an influx of research in this field in the past decade. While the majority of prior research on this field has been done to improve recommendation system's performance based on accuracy as the sole performance metric, the roles of other performance metrics like coverage and novelty have long been realized by researchers [1,2]. As a result, other performance metrics are increasingly being used for current research.

The aim of this study is to implement three popular recommendation systems and evaluate their performances using 'accuracy', 'scope' and 'similarity' as evaluation or performance metrics. Furthermore, a secondary objective of this work is to observe and report any trends or similarities between the performance metrics based on the results observed.

TABLE OF CONTENTS

List of Tables:

List of Figures:

List of Abbreviations, Symbols and Nomenclature:

List of Abbreviations:

i. MAE-Mean Absolute Error
ii. MAEP- Mean Absolute Error Percentage
iii. Cov- Covariance
iv. KNN- K-Nearest Neighbours
v. PCC- Pearson's Correlation Coefficient
vi. Avg- Average
vii. S.D.- Standard Deviation
viii. IEEE- Institute of Electrical and Electronics Engineers

List of symbols:

i. µ: Mean
ii. ∩: Intersection of two sets.
iii. ρx : standard deviation of x
iv. ρ X,Y: coefficient of correlation between X and Y
v. Σ: summation of elements
vi. ∈: belongs to
vii. = : equals to
viii. { } : represents a set
ix. |a| : absolute value of a
x. + :sum
xi. - : difference
xii. * : product
xiii. / : division
xiv. ∀: Universal Quantifier
xv. == : is equal to (condition)
xvi. != : not equal to (condition)

List of Nomenclature:

i. anime_id- unique id provided to each anime
ii. user_id- unique id provided to each user
iii. X_train-training set of independent variables
iv. X_test- test set for independent variables
v. y_train- training set of dependent variables
vi. y_test- test set for dependent variables
vii. y_pred- set of predicted values
viii. getNeighbors: function to produce recommended results for KNN model
ix. anime_user_matrix- a pivot table which contains ratings provided by the users to the anime.
x. getSimilarItems: function to produce recommendations of items similar to a given item in item-based collaborative filtering model.
xi. getSimilarUsers: function to produce a list of users similar to a given user in user-based system
xii. recommendAnime: function to generate a list of recommended items for a user in user-based system
xiii. similarity_df- dataset containing similarity values.

1. INTRODUCTION:

Recommendation systems have emerged as one of the more popular tools with growing popularity of internet marketplace and content streaming services. Large internet companies like Amazon, Google and Netflix all use their highly sophisticated recommendation systems to improve user experience and ensure higher visibility of certain products. A recommendation system can be defined as a model that constantly trains itself and provides a set of recommendations to users based on their previous interactions with the system [3]. One of the most popular techniques used in design of recommendation systems is collaborative filtering. Collaborative filtering technique makes recommendations based on similarities between users' interests from the ratings provided by the users to items [4]. Collaborative filtering models can be user-based, item-based or hybrid in nature. This has been discussed further under experiment section.

In order to evaluate recommendation systems implemented by different methodologies, three different systems have been implemented. First model is an item-based K-Nearest Neighbours model that has been implemented using machine learning technique, the second model is an item-based collaborative filtering model and the third model is a user-based collaborative filtering model. Both collaborative models are memory-based systems.

The evaluation metrics used in this study are accuracy, scope and similarity. Accuracy and scope are commonly used evaluation metrics for recommendation systems and ideas of similarity between recommended items has been proposed in earlier studies [1,2]. However, this study proposes a different method for calculation of degree of similarity between recommended items. The proposed similarity metric uses attributes like type of content- TV show, movie etc., number of episodes and number of ratings to provide a similarity score to all items in the dataset.

The datasets used for this work have been collected from www.myanimelist.net, one of the most popular anime aggregator websites, by user CooperUnion and have been made publicly available on Kaggle under CCO: public domain license. The reason behind the selection of these datasets is that despite the increasing popularity of anime in current popular culture, not much work has been done on anime recommender systems. Two datasets have been used for the study- an anime dataset and a user ratings dataset. The anime dataset contains data about 12,294 anime shows and movies based on the following attributes: “name”, “anime_id”, “genre”, “episodes”, “type”, “rating” and “members”. Similarly, the user ratings dataset contains the user ratings provided by over 73000 users for anime in the previous set. It contains the following attributes: “user id”, “anime id” and “rating”. The detailed description of both datasets has been given under the experiments section.

According to Wikipedia, “Anime is a hand-drawn and computer animation originating from Japan. The word anime is the Japanese term for animation, which means all forms of animated media. Owing to rapid growth of distribution platforms like Crunchyroll, Daisuki, Netflix, Amazon, among others, Japanese anime has found remarkable number of new takers. The live entertainment and internet streaming of such content has led to a substantial rise in international distribution of Japanese anime. Thus, internet distribution has become the most reliable and lucrative route for its distribution across the globe.

The recommendation systems implemented in this project are content recommendation systems, that recommend content to users based on their interests, or in case of user-based recommendation systems based on interests of similar users. From here on the terms item anime and content have been used interchangeably, due to the former being the formal term for research in recommendation systems and the later being the type of item being recommended.

The experiment section of this paper discusses the recommendation models, evaluation metrics and datasets in detail. The implementation of each recommendation system has been discussed under the models subsection in the experiment section, the evaluation metrics and the methodology of calculations of these metrics have been discussed under evaluation metrics and methods subsection under the experiment section.

The results and discussion section contains observed results from the study in form of graphs and tables and discusses the conclusions derived from these results in details.

The future work section provides suggestions for future studies based on the results and conclusions derived from this study.

2. Literature Review:

[1.] “Improving Recommendation Lists Through Topic Diversification” by Ziegler et. al.
The study is based on the premise that the use of accuracy as a sole metric for evaluation of performances of recommendation systems is not very effective and masks some inherent flaws of recommendation systems such as recommendations not being completely based on user interests. This research focuses on the concept of diversification in evaluating recommendation systems using metrics like novelty and coverage in addition to the more popular accuracy metric, the paper also presents intra-list similarity as a new metric for evaluation for recommendation systems. After using the proposed metrics to assess user-based as well as item-based recommendation systems it was concluded that though the results of diversification were not very effective on user-based systems, the performance of item-based systems could be improved significantly, and suggested finding the right trade-off between accuracy and diversification.

[2] “ Rank and Relevance in Novelty and Diversity Metrics for Recommender Systems” by Saúl Vargas and Pablo Castells
The study builds upon the work done above study and others and aims to establish a clear common methodological and conceptual ground between various metrics for evaluation of recommendation systems. It proposes ‘discovery’, ‘choice’ and ‘relevance’ as three factors in relationship between users and items, and discusses item discovery and diversity.

[3] “Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions” by Adomavicius G. and Tuzhilin A., *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734-749, June 2005.
This paper discusses various state-of-the-art recommendation systems, their developments, discusses limitations of each of these systems and suggests measures to improve their performances.

3.Methodology Adopted:

This study was conducted to compare the performances of three animated show recommendation system models on basis of three metrics- accuracy, scope and similarity. The metrics can be defined as:

i. Accuracy- the degree of precision in recommending content based on item attributes or user interests.
ii. Scope- the percentage of content recommended from entire dataset.
iii. Similarity- the degree of similitude or likeness between various items within the recommended content.

The three recommendation system models used for the study are:

i. An item-based recommendation system based on K-nearest-neighbours regression algorithm.
ii. An item-based collaborative filtering model.
iii. A user-based collaborative filtering model.

Collaborative filtering technique makes recommendations based on similarities between users' interests from the ratings provided by the users to items. Item-based collaborative filtering recommends items to users based on the similarity between these items and the items that user has interacted with or rated in the past. User based collaborative filtering finds similar users based on their mutual likes or dislikes and recommend items to a particular user based on items liked by similar users. All three recommendation systems used in the study are top-n recommendation systems. These models have been discussed in below.

3.1 Data Collection and Dataset Used:

The datasets used for the study have been collected from anime aggregator Myanimelist.net by user CooperUnion and have been made publicly available on Kaggle under CCO: public domain license. Two datasets have been used for the study- an anime dataset and a user ratings dataset. The anime dataset contains data about 12,294 anime shows and movies based on the following attributes:

(a)anime_id- a unique id for each anime.

(b)name- title of the anime.

(c)genre- genre of the anime, an anime can have multiple genres.

(d)type-type of anime such as TV, OVA, special, musical or movie.

(e)rating-average rating out of 10 for the anime

(f)members- number of community members in an anime's group.

(g)episodes-the number of episodes in the anime.

Similarly, the user ratings dataset contains the user ratings provided by over 73000 users for anime in the previous set. It contains the following attributes:

(a) user_id – a unique user id for each user.

(b) anime_id- the anime that the user has rated and,

(c)rating- user rating from 1 to 10 and -1 if user has seen the anime but not assigned the rating.

Due to restrictions in hardware capability here we have used data for 10000 users containing over a million ratings.

A	B	C	D	E	F	G
anime_id	name	genre	type	episodes	rating	members
32281	Kimi no Na wa.	Drama| Romance| School| Supernatural	Movie	1	9.37	200630
5114	Fullmetal Alchemist: Brotherhood	Action| Adventure| Drama| Fantasy| Magic| Military| Shounen	TV	64	9.26	793665
28977	GintamaÂ°	Action| Comedy| Historical| Parody| Samurai| Sci-Fi| Shounen	TV	51	9.25	114262
9253	Steins;Gate	Sci-Fi| Thriller	TV	24	9.17	673572
11061	Hunter x Hunter (2011)	Action| Adventure| Shounen| Super Power	TV	148	9.13	425855
21469	Stand By Me Doraemon	Comedy| Kids| Sci-Fi| Shounen	Movie	1	8.12	5712

Figure 1. A sample of Anime Dataset

A	B	C
user_id	anime_id	rating
5496	32013	8
5496	32093	6
5496	32542	8
5496	32828	-1
5497	16	8
5497	44	7
5497	45	-1
5497	46	9
5497	47	7
5497	53	8
5497	101	7

Figure 2. A sample of User Ratings Dataset

3.2 Models Used:

3.2.1 KNN regression-based recommendation system:

Upon analyzing the anime dataset, it can be assumed that out of all attributes for each anime ‘genre’ and ‘members’ attributes have a significant impact on an anime’s average rating whereas the attributes ‘type’ and ‘episodes’ do not have a direct relation with the average rating of an anime. This assumption can be explained based on the empirical evidence that anime that contained ‘action’ as their major genre had an average rating of 6.7 whereas anime that had ‘kids’ as their primary genre scored only 5.5 on average. The attribute ‘members’ relates to the popularity of an anime, therefore, it can be assumed that anime with more members will tend to have a higher average rating, whereas the attributes ‘type’ do not directly impact the average ratings as both movies and show can have high as well as low ratings, and movies can only have 1 episode despite being rated highly.

KNN regression-based recommendation system model is an item-based, machine learning model that uses k-nearest neighbors regression algorithm to predict the likely rating of an anime. Based on the predicted average rating for an anime, the model produces top-n recommendations having average ratings in the neighborhood of predicted ratings.

The anime dataset contains content from 44 genres. Therefore, implementing a model based on genre as a factor requires 43 dummy variables for genres. This can lead to high dimensionality of data which affects the robustness and accuracy of the model. In order to counter this, we use a method proposed by the BellKor team during the Netflix prize competition [5,6]. It was proposed that only the top-n most frequently occurring genre are used. In this study, top 15 most frequently occurring genres were considered.

The independent variable set X contains ‘members’ attribute and 15 dummy variables for the top-15 genres and the dependent variable set y consisted of the ‘rating’ attribute. Here n variables have been used instead of n-1, as the value of a variable cannot be predicted based on set of n-1 values.

X= {‘members’, ‘Comedy’, ‘Action’, ‘Adventure’, ‘Drama’, ‘Mature’, ‘Fantasy’, ‘Kids’, ‘Music’, ‘Dementia’, ‘Historical’, ‘Mecha’, ‘Slice of life’, ‘Romance’, ‘Demons’, ‘Sci-fi’}

Y={‘ratings’}

After separation of independent and dependent variables, the dataset is split into training and testing sets. A ratio of 80:20 for training and testing set has been used for this study. The resultant variable sets are X_train, X_test, y_train and y_test. Upon the separation of training and testing sets, feature scaling is performed on the independent variable-sets X_train and X_test, so all the values are scaled between -1 to 1. Finally, the regressor is trained on training data and the model is ready. The vector y_pred is used for storing predicted ratings on the test data.

This recommendation system is a top-n recommendation system. The recommendations for an item c, produced by this system, are based on the distance between the predicted rating for c and actual values of ratings for other items in the dataset.

For evaluation of this model- accuracy, scope and similarity metrics have been applied to the model. The results of evaluation are discussed in the results section.

3.2.2 Item-Based Collaborative Filtering Model:

The Item-based collaborative filtering model is memory-based recommendation system model that uses collaborative filtering technique to recommend items from item set C related to an item c_i, such that the top-n items having highest correlation with c_i are recommended. For calculation of correlation between items Pearson's correlation coefficient (ρ) has been used. The formula for PCC for a pair of random variables X and Y is given by

$$\rho X, Y = \frac{Cov(X, Y)}{\rho X \ \rho Y}$$

Where:

Cov (X, Y) = covariance

ρx is standard deviation of X

ρy is standard deviation of Y

For data cleaning, all unused tuples where the user has not watched and rated any content and all columns representing items that have not been rated by any users are removed. This removes most of the runtime errors as correlation on an empty matrix consisting only of nan values cannot be calculated. This model makes use of the anime dataset as well as the user ratings dataset to create a user-item matrix M_{ui}: M_{ui} = [U * C], where U is the set of all the users and C is the set of all the items. This matrix stores the value of the rating provided by a user u, to an item c for all u in U and all c in C.

$M_{ui}[i][j]$ = rating provided by the user u_i to the item c_j

To generate recommendations from item-set C for a particular item, say c_i, a correlation vector Ri is created which stores the correlation of c_i with all the other items in item-set C. This can be written as:

$$Ri[j] = \rho\ c_i, c_j, \qquad \forall c_j \in C$$

After generating the correlation vector R, the top n items having highest correlation with item c_i can be produced. This may lead to the system recommending some content that has high correlation with item c_i, but is not rated by many users. This may affect the accuracy of the system. Therefore, a final filter is applied where the system filters out the content which has not been rated by more than n users. In this study this n is taken as 100.

This model is evaluated on the basis of accuracy, scope and similarity. The results of these evaluations have been discussed under the results section.

3.2.3 User-Based Collaborative Filtering Model:

Similar to the previous model, the user-based collaborative filtering model is a memory-based recommendation system and is more computationally intensive than the K-NN regressor based model. The model uses collaborative filtering technique to find the users in user-set U, having similar interests as a particular user u_i. Again, similar to the previous model Pearson's Correlation Coefficient is used for calculating the correlation. The formula for PCC is discussed in the previous system.

In data pre-processing, all unused tuples for items that have not been rated by any user and all columns representing users that have not rated any movie are removed. This reduces dimensions of user-item matrix to be formed and provides better accuracy.

Like the item-based recommendation model, this model also uses both the anime and user ratings datasets. This model creates an item-user matrix M_{iu}: M_{iu} = [C * U] where C is the set of all the items and U is the set of all the users. This matrix stores the values of the ratings provided to an item c by a user u for all u in U and for all c in C.

$$M_{iu}\,[i][j] = rating\ r\ provided\ to\ the\ item\ c_i\ by\ user\ u_j$$

To generate recommendations from user set U for a particular user u_i, a correlation vector $R_u i$ is created which stores the correlation of user u_i with all other users in user-set U. This is denoted as:

$$R_u i[j] = \rho\, u_i, u_j \,\forall u_j \,\in U$$

Upon generating the correlation vector R_u, a set U_{sim} of top-n users with highest correlation coefficient values when correlated with user u_i can be obtained. Merging this set with user ratings dataset on 'user_id' the dataset 'U_{sim}-item dataset' of all items rated by the top n similar users (denoted by u_{sim}) is obtained. A question that arises now is in regards to the process that should be used to recommend top-m content based on interests of top-n similar users. The solution proposed here is using a recommendation score metric. The recommendation score for each tuple in U_{sim}-item dataset is given by multiplication of rating provided by user u_{sim} to an item c_i and the correlation between the users u_i and u_{sim}. This metric is not an indicator of the quality of an item c_i, but provides a ranking order for top-m items. An item c_{high} having a high value of this metric indicates that it has been highly rated by a user $u_{sim\text{-}high}$ who is also highly correlated with user u_i. After this the average recommendation score for each item in U_{sim}-Item dataset is calculated, as some items may have been ranked by multiple users u_{sim}. The number of times an item appears in U_{sim}-item dataset is also counted; this is another important metric that indicates the number of similar users who have watched the item/content. An item with high frequency in U_{sim}-item dataset indicates that high number of people have watched the content and therefore, if its recommendation score is high, it is likely to be recommended.

To recommend a list of top-m items based on top-n users, the top m*n contents having highest frequencies in U_{sim}-item dataset were filtered out first. From this list, another list containing top-m items with highest recommendation scores is produced.

$$Recommendation\ Score\left(c_{i,} u_{sim} j\right) = Rating\left(c_{i,} u_{sim} j\right) * \rho\, u_i, u_{sim} j$$

This recommendation system is evaluated on basis of accuracy, scope and similarity. The results of these evaluations have been discussed under the results section.

3.3 Evaluation Metrics Used:

Like any other study, evaluation of the models helps not only in determining the consistency, robustness and precision of the models, but also the inaccuracies and scope for future development. Therefore, it becomes necessary that the models are evaluated using multiple and meaningful metrics. Accuracy is the most widely used evaluation metric on recommendation systems but it presents an incomplete picture upon which the models should be evaluated. For example, accuracy does not take into account the diversity or similarity of items recommended by a system or what fraction of items out of all the items are ever recommended. This may lead to an inefficiency where the recommendations are not up to the user's interests. Therefore, we use two additional metrics- scope and similarity.

(a)Scope- the percentage of shows in a sample set that are recommended by a recommendation system.

(b)Similarity-the degree of similarity between the recommended items.

Since the study uses two item-based recommendation systems and one user-based recommendation system, same sample cannot be used to compare all the models on all metrics. However, in order to maintain consistency in the experiments both item-based recommender systems have been evaluated on same item samples and a different user sample has been used for user-based recommendation system.

Moreover, the K-NN regressor model is a machine learning model, capable of high throughput, whereas, the collaborative filtering models are memory-based models and take some time in making each recommendation. During the data pre-processing phase, a lot of users and items having sparse or empty vectors have been cleaned out to increase efficiency. To avoid unexpected results and runtime errors, evaluation has not been performed on entire dataset in case of both collaborative filtering models. However, results for entire dataset are provided for KNN based model along with its sample results.

3.3.1 Accuracy:

Accuracy measures the precision in recommending an item based on an item attribute or user interests. Accuracy is given by:

$$Accuracy = \left(1 - \frac{MAEP}{100}\right) * 100$$

Where:

MEAP- Mean Absolute Error Percentage

For a set of n operations:

$$MAEP = \frac{1}{n}[\sum_{i=1}^{n} \frac{|(value_{actual-} \ value_{observed})|}{value_{actual}}] * 100$$

In KNN regressor system MAEP is calculated for each item c_i using the mean of ratings of top 5 recommendations as observed value and average rating of c_i as actual value.

In Item-based collaborative filtering model MAEP for each item c_i is calculated using mean rating of recommended contents as observed value and average rating of c_i as actual value. In user based collaborative filtering model MAEP for each user is calculated using mean recommendation score for each item c_i as observed value and rating of item c_i as actual value. The size of sample set of items used for comparison of item-based recommendation systems is 100. Similarly, the size of sample set of users used for comparison of user-based recommendation systems is 100.

3.3.2 Scope:

Scope metric indicates the ability of a recommendation system to make diverse predictions. Scope is the percentage of items in the scope set that have been recommended at least once while producing recommendations for items in test set. For a top-n recommendation system if the test set size is x, then size of scope set should ideally be n*x. This study takes the value of n as n=5, test set size = 100 and scope set size=500.

3.3.3 Similarity:

Similarity metric indicates likeness between items in the set of recommended items. Having a very high similarity value between the recommended items means the recommended items are alike in nature. For calculation of similarity between recommended items, ‘similarity score’ metric is introduced. Earlier it has been discussed how some attributes in the anime dataset, like ‘type’ and ‘episodes’, do not impact average ratings of items/content. However, these attributes can be used to determine similarity between recommended items. For calculation of similarity scores across anime dataset, another KNN regressor model is created. This model takes ‘members’, ‘episodes’ and a dummy variable set containing five variables for ‘type’, as its independent variables. Average ratings are taken as dependent variable. After obtaining similarity scores calculations are performed as follows:

Assume a sample set S of m users/items (u_i /c_i). For each user/item a set of recommended items S_{rec} is generated containing top-n recommended items denoted as c_{rec}. Now, similarity scores for each recommended item are obtained by the intersection, S_{rec} ∩ anime dataset on ‘user id. Now mean value of similarity in S_{rec}, denoted by μ_{rec}, is calculated. Absolute deviation from mean μ_{rec} for each item c_{rec} is calculated in recommended set S_{rec} as $| \mu_{rec} - c_{rec}|$, for all c_{rec} in S_{rec}. Now percentage mean deviation is calculated for each item as:

$$percent\ mean\ deviation = \left(\frac{| \mu_{\mathrm{rec}} - c_{rec}|}{\mu_{\mathrm{rec}}}\right) * 100$$

Average of percent mean deviation for all items in a sample set is calculated. Similarity% is given as:

$$Similarity\ \% = 100 - average\ percentage\ mean\ deviation$$

Now average similarity% is calculated for Set S.

4. Experiment:

In this section major steps in the implementation of each recommendation system have been discussed in detail, along with relevant figures and tables. The models have been implemented in python, using spyder IDE. This section uses various python modules such as NumPy. Pandas, Matplotlib, Scikit-learn and Seaborn. The section also shows the methods used for collection of results to evaluate each Recommendation System.

4.1 Implementation of Models:

4.1.1. The KNN Regressor based recommendation system:

The first step for implementing KNN Regressor based system is to import the relevant libraries. NumPy, Pandas and Matplotlib libraries have been used in this model. After implementing the libraries, the anime dataset is imported as anime_df. Now pre-processing and cleaning of data is performed using the fillna() and dropna() methods of pandas module. In the data cleaning step, we remove the anime where average ratings, genre are not available, as these attributes are crucial in this model.

anime_df - DataFrame

Index	anime_id	name	genre	type	episodes	rating	members
10886	32557	Zombie Clay Animation: I'm Stuck!!	Comedy| Horror	ONA	4	4.33	72
10887	30089	Zombie Clay Animation: Life of the Dead	Comedy| Horror	OVA	4	4.95	125
10888	30090	Zombie Ehon	Comedy	ONA	1	3.54	86
10889	13167	Zoobles!	Kids	TV	26	5.57	109
10890	11097	Zou no Inai Doubutsuen	Drama	Movie	1	6.07	85
10891	11095	Zouressha ga Yatte Kita	Adventure	Movie	1	6.06	78
10892	7808	Zukkoke Knight: Don De La Mancha	Adventure| Comedy| Historical| Romance	TV	23	6.47	172
10893	28543	Zukkoke Sannin-gumi no Hi Asobi Boushi Daisakusen	Drama| Kids	OVA	1	5.83	50
10894	18967	Zukkoke Sannin-gumi: Zukkoke Jikuu Bouken	Comedy| Historical| Sci-Fi	OVA	1	6.13	76
10895	13455	Zumomo to Nupepe	Comedy	TV	32	7.00	120
10896	34096	Gintama (2017)	Action| Comedy| Historical| Parody| Samurai| Sci-Fi| Shounen	TV	Unknown	nan	13383
10897	34134	One Punch Man 2	Action| Comedy| Parody| Sci-Fi| Seinen| Super Power| Supernatural	TV	Unknown	nan	90706
10898	30484	Steins;Gate 0	Sci-Fi| Thriller	nan	Unknown	nan	60999
10899	25777	Shingeki no Kyojin Season 2	Action| Drama| Fantasy| Shounen| Super Power	TV	Unknown	nan	170054
10900	34437	Code Geass: Fukkatsu no Lelouch	Action| Drama| Mecha| Military| Sci-Fi| Super Power	nan	Unknown	nan	22748
10901	33486	Boku no Hero Academia 2nd Season	Action| Comedy| School| Shounen| Super Power	TV	Unknown	nan	46892

Figure-3: Figure showing the dataset anime_df before data cleaning anime_df - DataFrame

Index	anime_id	name	genre	type	episodes	rating	members
10891	11095	Zouressha ga Yatte Kita	Adventure	Movie	1	6.06	78
10892	7808	Zukkoke Knight: Don De La Mancha	Adventure| Comedy| Historical| Romance	TV	23	6.47	172
10893	28543	Zukkoke Sannin-gumi no Hi Asobi Boushi Daisakusen	Drama| Kids	OVA	1	5.83	50
10894	18967	Zukkoke Sannin-gumi: Zukkoke Jikuu Bouken	Comedy| Historical| Sci-Fi	OVA	1	6.13	76
10895	13455	Zumomo to Nupepe	Comedy	TV	32	7.00	120
11114	11879	Oni Chichi: Re-born	mature	OVA	2	7.89	14342
11115	29575	Mankitsu Happening	mature	OVA	4	7.83	8510
11116	15843	Koiito Kinenbi The Animation	mature	OVA	2	7.75	6940
11117	21097	Oni Chichi: Rebuild	mature	OVA	3	7.75	9825
11118	2238	Fuyu no Semi	Drama| Historical| Romance| Samurai| Yaoi	OVA	3	7.74	12270
11119	10779	Eroge! H mo Game mo Kaihatsu Zanmai	mature	OVA	6	7.68	20316
11120	10380	Oni Chichi: Re-birth	mature	OVA	1	7.65	14925
11121	25345	Rance 01: Hikari wo Motomete The Animation	Fantasy| mature| Magic	OVA	4	7.61	6158
11122	22069	Swing Out Sisters (2014)	mature	OVA	1	7.61	5099
11123	8634	Koisuru Boukun	Comedy| Romance| Yaoi	OVA	2	7.59	31195

Figure-4 showing the dataset anime_df after cleaning

After data cleaning re-indexing is performed using the reset index() method. One of the attributes that has a direct impact on average ratings is “genre”. The anime dataset contains content from 44 genres. Therefore, implementing a model based on genre as a factor requires 44 dummy variables for genres(usually n-1 dummy variables are taken, but in this case the last attribute is not dependent on other attributes as multiple genres can exist for one item). This can lead to high dimensionality of data which affects the robustness and accuracy of the model. In order to counter this, we use a method proposed by the Bellor team during the Netflix prize competition [5,6]. It was proposed that only the top-n most frequently occurring genre are used. In this study, top 15 most frequently occurring genres were considered.

top_15_genre_list - Index

Index	0
0	Comedy
1	Action
2	Adventure
3	Drama
4	mature
5	Fantasy
6	Kids
7	Music
8	Dementia
9	Historical
10	Mecha
11	Slice of Life
12	Romance
13	Demons
14	Sci-Fi

Figure-5:Figure showing the top-15 most frequent genres.

The next step consists of populating these dummy variables.

The independent variable set X contains ‘members’ attribute and 15 dummy variables for the top-15 genres and the dependent variable set y consisted of the ‘rating’ attribute. Here n variables have been used instead of n-1, as the value of a variable cannot be predicted based on set of n-1 values.

X= {‘members’, ‘Comedy’, ‘Action’, ‘Adventure’, ‘Drama’, ‘Mature’, ‘Fantasy’, ‘Kids’, ‘Music’, ‘Dementia’, ‘Historical’, ‘Mecha’, ‘Slice of life’, ‘Romance’, ‘Demons’, ‘Sci-fi’}

Y={‘ratings’}

X - NumPy object array

	0	1	2	3	4	5	6	7	8	9	10	11
0	200630	0	0	0	1	0	0	0	0	0	0	0
1	793665	0	1	1	1	0	1	0	0	0	0	0
2	114262	1	1	0	0	0	0	0	0	0	1	0
3	673572	0	0	0	0	0	0	0	0	0	0	0
4	151266	1	1	0	0	0	0	0	0	0	1	0
5	93351	1	0	0	1	0	0	0	0	0	0	0
6	425855	0	1	1	0	0	0	0	0	0	0	0
7	80679	0	0	0	1	0	0	0	0	0	0	0
8	72534	1	1	0	0	0	0	0	0	0	1	0
9	81109	1	1	0	0	0	0	0	0	0	1	0
10	456749	0	0	0	1	0	1	0	0	0	0	0
11	102733	0	0	0	1	0	0	0	0	0	0	0
12	336376	1	1	0	0	0	0	0	0	0	1	0
13	572888	0	1	0	1	0	0	0	0	0	0	1
14	179342	1	0	0	1	0	0	0	0	0	0	0

Figure-6:Figure showing the independent variable set X.

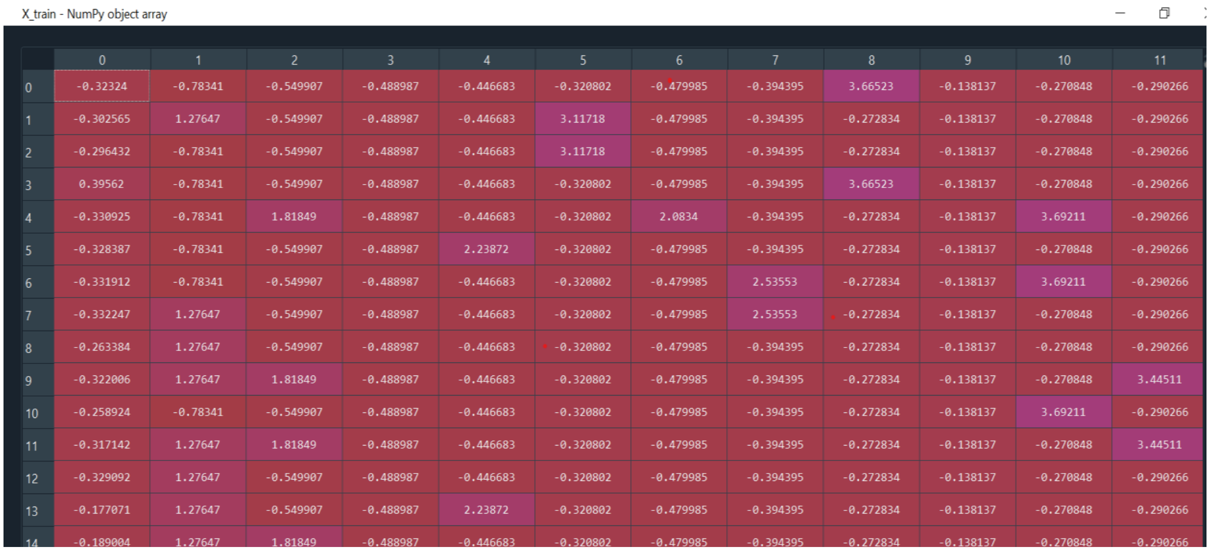
y - NumPy object array

	0
0	9.37
1	9.26
2	9.25
3	9.17
4	9.16
5	9.15
6	9.13
7	9.11

Figure-7:figure showing the dependent variable set y

After this we split the data set into training and testing sets using train_test_split() method of scikit-learn's model-selection library.

A ratio of 80:20 for training and testing set has been used for this study. The resultant variable sets are X_train, X_test, y_train and y_test. Upon the separation of training and testing sets, feature scaling is performed on the independent variable-sets X_train and X_test, so all the values are scaled between -1 to 1.

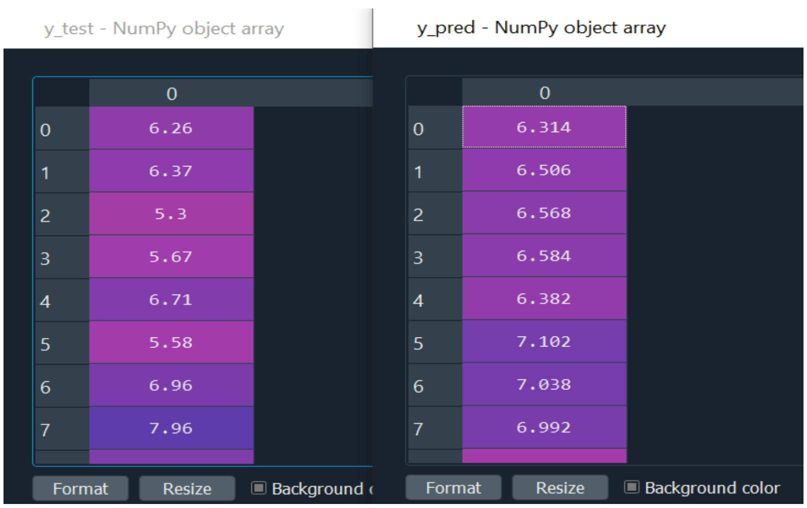


X_train - NumPy object array

	0	1	2	3	4	5	6	7	8	9	10	11
0	-0.32324	-0.78341	-0.549907	-0.488987	-0.446683	-0.320802	-0.479985	-0.394395	3.66523	-0.138137	-0.270848	-0.290266
1	-0.302565	1.27647	-0.549907	-0.488987	-0.446683	3.11718	-0.479985	-0.394395	-0.272834	-0.138137	-0.270848	-0.290266
2	-0.296432	-0.78341	-0.549907	-0.488987	-0.446683	3.11718	-0.479985	-0.394395	-0.272834	-0.138137	-0.270848	-0.290266
3	0.39562	-0.78341	-0.549907	-0.488987	-0.446683	-0.320802	-0.479985	-0.394395	3.66523	-0.138137	-0.270848	-0.290266
4	-0.330925	-0.78341	1.81849	-0.488987	-0.446683	-0.320802	2.0834	-0.394395	-0.272834	-0.138137	3.69211	-0.290266
5	-0.328387	-0.78341	-0.549907	-0.488987	2.23872	-0.320802	-0.479985	-0.394395	-0.272834	-0.138137	-0.270848	-0.290266
6	-0.331912	-0.78341	-0.549907	-0.488987	-0.446683	-0.320802	-0.479985	2.53553	-0.272834	-0.138137	3.69211	-0.290266
7	-0.332247	1.27647	-0.549907	-0.488987	-0.446683	-0.320802	-0.479985	2.53553	-0.272834	-0.138137	-0.270848	-0.290266
8	-0.263384	1.27647	-0.549907	-0.488987	-0.446683	-0.320802	-0.479985	-0.394395	-0.272834	-0.138137	-0.270848	-0.290266
9	-0.322006	1.27647	1.81849	-0.488987	-0.446683	-0.320802	-0.479985	-0.394395	-0.272834	-0.138137	-0.270848	3.44511
10	-0.258924	-0.78341	-0.549907	-0.488987	-0.446683	-0.320802	-0.479985	-0.394395	-0.272834	-0.138137	3.69211	-0.290266
11	-0.317142	1.27647	1.81849	-0.488987	-0.446683	-0.320802	-0.479985	-0.394395	-0.272834	-0.138137	-0.270848	3.44511
12	-0.329092	1.27647	-0.549907	-0.488987	-0.446683	-0.320802	-0.479985	-0.394395	-0.272834	-0.138137	-0.270848	-0.290266
13	-0.177071	1.27647	-0.549907	-0.488987	2.23872	-0.320802	-0.479985	-0.394395	-0.272834	-0.138137	-0.270848	-0.290266
14	-0.189004	1.27647	1.81849	-0.488987	-0.446683	-0.320802	-0.479985	-0.394395	-0.272834	-0.138137	-0.270848	-0.290266

Figure-8: Figure showing the X_train DataFrame after feature scaling

Finally, the KNN regressor model is fit into the training set using KNeighborsRegressor() method of scikit-learn. Here we use algorithm= 'auto', leaf size=30, and metric= 'minkowski'. Now rating values for the test set are predicted using predict() method.



y_test - NumPy object array

	0
0	6.26
1	6.37
2	5.3
3	5.67
4	6.71
5	5.58
6	6.96
7	7.96

y_pred - NumPy object array

	0
0	6.314
1	6.506
2	6.568
3	6.584
4	6.382
5	7.102
6	7.038
7	6.992

Figure-9:Figure showing the predicted and actual values of y.

For recommending content distance attribute is introduced, for each item the set of recommended content will have the least value of distance from the predicted value of the said item.

The getNeighbors() function has been implemented to produce recommended results for any given item. It takes the following attributes:

n= number of required recommendations, item= the items for which recommendations are produced. Matrix- the dataset used for recommendations.

For the anime 'Nano Invaders', recommendations produced are:

9633	27943	Nano Invaders	Action| Adventure| Shounen| Super Power	TV	52	7.08

Value of y_pred[21] =7.35

recommended_items - DataFrame

Index	anime_id	name	genre	type	episodes	rating	members
2133	16417	Tamako Market	Comedy| Slice of Life	TV	12	7.35	128529
2116	11777	Lupin III: Chi no Kokuin - Eien no Mermaid	Action| Adventure| Comedy| Shounen	Special	1	7.35	3069
2115	32245	Kuromukuro	Action| Mecha| Sci-Fi	TV	26	7.35	46323
2114	1516	Kirarin☆Revolution	Comedy| Drama| Romance| Shoujo	TV	153	7.35	14932
2113	3272	Kinnikuman	Adventure| Comedy| Shounen| Sports	TV	137	7.35	3623

Figure 10(a) showing the anime for which recommendations are made, figure10(b) shows set of recommended shows.

Here we can observe that since this recommendation system finds the top-n shows with closest ratings to the predicted rating of anime, all 5 recommendations have the rating of 7.35.

Similarly, we can make more than 5 predictions by increasing the value of n.

For n=10, the same anime gives following recommendations:

recommended_items - DataFrame

Index	anime_id	name	genre	type	episodes	rating
2133	16417	Tamako Market	Comedy| Slice of Life	TV	12	7.35
2116	11777	Lupin III: Chi no Kokuin - Eien no Mermaid	Action| Adventure| Comedy| Shounen	Special	1	7.35
2115	32245	Kuromukuro	Action| Mecha| Sci-Fi	TV	26	7.35
2114	1516	Kirarin☆Revolution	Comedy| Drama| Romance| Shoujo	TV	153	7.35
2113	3272	Kinnikuman	Adventure| Comedy| Shounen| Sports	TV	137	7.35
2112	3245	Kindaichi Shounen no Jikenbo Specials	Mystery| Shounen	Special	2	7.35
2111	3229	Kimi ga Aruji de Shitsuji ga Ore de	Comedy| Ecchi| Harem| Parody| Romance	TV	13	7.35
2117	1967	Mobile Suit Zeta Gundam: A New Translation - Heir to the Stars	Drama| Mecha| Military| Sci-Fi| Space	Movie	1	7.35
2110	2544	Kazoku Robinson Hyouryuuki: Fushigi na Shima no Flone	Adventure| Drama| Historical| Slice of Life	TV	50	7.35
2108	19951	Hunter x Hunter Movie: The Last Mission	Action| Adventure| Shounen| Super Power	Movie	1	7.35

Figure10(c) shows the set of recommended content for n=10.

In the next section we will discuss about the item-based recommendation system.

4.1.2 Item-Based Recommendation System using Collaborative Filtering:

In this section, we will implement our second recommendation system model which is based on item-based collaborative filtering technique. In item-based collaborative filtering, the ratings data for multiple items are combined to find the items that were similarly rated by some users, once we find a set of similar items these items can be recommended to users who have not used both items.

The first step for building a recommendation system is to import the libraries. For this system, NumPy, Pandas and Matplotlib libraries have been used. The next step is to import both the anime and user_ratings datasets as anime_df and ratings_df. Now these two datasets are cleaned to remove any users who have not rated any content or any content which has not been rated yet. Since, this model is memory based, we now create an anime_user_matrix which contains ratings provided by the users to the anime. This is done using the pivot_table() method.

show_pivot - DataFrame

user_id	Stocking i	Stocking v	### Pisto	###frienc	Gaiden: Su	en: Sugoi	/ Sailor Sc	uot;0&qu	nashi yori:	" Ky	Shoujo&	ku Shoujo
1	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
2	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
3	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
4	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
5	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
6	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
7	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
8	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
9	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
10	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
11	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
12	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
13	-1.00	-1.00	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
14	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	8.00
15	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan

Figure 11: the pivot table is a sparse matrix containing all users ratings for all shows, here nan means the show has not been rated by the user and -1 means the show has been seen, but no rating has been provided.

The Item-based collaborative filtering model is memory-based recommendation system model that uses collaborative filtering technique to recommend items from item set C related to an item c_i, such that the top-n items having highest correlation with c_i are recommended. For calculation of correlation between items Pearson's correlation coefficient (ρ) has been used. The formula for PCC for a pair of random variables X and Y is given by

$$\rho X, Y = \frac{Cov(X, Y)}{\rho X \ \rho Y}$$

Where:

Cov (X, Y) = covariance

ρx is standard deviation of X

ρy is standard deviation of Y

In order to produce recommendations of items similar to a given item getSimilarItems() method has been implemented. This method takes the following parameters:

n= number of recommendations to be produced

item- the item for which recommendations have to be made

matrix- the matrix userid_anime_matrix.

For example, if top 10 recommendations for the show "Stiens; Gate" have to be produced, the function call is:

getSimilarItems(10,'Gintama', userid_anime_matrix)

<table>
<tr><td>2</td><td>28977</td><td>Gintama°</td><td>Action| Comedy| Historical| Parody| Samurai| Sci-Fi| Shounen</td><td>TV</td><td>51</td><td>9.25</td><td>114262</td></tr>
</table>

The list of recommended items is:

recommended_item_list1 - DataFrame

name	Correlation	count
Gintama	1.00	4974.00
Ojamajo Doremi Na-i-sho	0.97	135.00
Waga Seishun no Arcadia	0.97	164.00
Elf no Wakaokusama	0.97	109.00
Green Legend Ran	0.97	109.00
Oseam	0.97	106.00
Samurai Spirits: Haten Gouma no Shou	0.96	101.00
Hyakujitsu no Bara: Jinginaki Nikukyuu-hen	0.96	153.00
Ys: Tenkuu no Shinden - Adol Christine no Bouken	0.96	114.00
Puchimas!!: Petit Petit iDOLM@STER	0.95	147.00
Harlock Saga: Nibelung no Yubiwa	0.95	111.00

Figure 12(a) showing the anime for which recommendations are made, figure12(b) shows set of recommended shows.

Similarly, if we require 5 recommendations for the same anime we call:

getSimilarItems(5,'Gintama', userid_anime_matrix)

recommended_item_list1 - DataFrame

name	Correlation	count
Gintama	1.00	4974.00
Ojamajo Doremi Na-i-sho	0.97	135.00
Waga Seishun no Arcadia	0.97	164.00
Elf no Wakaokusama	0.97	109.00
Green Legend Ran	0.97	109.00
Oseam	0.97	106.00

Figure12(c) for n=5

4.1.3 User Based Collaborative Filtering Model:

In this section, we will implement our third recommendation system model which is based on user-based collaborative filtering technique. Similar to the previous model, the user-based collaborative filtering model is a memory-based recommendation system and is more computationally intensive than the K-NN regressor based model. The model uses collaborative filtering technique to find the users in user-set U, having similar interests as a particular user u_i. Again, similar to the previous model Pearson's Correlation Coefficient is used for calculating the correlation. The formula for PCC is discussed in the previous system.

The first step for building a recommendation system is to import the libraries. For this system, NumPy, Pandas and Matplotlib libraries have been used. The next step is to import both the anime and user_ratings datasets as anime_df and ratings_df. Now these two datasets are cleaned to remove any users who have not rated any content or any content which has not been rated yet. Since, this model is memory based, we now create an anime_user_matrix which contains each user's ratings for the anime he/she has rated. This is done using the pivot_table() method.

```
   ...: #merge the dataframes
   ...: anime_rating_df=pd.merge(anime_df,rating_df, on='anime_id')
   ...:
   ...: anime_rating_df.groupby('user_id')['rating_y'].describe()
Out[2]:
         count       mean       std   min   25%   50%    75%   max
user_id
1          4.0  10.000000  0.000000  10.0  10.0  10.0  10.00  10.0
2          1.0  10.000000       NaN  10.0  10.0  10.0  10.00  10.0
3         92.0   7.565217  1.549933   3.0   7.0   7.0   8.25  10.0
4          0.0        NaN       NaN   NaN   NaN   NaN    NaN   NaN
5        459.0   4.355120  2.381293   1.0   2.0   5.0   6.00  10.0
...        ...        ...       ...   ...   ...   ...    ...   ...
10089    176.0   7.357955  1.876850   2.0   7.0   8.0   9.00  10.0
10090     24.0   7.250000  0.737210   6.0   7.0   7.0   8.00   9.0
10091     37.0   8.270270  1.017859   6.0   8.0   8.0   9.00  10.0
10092     79.0   8.493671  1.023736   6.0   8.0   8.0   9.00  10.0
10093    110.0   8.118182  1.098221   6.0   7.0   8.0   9.00  10.0

[10093 rows x 8 columns]
```

Figure13: Description of each user's ratings, containing number of ratings, mean of ratings, percentile ratings etc.

Now average rating provided by each user and the number of ratings provided by each user are stored as ratings_df_mean and ratings_df_count respectively. These dataframes are now combined to form ratings_df_mean_count dataframe.

ratings_df_mean -

user_id	mean
1	10.00
2	10.00
3	7.57
5	4.36
7	7.39
8	8.33
9	8.00
10	9.33
11	7.33

ratings_df_count -

user_id	count
1	4.00
2	1.00
3	92.00
5	459.00
7	343.00
8	12.00
9	1.00
10	3.00
11	110.00

ratings_mean_count_df - Data

user_id	mean	count
1	10.00	4.00
2	10.00	1.00
3	7.57	92.00
5	4.36	459.00
7	7.39	343.00
8	8.33	12.00
9	8.00	1.00
10	9.33	3.00
11	7.33	110.00

Figures14(a) showing ratings_df_mean dataframe, 14(b) showing ratings_df_count dataframe and 14(c) showing ratings_df_mean_count dataframe

In order to produce a list of users similar to a given user getSimilarUsers() method has been implemented. This method takes the following parameters:

n= number of similar users to be found

user- the user for which similar users have to be found

matrix- the matrix userid_anime_matrix.

similar_user_list - DataFrame

user_id	orrelation	count
10072	1.00	589.00
2695	1.00	672.00
6467	1.00	264.00
2417	1.00	346.00
2441	1.00	252.00
2526	1.00	207.00
2555	1.00	251.00
6316	1.00	251.00
6262	1.00	264.00
6167	1.00	374.00

Figure15: Figure showing ten similar users for user id=10

Here it is possible to find many users with perfect correlation as user 10 has only rated three items, so anyone among the 10000 users rating same items as user 10 and giving them the same ratings can achieve perfect correlation.

To generate recommendations from user set U for a particular user u_i, a correlation vector $R_u i$ is created which stores the correlation of user u_i with all other users in user-set U. This is denoted as:

$$R_u i[j] = \rho\, u_i, u_j \,\forall u_j \ \in U$$

Upon generating the correlation vector R_u, a set U_{sim} of top-n users with highest correlation coefficient values when correlated with user u_i can be obtained. Merging this set with user ratings dataset on ‘user_id’ the dataset ‘U_{sim}-item dataset’ of all items rated by the top n similar users (denoted by u_{sim}) is obtained. A question that arises now is in regards to the process that should be used to recommend top-m content based on interests of top-n similar users. The solution proposed here is using a recommendation score metric. The recommendation score for each tuple in U_{sim}-item dataset is given by multiplication of rating provided by user u_{sim} to an item c_i and the correlation between the users u_i and u_{sim}.

The function recommend_Anime_list() is used to generate a list of recommended items for a user. It takes the following parameters:

number_recommended = number of similar users to be found

n= number of anime to be recommended

user- the user for which recommendations have to be produced.

matrix- the matrix userid_anime_matrix.

For user 10 on userid_anime_matrix to get a list of 10 recommended anime using 10 most similar users we call:

recommended_anime_list1=recommendAnimeList(10,10,userid_anime_matrix[10],userid_anime_matrix)

recommended_anime_list1 - DataFram

anime_id	count	mean
5114	10.00	9.50
9253	7.00	9.43
2904	8.00	9.12
11061	6.00	9.00
1575	8.00	9.00
12355	5.00	8.80
18115	7.00	8.71
11771	7.00	8.71
16894	6.00	8.67
19815	8.00	8.62

Format Resize Background color

Figure-16:Recommendation list in User-Based Recommendation System

But this does not show the names of recommended anime, therefore we map the following list with anime_df on anime_id to get the names of the anime:

For this purpose a new function recommendAnime() is created, it takes the following parameters:

number_recommended = number of similar users to be found

n= number of anime to be recommended

user- the user for which recommendations have to be produced.

matrix- the matrix userid_anime_matrix.

Using the recommendAnime() function on the same user with same parameters we get:

recommended_anime_list1 - DataFrame

Index	anime_id	count	mean	name	rating
0	5114	10.00	9.50	Fullmetal Alchemist: Brotherhood	9.26
1	9253	7.00	9.43	Steins;Gate	9.17
2	2904	8.00	9.12	Code Geass: Hangyaku no Lelouch R2	8.98
3	11061	6.00	9.00	Hunter x Hunter (2011)	9.13
4	1575	8.00	9.00	Code Geass: Hangyaku no Lelouch	8.83
5	12355	5.00	8.80	Ookami Kodomo no Ame to Yuki	8.84
6	18115	7.00	8.71	Magi: The Kingdom of Magic	8.50
7	11771	7.00	8.71	Kuroko no Basket	8.46
8	16894	6.00	8.67	Kuroko no Basket 2nd Season	8.58
9	19815	8.00	8.62	No Game No Life	8.47

Figure17:figure showing the list of recommended anime for user with user id=10, based on its top 10 most similar users.

Similarly if we want to recommend top-15 shows to user with user_id=100, based on top 10 neighbours we call the function:

recommendAnimeList(10,15,userid_anime_matrix[100],userid_anime_matrix) recommended_anime_list1 - DataFrame

Index	anime_id	count	mean	name	rating
0	4181	12.00	9.33	Clannad: After Story	9.06
1	9253	9.00	9.33	Steins;Gate	9.17
2	1535	13.00	9.31	Death Note	8.71
3	2904	11.00	9.18	Code Geass: Hangyaku no Lelouch R2	8.98
4	1	11.00	9.09	Cowboy Bebop	8.82
5	1575	13.00	9.08	Code Geass: Hangyaku no Lelouch	8.83
6	16498	12.00	9.00	Shingeki no Kyojin	8.54
7	245	9.00	9.00	Great Teacher Onizuka	8.77
8	7311	9.00	8.89	Suzumiya Haruhi no Shoushitsu	8.81
9	2001	12.00	8.83	Tengen Toppa Gurren Lagann	8.78
10	2236	10.00	8.80	Toki wo Kakeru Shoujo	8.44
11	31043	9.00	8.78	Boku dake ga Inai Machi	8.65
12	10162	11.00	8.73	Usagi Drop	8.56
13	4477	9.00	8.67	Nodame Cantabile: Paris-hen	8.27
14	1698	9.00	8.67	Nodame Cantabile	8.46

Figure 18: figure showing the list of 15 recommended anime for user with user id=100, based on its top 10 most similar users.

Similarly, to increase the number of similar users to a number we have to change the value of number_recommended to that number:

Example: To get the top-10 recommendations for user with user_id=100, based on his/her 10 most similar users, we call the following function:

recommendAnimeList(15,10,userid_anime_matrix[100],userid_anime_matrix) recommended_anime_list1 - DataFrame

Index	anime_id	count	mean	name	rating
0	4181	8.00	9.38	Clannad: After Story	9.06
1	2904	7.00	9.14	Code Geass: Hangyaku no Lelouch R2	8.98
2	9253	7.00	9.14	Steins;Gate	9.17
3	1	8.00	9.12	Cowboy Bebop	8.82
4	1535	9.00	9.11	Death Note	8.71
5	245	7.00	9.00	Great Teacher Onizuka	8.77
6	2236	7.00	8.86	Toki wo Kakeru Shoujo	8.44
7	1575	8.00	8.75	Code Geass: Hangyaku no Lelouch	8.83
8	10162	8.00	8.75	Usagi Drop	8.56
9	16498	7.00	8.71	Shingeki no Kyojin	8.54

Figure 19: Figure showing top-10 recommendations for user with user_id=100, based on his/her 10 most similar users, we call the following function:

The next section discusses the evaluation of these recommendation systems based on accuracy, scope and similarity.

4.2 Evaluation Metrics Used:

4.2.1 Accuracy:

Accuracy measures the precision in recommending an item based on an item attribute or user interests. Accuracy is given by:

$$Accuracy = \left(1 - \frac{MAEP}{100}\right) * 100$$

For the KNN regressor system MAEP is calculated for each item c_i using the mean of ratings of top 5 recommendations as observed value and average rating of c_i as actual value.

In Item-based collaborative filtering model MAEP for each item c_i is calculated using mean rating of recommended contents as observed value and average rating of c_i as actual value. In user based collaborative filtering model MAEP for each user is calculated using mean recommendation score for each item c_i as observed value and rating of item c_i as actual value. The size of sample set of items used for comparison of item-based recommendation systems is 100. Similarly, the size of sample set of users used for comparison of user-based recommendation systems is 100.

In case of KNN regressor system we calculate the values of MAE simply by finding the absolute value of difference between y_pred and y_test. After this MAEP is calculated simply by dividing obtained MEA value by original y_test value and multiplying the resultant by 100.

For example, for an anime its predicted value is 6.07 and actual value is 6.35

MAE= |6.35-6.07|= 0.28

MAEP = (MAE/actual_value)*100 -> (0.28/6.35)*100= 4.4%

Similarly, we calculate MAEP for all test items and value of accuracy% is:

Accuracy%=100-MAEP

In case of item-based recommendation system we define the functions meanAbsoluteError() and meanAbsoluteErrorPercentage() to calculate MAE and MAEP respectively. These functions take the following parameters:

n= number of recommendations to be produced

item- the item for which recommendations have to be made

matrix- the matrix userid_anime_matrix.

For example, to calculate MAE for an anime ‘Persona 3 the Movie 1: Spring of birth’

```
In [10]: MAE_example= meanAbsoluteError(5,'Persona 3 the Movie 1: Spring of
Birth',userid_anime_matrix)
    ...: MAEP_example= meanAbsoluteErrorPercentage(5,'Persona 3 the Movie 1:
Spring of Birth',userid_anime_matrix)

In [11]: MAE_example
Out[11]: 0.6633333333333331

In [12]: MAEP_example
Out[12]: 8.47169008088548

In [13]:
```

Figure 20: figure showing sample MAE and MAEP values for item-based recommendation systems.

Here, MAE=0.66 and MAEP=8.47%, therefore

Accuracy%=100-MAEP=91.53%

In case of user-based recommendation system, we define the functions meanAbsoluteError() and meanAbsoluteErrorPercentage() to calculate MAE and MAEP respectively. These functions take the following parameters:

number_recommended = number of similar users to be found

n= number of anime to be recommended

user- the user for which recommendations have to be produced.

matrix- the matrix userid_anime_matrix.

For example, if we want to calculate MEA and MEAP for a user with user_id=85

We call the functions as:

meanAbsoluteError(10,10,userid_anime_matrix[85],userid_anime_matrix)

meanAbsoluteErrorPercentage(10,10,userid_anime_matrix[85],userid_anime_matrix)

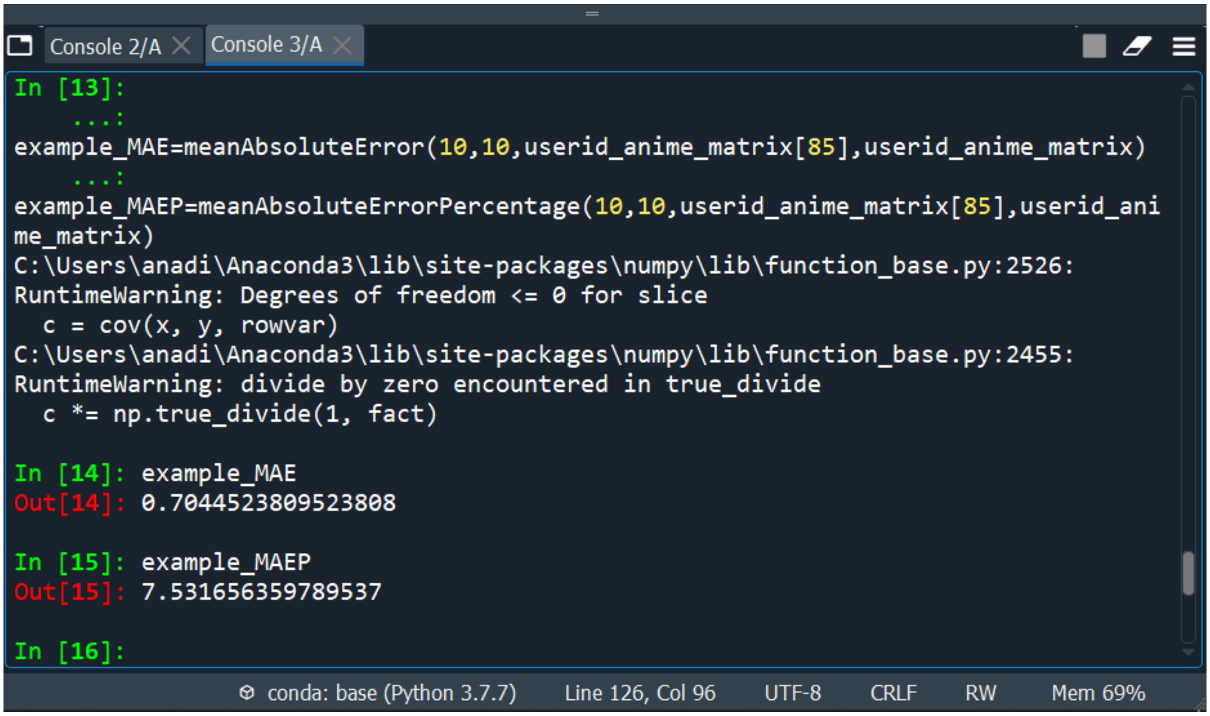


Figure 21: figure showing sample MAE and MAEP values for user-based recommendation systems.

Here MAE=0.704 and MAEP=7.53%

Therefore accuracy% =100-7.53=92.47%

4.2.2 Scope:

Scope metric indicates the ability of a recommendation system to make diverse predictions. Scope is the percentage of items in the scope set that have been recommended at least once while producing recommendations for items in test set. For a top-n recommendation system if the test set size is x, then size of scope set should ideally be n*x. This study takes the value of n as n=5, test set size = 100 and scope set size=500.

No specific functions were defined for calculation of scope but it was simply calculated by taking all the recommendations from experiment set and matching them from sample set, then percentages of matching items were calculated from total number of items in the matching set.

4.2.3 Similarity:

Similarity metric indicates likeness between items in the set of recommended items. Having a very high similarity value between the recommended items means the recommended items are alike in nature. For calculation of similarity we use a new metric similarity using a different KNN regressor model that takes 'episode', 'members' and 'type' as the set of independent variables and 'rating' as the dependent variable.

Assume a sample set S of m users/items (u_i /c_i). For each user/item a set of recommended items S_{rec} is generated containing top-n recommended items denoted as c_{rec}. Now, similarity scores for each recommended item are obtained by the intersection, $S_{rec} \cap$ anime dataset on 'user id. Now mean value of similarity in S_{rec}, denoted by μ_{rec}, is calculated. Absolute deviation from mean μ_{rec} for each item c_{rec} is calculated in recommended set S_{rec} as $| \mu_{rec} - c_{rec}|$, for all c_{rec} in S_{rec}. Now percentage mean deviation is calculated for each item as:

$$percent\ mean\ deviation = \left(\frac{| \mu_{\mathrm{rec}} - c_{rec}|}{\mu_{\mathrm{rec}}}\right) * 100$$

$$Similarity\ \% = 100 - average\ percentage\ mean\ deviation$$

To calculate similarity, we first define a similarity dataset similarity_df, this is just the anime_df dataset with an added predicted similarity column.

similarity_df1 - DataFrame

Index	index	anime_id	name	genre	type	episodes	rating	members	0
0	0	32281	Kimi no Na wa.	Drama| Romance| School| Supernatural	Movie	1	9.37	200630	6.15
1	1	5114	Fullmetal Alchemist: Brotherhood	Action| Adventure| Drama| Fantasy| Magic| Military| Shounen	TV	64	9.26	793665	7.74
2	2	28977	Gintama°	Action| Comedy| Historical| Parody| Samurai| Sci-Fi| Shounen	TV	51	9.25	114262	5.90
3	3	9253	Steins;Gate	Sci-Fi| Thriller	TV	24	9.17	673572	5.38
4	4	9969	Gintama'	Action| Comedy| Historical| Parody| Samurai| Sci-Fi| Shounen	TV	51	9.16	151266	4.57
5	5	32935	Haikyuu!!: Karasuno Koukou VS Shiratorizawa Gakuen Koukou	Comedy| Drama| School| Shounen| Sports	TV	10	9.15	93351	6.80
6	6	11061	Hunter x Hunter (2011)	Action| Adventure| Shounen| Super Power	TV	148	9.13	425855	7.57
7	7	820	Ginga Eiyuu Densetsu	Drama| Military| Sci-Fi| Space	OVA	110	9.11	80679	6.67
8	8	15335	Gintama Movie: Kanketsu-hen - Yorozuya yo Eien Nare	Action| Comedy| Historical| Parody| Samurai| Sci-Fi| Shounen	Movie	1	9.10	72534	7.38
9	9	15417	Gintama': Enchousen	Action| Comedy| Historical| Parody| Samurai| Sci-Fi| Shounen	TV	13	9.11	81109	7.57
10	10	4181	Clannad: After Story	Drama| Fantasy| Romance| Slice of Life| Supernatural	TV	24	9.06	456749	7.31
11	11	28851	Koe no Katachi	Drama| School| Shounen	Movie	1	9.05	102733	6.43
12	12	918	Gintama	Action| Comedy| Historical| Parody| Samurai| Sci-Fi| Shounen	TV	201	9.04	336376	6.96
13	13	2904	Code Geass: Hangyaku no Lelouch R2	Action| Drama| Mecha| Military| Sci-Fi| Super Power	TV	25	8.98	572888	6.79
14	14	28891	Haikyuu!! Second Season	Comedy| Drama| School| Shounen| Sports	TV	25	8.93	179342	5.71
15	15	199	Sen to Chihiro no Kamikakushi	Adventure| Drama| Supernatural	Movie	1	8.93	466254	6.44
16	16	23273	Shigatsu wa Kimi no Uso	Drama| Music| Romance| School| Shounen	TV	22	8.92	416397	5.77
17	17	24701	Mushishi Zoku Shou 2nd Season	Adventure| Fantasy| Historical| Mystery| Seinen| Slice of Life| Supernatural	TV	10	8.88	75894	5.67
18	18	12355	Ookami Kodomo no Ame to Yuki	Fantasy| Slice of Life	Movie	1	8.84	226193	5.92
19	19	1575	Code Geass: Hangyaku no Lelouch	Action| Mecha| Military| School| Sci-Fi| Super Power	TV	25	8.83	715151	7.07
20	20	263	Hajime no Ippo	Comedy| Drama| Shounen| Sports	TV	75	8.83	157670	6.96
			Rurouni Kenshin: Meiji Kenkaku Romantan -	Action| Drama| Historical| Martial Arts|					

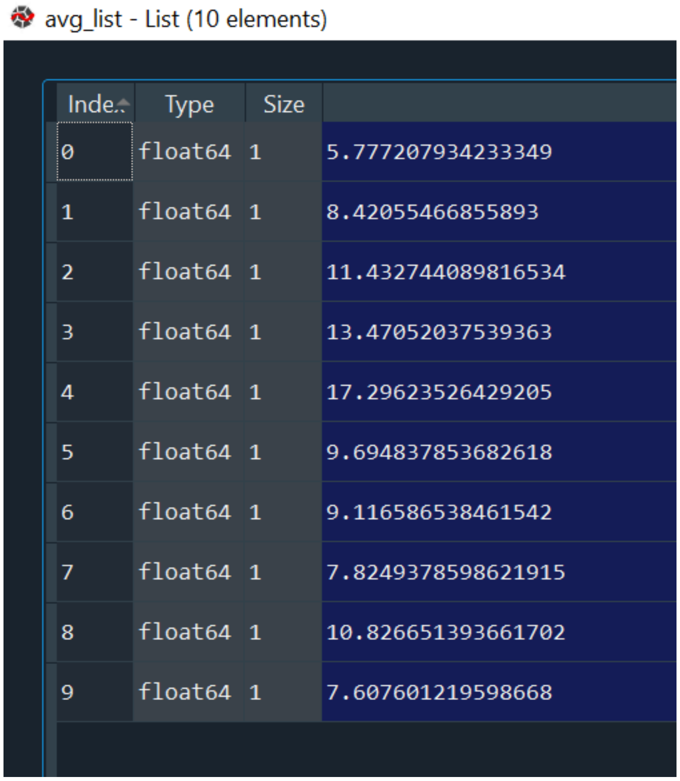
Format Resize Background color Column min/max Save and Close Close

Figure22: figure showing similarity dataset. Here similarity column is labelled as 0.

In each case of calculation of similarity, we maintain an average list to find the average mean deviation of each recommended item from the mean similarity in the recommendation set. Now we take the average of this list to give us average of average mean deviation. This value is then subtracted from 100 to give us the similarity percentage:

Merged_set in each case is the dataframe formed by merging the recommendation list with similarity_df to get similarity values for all items in the recommended list.

For example, in case of KNN system if we take values between 100 to 109 in a loop to calculate similarity, the average list contains 10 values of average mean deviation in similarity metric.



avg_list - List (10 elements)

Index	Type	Size	
0	float64	1	5.777207934233349
1	float64	1	8.42055466855893
2	float64	1	11.432744089816534
3	float64	1	13.47052037539363
4	float64	1	17.29623526429205
5	float64	1	9.694837853682618
6	float64	1	9.116586538461542
7	float64	1	7.8249378598621915
8	float64	1	10.826651393661702
9	float64	1	7.607601219598668

Figure23: List of average mean deviation in similarity metric

Taking the average of values in this list we get:

Percentage mean deviation= 7.61%

Therefore Similarity= 100- Percentage mean deviation= 92.39%

Similarly, in case of item-based recommendation system using collaborative filtering for a list of 10 anime in anime_df defined by aid_list(figure a) the average list is given by(figure b):

Index	Type	Size	
0	str	1	Diamond no Ace: Second Season
1	str	1	Magi: The Kingdom of Magic
2	str	1	Mahou Shoujo Madoka★Magica Movie 3: Hangyaku no Monogatari
3	str	1	Major: World Series
4	str	1	Samurai Champloo
5	str	1	Shokugeki no Souma: Ni no Sara
6	str	1	Katanagatari
7	str	1	Mahou Shoujo Madoka★Magica Movie 2: Eien no Monogatari
8	str	1	Major S6
9	str	1	Mononoke

avg_list - List (10 elements)

Index	Type	Size	
0	float64	1	5.144593461860851
1	float64	1	2.7959661616402975
2	float64	1	4.990954046556504
3	float64	1	12.063748234819448
4	float64	1	8.464281323005043
5	float64	1	7.584107489634178
6	float64	1	10.970788491104765
7	float64	1	5.924466338259442
8	float64	1	13.590206276757598
9	float64	1	11.328878990348922

Figure 24(a) aid_list and 24(b) avg_list for item-based collaborative filtering system

Taking the average of values in this list we get:

Percentage mean deviation= 11.33%

Therefore Similarity= 100- Percentage mean deviation= 88.67%

in case of user-based recommendation system using collaborative filtering for a list of 5 users with user_ids from 17 to 21 the average list is given by:

avg_list - List (5 elements)

Index	Type	Size	Value
0	float64	1	10.091856617078527
1	float64	1	11.67102825722603
2	float64	1	8.521415747627596
3	float64	1	9.64429303624589
4	float64	1	10.040813743790935

Figure 25: avg_list for user-based system

Taking the average of values in this list we get:

Percentage mean deviation= 10.09%

Therefore Similarity= 100- Percentage mean deviation= 89.91%

This concludes our experiment section, results and discussions for these experiments are given in the next section.

5.Results and Discussion:

Accuracy:

Accuracy for item-based recommendation systems were taken using a common continuous sample consisting of 100 items and calculating MAEP for each item. The observed results were as follows: - (a)The value of MAEP for KNN regressor model was found to be 5.37% and its accuracy was found to be 94.63%. (b.) The value of MAEP for item-based collaborative filtering model was found to be 5.03% and hence its accuracy was calculated as 94.97%. For User-based collaborative filtering model a sample of 100 users was taken for calculation of MAEP and accuracy. The results observed were: - (a)The value of MAEP for user-based collaborative filtering model was found to be 8.53%, hence its accuracy was calculated as 91.47%.

Serial No.	Model Name	MAEP	Accuracy
1.	KNN regressor based model	5.37%	94.63%
2.	Item-based collaborative filtering model	5.03%	94.97%
3.	User-based collaborative filtering model	8.53%	91.47%

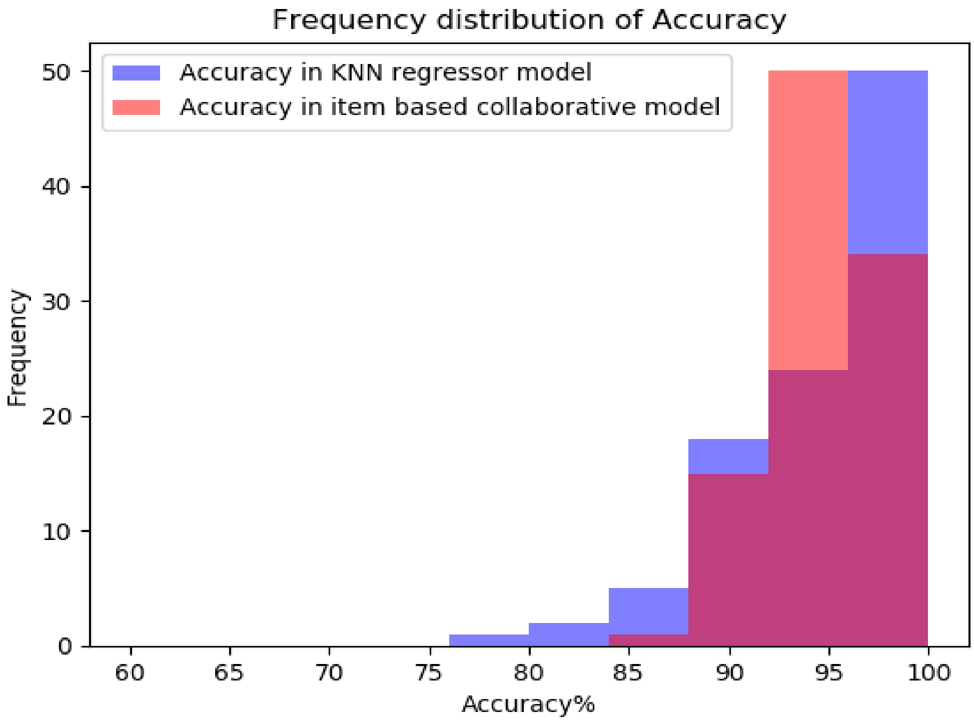
TABLE 3- Table showing the values of MAEP and Accuracy for each model.

Serial No.	KNN regression model		Item-Based Collaborative Filtering Model	
	MAEP	Accuracy	MAEP	Accuracy
802	6.41%	93.59%	8.47%	91.53%
803	2.17%	97.83%	0.87%	99.13%
804	3.04%	96.96%	5.13%	94.87%
805	2.32%	97.68%	6.9%	93.1%
806	6.49%	93.51%	6.53%	93.47%

TABLE-4 Table comparing the values of MAEP and Accuracy between both item-based recommendation system models. Only 5 comparisons are shown here.

User Id	User-Based Collaborative Filtering Model	
	MAEP	Accuracy
140	4.92%	95.08%
141	7.61%	92.39%
142	5.24%	94.76%
143	6.52%	93.48%
144	12.13%	87.87%

TABLE 5- showing the values of MAEP and Accuracy for user-based recommendation system model.


Frequency distribution of Accuracy
Accuracy in KNN regressor model
Accuracy in item based collaborative model
Frequency
Accuracy%
0
10
20
30
40
50
60
65
70
75
80
85
90
95
100

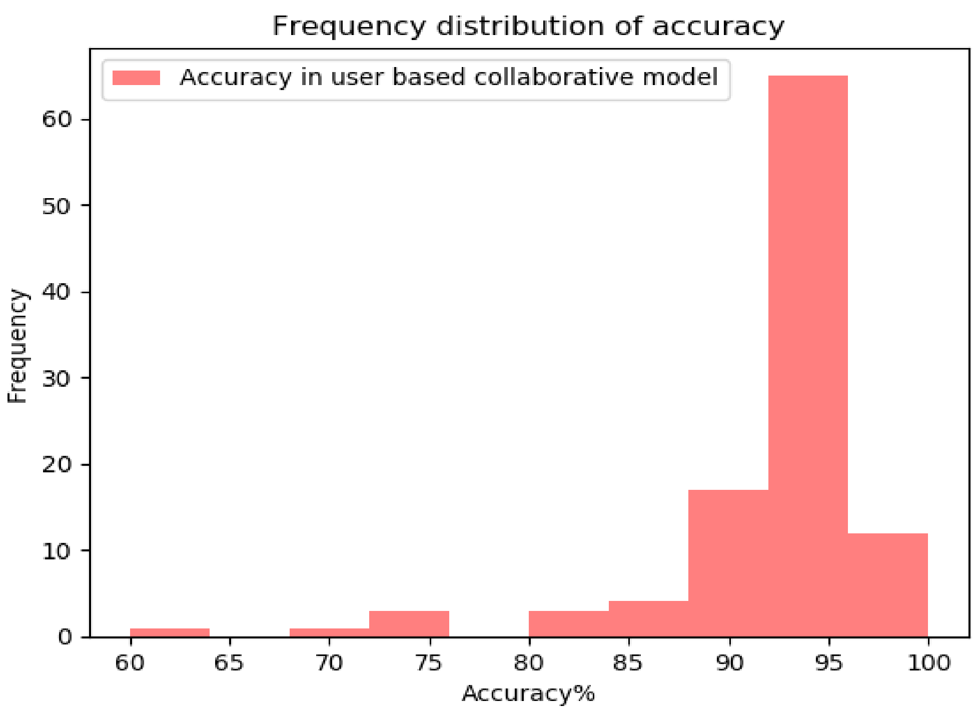

Frequency distribution of accuracy
Accuracy in user based collaborative model
Frequency
Accuracy%
0
10
20
30
40
50
60
60
65
70
75
80
85
90
95
100

Figure 26:Graph (a) showing the comparative frequency distribution of accuracy% for KNN regressor based and Item Based Collaborative filtering system. Figure27:Graph (b) shows frequency distribution of accuracy% for user based collaborative filtering model.

Upon calculating the accuracy for the entire test set in KNN regressor based system, the accuracy was found to be 90.27%.

From the above results following conclusions can be made:

(1) All the recommendation systems performed well on this metric, as accuracy was found to be greater than 90% in all cases.

(2) Out of KNN regressor system and Item-Based collaborative filtering model, the collaborative filtering model performed slightly better, with the difference in accuracy percent being 0.34% between the two of them.

(3) 3 out of 100 samples gave accuracy less than 75% for user based collaborative filtering model, this could be the result of cold-start problem, where it becomes difficult to recommend items to the new user due to the data on the user being sparse. This results in loss of accuracy.

Scope:

Upon using the same scope sample set and experiment set for all three recommendation systems, the following results were observed: (a)54 out of 500 items in scope sample set were recommended by KNN regressor based model. (b)15 out of 500 items in scope sample set were recommended by item based collaborative filtering system. And (c) 7 out of 500 items in scope sample set were recommended by user based collaborative filtering system. From this result it can be concluded that:

(1) The low values of scope can be due to the systems being top-N recommendation system and hence neglecting majority of items that have low average ratings.

Similarity:

For calculation of similarity a sample set of items of size =100 for KNN regressor and Item-based collaborative filtering model is used. Another sample set of 100 users is used for user-based collaborative filtering model. Upon application of similarity score metric and calculation of average similarity percentage, the results observed were: (a)The value of Average similarity percentage for KNN regressor based model was found to be 90.31%, whereas, average similarity percentage for item-based collaborative filtering model was found to be 91.25%. (b) the value average similarity percentage for user-based collaborative filtering model was observed to be 90.01%.

Serial No.	Model Name	Similarity
1.	KNN regressor based model	90.31%
2.	Item-based collaborative filtering model	91.25%
3.	User-based collaborative filtering model	90.01%

TABLE 6- Table showing the values of similarity percentage for each model.

Following Conclusions can be made from the above results:

(1) All three recommendation systems achieve high values of similarity, which indicates strong likeness between recommended items in a set.

(2) However, having high values of similarity can make recommendations systems monotonous and affect their ability to make unique recommendations.

Finally, the results in this section can be summarized as:

Serial No.	Model Name	Accuracy	Scope	Similarity
1.	KNN regressor based model	94.63%	10.8%	90.31%
2.	Item-based collaborative filtering model	94.97%	3%	91.25%
3.	User-based collaborative filtering model	91.47%	1.4%	90.01%

Table 7: overall performance of all systems under all the metrics.

From the above study, it can therefore, be concluded that:

(1) There seems to be some degree of correlation between the accuracy and similarity values, based on the results of this study, the item-based collaborative filtering model had the higher values of accuracy and similarity percentages than KNN regressor based model for the same samples, while the user-based recommendation system should not be compared with the other systems due to the use of different samples, it is worth noting that it had the lowest values of accuracy and similarity for any recommendation system model.

(2) None of the three recommendation system models performed well in case of scope metric. The possible reasons for this could be: - (a) lack of observations performed. (b)poor definition of metric.

(3) The recommendation systems performed well in case of accuracy metric. The content recommended by these systems had less difference with ratings of the content based on which they were recommended, or in case of user-based recommendation systems the recommendation scores for items had low difference compared to actual rating of items.

Additional Tables, Figures and Graphs:

	A	B	C	D	E	F	G
1		MAE	MAEP	MAEP	MAEP/100	Accuracy	
2	0	0.502	6.411239	6.41	0.0641	0.9359	
3	1	0.17	2.171137	2.17	0.0217	0.9783	
4	2	0.238	3.039591	3.04	0.0304	0.9696	
5	3	0.182	2.324393	2.32	0.0232	0.9768	
6	4	0.508	6.487867	6.49	0.0649	0.9351	
7	5	0.556	7.100894	7.1	0.071	0.929	
8	6	0.818	10.447	10.45	0.1045	0.8955	
9	7	0.178	2.276215	2.28	0.0228	0.9772	
10	8	0.856	10.94629	10.95	0.1095	0.8905	
11	9	0.308	3.938619	3.94	0.0394	0.9606	
12	10	0.88	11.2532	11.25	0.1125	0.8875	
13	11	0.214	2.736573	2.74	0.0274	0.9726	
14	12	0.274	3.503836	3.5	0.035	0.965	
15	13	0.53	6.777494	6.78	0.0678	0.9322	
16	14	0.254	3.248082	3.25	0.0325	0.9675	
17	15	0.038	0.485934	0.49	0.0049	0.9951	
18	16	0.576	7.365729	7.37	0.0737	0.9263	
19	17	0.576	7.365729	7.37	0.0737	0.9263	
20	18	1.008	12.89003	12.89	0.1289	0.8711	
21	19	0.3	3.836317	3.84	0.0384	0.9616	
22	20	1.228	15.70332	15.7	0.157	0.843	
23	21	0.514	6.57289	6.57	0.0657	0.9343	
24	22	0.132	1.68798	1.69	0.0169	0.9831	
25	23	0.664	8.491049	8.49	0.0849	0.9151	
26	24	0.108	1.381074	1.38	0.0138	0.9862	

Table 8: Table showing sample results for accuracy metric in KNN Regressor based Recommendation system.

	A	B	C	D	E	F	G
1	NAME	MAE	MAEP	MAEP	MAEP/10(	Accuracy	
2	Persona 3 the Movie 1: Spring of Birth	0.663333	8.47169	8.47	0.0847	0.9153	
3	Rozen Maiden: Ouvertüre	0.068333	0.872712	0.87	0.0087	0.9913	
4	Seikai no Monshou	0.401667	5.129842	5.13	0.0513	0.9487	
5	Shaman King	0.54	6.896552	6.9	0.069	0.931	
6	Sword Art Online	0.511667	6.534696	6.53	0.0653	0.9347	
7	Tetsuwan Birdy Decode:02	0.18	2.298851	2.3	0.023	0.977	
8	Bishoujo Senshi Sailor Moon Crystal Season III	0.558333	7.130694	7.13	0.0713	0.9287	
9	Bokurano	0.386667	4.944587	4.94	0.0494	0.9506	
10	Bounen no Xamdou	0.205	2.621483	2.62	0.0262	0.9738	
11	Break Blade 6: Doukoku no Toride	0.651667	8.333333	8.33	0.0833	0.9167	
12	Detective Conan: Conan vs. Kid - Shark & Jewel	0.125	1.598465	1.6	0.016	0.984	
13	Dragon Ball Z Special 2: Zetsubou e no Hankou!! Nokosaret	0.283333	3.623188	3.62	0.0362	0.9638	
14	Hakkenden: Touhou Hakken Ibun 2nd Season	0.695	8.887468	8.89	0.0889	0.9111	
15	Hidamari Sketch x 365 Specials	0.623333	7.971014	7.97	0.0797	0.9203	
16	K: Return of Kings	0.581667	7.438193	7.44	0.0744	0.9256	
17	Kill la Kill Special	0.676667	8.653026	8.65	0.0865	0.9135	
18	Kochira Katsushikaku Kameari Kouenmae Hashutsujo (TV)	0.011667	0.14919	0.15	0.0015	0.9985	
19	Macross Plus	0.246667	3.154305	3.15	0.0315	0.9685	
20	Macross Plus Movie Edition	0.226667	2.898551	2.9	0.029	0.971	
21	Mitsudomoe Zouryouchuu!	0.331667	4.241262	4.24	0.0424	0.9576	
22	Prince of Tennis: Another Story - Messages From Past and F	0.528333	6.756181	6.76	0.0676	0.9324	
23	Saint Seiya: Meiou Hades Elysion-hen	0.321667	4.113384	4.11	0.0411	0.9589	
24	To LOVE-Ru Darkness	0.403333	5.157715	5.16	0.0516	0.9484	
25	To LOVE-Ru Darkness OVA	0.616667	7.885763	7.89	0.0789	0.9211	
26	Working!!	0.203333	2.600171	2.6	0.026	0.974	

Table 9 : Table showing sample results for accuracy metric in item-based collaborative filtering Recommendation system.

User_ID	MAE	MAE_percent	MAEP	MAEP./100	Accuracy
140	0.399464	4.91879257	4.92	0.0492	0.9508
141	0.601128	7.606530303	7.61	0.0761	0.9239
142	0.464167	5.23728077	5.24	0.0524	0.9476
143	0.585214	6.515772261	6.52	0.0652	0.9348
144	1.141333	12.12262469	12.13	0.1213	0.8787
146	1.138333	12.02109663	12.03	0.1203	0.8797
147	1.753289	25.08237926	25.09	0.2509	0.7491
148	1.379405	19.1476662	19.15	0.1915	0.8085
149	0.387087	4.267635039	4.27	0.0427	0.9573
150	0.655286	6.959175038	6.96	0.0696	0.9304
152	2.391861	38.73327937	38.74	0.3874	0.6126
153	0.46015	5.510032262	5.52	0.0552	0.9448
154	0.48719	5.237318346	5.24	0.0524	0.9476
600	0.543	5.921539844	5.93	0.0593	0.9407
601	0.26395	3.073661153	3.08	0.0308	0.9692
602	0.834667	8.667180984	8.67	0.0867	0.9133
603	0.644667	7.118829808	7.12	0.0712	0.9288
604	2.580985	41.54388492	41.55	0.4155	0.5845
605	0.518285	5.892859018	5.9	0.059	0.941
606	0.48213	6.07848367	6.08	0.0608	0.9392
607	1.097857	11.26709926	11.27	0.1127	0.8873
608	0.220333	2.41889895	2.42	0.0242	0.9758
609	0.331179	3.857462521	3.86	0.0386	0.9614
705	0.626293	7.854144952	7.86	0.0786	0.9214
706	0.657	7.119286134	7.12	0.0712	0.9288
707	0.598305	6.443346328	6.45	0.0645	0.9355
708	0.65631	6.894647054	6.9	0.069	0.931

Table10 : Table showing sample results for accuracy metric in user-based collaborative filtering Recommendation system.

	A	B	C	D	E
1		name	mean_similarity_deviation	M_S_D	similarity percentage
2	100	Diamond no Ace: Second Season	11.63354665	11.63	88.37
3	101	Magi: The Kingdom of Magic	9.842738205	9.84	90.16
4	102	Mahou Shoujo Madoka★Magica Movie 3: Hangyaku no Monogatari	12.4902298	12.49	87.51
5	103	Major: World Series	9.249370044	9.25	90.75
6	104	Samurai Champloo	11.430986	11.43	88.57
7	105	Shokugeki no Souma: Ni no Sara	16.24181988	16.24	83.76
8	106	Katanagatari	7.875194032	7.88	92.12
9	107	Mahou Shoujo Madoka★Magica Movie 2: Eien no Monogatari	12.46320292	12.46	87.54
10	108	Major S6	9.783000033	9.78	90.22
11	109	Mononoke	6.525349101	6.53	93.47
12	110	Shirobako	14.41623333	14.42	85.58
13	111	Ashita no Joe 2	12.22384036	12.22	87.78
14	112	Hunter x Hunter	11.70349936	11.7	88.3
15	113	Noragami Aragoto	13.986767	13.99	86.01
16	114	Sakamichi no Apollon	10.16546457	10.17	89.83
17	115	Tonari no Totoro	10.80308031	10.8	89.2
18	116	Ghost in the Shell: Stand Alone Complex	8.025493472	8.03	91.97
19	117	Kaze no Tani no Nausicaä	6.040162498	6.04	93.96
20	118	No Game No Life	8.342407743	8.34	91.66
21	119	Romeo no Aoi Sora	11.53926702	11.54	88.46
22	120	Yuu☆Yuu☆Hakusho	9.206063335	9.21	90.79
23	121	Kino no Tabi: The Beautiful World	7.391629728	7.39	92.61
24	122	Kuroko no Basket	7.91655845	7.92	92.08
25	123	Nodame Cantabile	8.713517433	8.71	91.29
26	124	Ookami to Koushinryou II	6.896300125	6.9	93.1
27	125	Shingeki no Kyojin: Kuinaki Sentaku	10.07787622	10.08	89.92
28	126	Steins;Gate: Oukoubakko no Poriomania	9.995033373	10	90

Table11: Table showing sample results for similarity metric in KNN Regressor based Recommendation system.

	A	B	C	D
1		name	percent deviation from mean similarity	similarity %
2	100	Diamond no Ace: Second Season	5.144593462	94.86
3	101	Magi: The Kingdom of Magic	2.795966162	97.2
4	102	Mahou Shoujo Madoka★Magica Movie 3: Hangyaku no Monogat	4.990954047	95.01
5	103	Major: World Series	12.06374823	87.94
6	104	Samurai Champloo	8.464281323	91.54
7	105	Shokugeki no Souma: Ni no Sara	7.58410749	92.42
8	106	Katanagatari	10.97078849	89.03
9	107	Mahou Shoujo Madoka★Magica Movie 2: Eien no Monogatari	5.924466338	94.08
10	108	Major S6	13.59020628	86.41
11	109	Mononoke	11.32887899	88.67
12	110	Shirobako	4.273158905	95.73
13	111	Ashita no Joe 2	9.68292033	90.32
14	112	Hunter x Hunter	6.766309836	93.23
15	113	Noragami Aragoto	9.301191389	90.7
16	114	Sakamichi no Apollon	15.31735581	84.68
17	115	Tonari no Totoro	7.83559425	92.16
18	116	Ghost in the Shell: Stand Alone Complex	3.979617834	96.02
19	117	Kaze no Tani no Nausicaä	8.507239141	91.49
20	118	No Game No Life	10.0920945	89.91
21	119	Romeo no Aoi Sora	7.187729201	92.81
22	120	Yuu☆Yuu☆Hakusho	14.2265015	85.77
23	121	Kino no Tabi: The Beautiful World	8.620577735	91.38
24	122	Kuroko no Basket	7.765301527	92.23
25	123	Nodame Cantabile	7.317839196	92.68
26	124	Ookami to Koushinryou II	6.265889038	93.73
27	125	Shingeki no Kyojin: Kuinaki Sentaku	12.33976105	87.66
28	126	Steins;Gate: Oukoubakko no Poriomania	5.457539402	94.54

Table12 : Table showing sample results for similarity metric in item-based collaborative filtering Recommendation system.

	A	B	C
1	User_ID	Percent Deviation from mean	similarity
2	140	5.166232572	94.83
3	141	10.72409135	89.28
4	142	14.44858884	85.55
5	143	11.83034008	88.17
6	144	12.73654976	87.26
7	146	7.599911401	92.4
8	147	6.490848825	93.51
9	148	4.069259022	95.93
10	149	5.962364795	94.04
11	150	9.248075107	90.75
12	152	9.171463457	90.83
13	153	6.992439609	93.01
14	154	7.133267699	92.87
15	600	11.56721805	88.43
16	601	11.34805241	88.65
17	602	8.237393507	91.76
18	603	7.798327686	92.2
19	604	9.342105263	90.66
20	605	8.304158306	91.7
21	606	7.524361663	92.48
22	607	8.976871815	91.02
23	608	8.746364251	91.25
24	609	14.03320497	85.97
25	705	15.63278177	84.37
26	706	10.76521739	89.23
27	707	5.262750469	94.74
28	708	11.55855296	88.44

Table13 : Table showing sample results for similarity metric in user-based collaborative filtering Recommendation system.

6. FUTURE WORK:

This aim of this study was to study three different types of recommendation systems and evaluate their performances based on three performance metrics – accuracy, scope and similarity. Based on the findings of this study the suggested future work includes studying the relationship between accuracy of recommendations and similarity between recommended results. Other suggested work includes:

I. The effect that number of episodes or type of content have on these recommendations.
II. Evaluation of more complex recommendation systems based on the similarity metric proposed.

REFERENCES:

[1.] “Improving Recommendation Lists Through Topic Diversification” c, Ziegler CN et. al.

[2.] “Rank and Relevance in Novelty and Diversity Metrics for Recommender Systems”, Vargas S. and Castells P., Universidad Autónoma de Madrid Escuela Politécnica Superior, Departamento de Ingeniería Informática.

[3.]"Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," Adomavicius G. and Tuzhilin A., *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734-749, June 2005.

[4.] "Collaborative Filtering Recommender Systems", Ekstrand M.D., Riedl J.T. and Konstan J. A. (2011), Foundations and Trends® in Human–Computer Interaction: Vol. 4: No. 2, pp 81-173

[5.] “The BellKor Solution to the Netflix Grand Prize”, Koren Y., August 2009

[6.] “Application of Dimensionality Reduction in Recommender System -- A Case Study”, Sarwar B.M. et. al., GroupLens Research Group / Army HPC Research Center Department of Computer Science and Engineering University of Minnesota.

Appendix 1: Sample code used for the project:

In K-NN Regressor Model:

```
# -*- coding: utf-8 -*-
"""
Created on Tue Mar 31 13:10:02 2020

@author: anadi
"""

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn  as sns



# Importing the dataset
anime_df=pd.read_csv('anime.csv')
anime3_df=pd.read_csv('anime3.csv')

#filling tuples with unknown values
values={'genre':'Unknown','members':0,'rating':0}
anime_df.fillna(value=values,inplace=True)

#dropping nan values
anime_df.dropna(how='any',inplace=True )
anime3_df.dropna(how='any',inplace=True)

#resetting the indices
anime_df=anime_df[(anime_df != 0).all(1)]
anime_df=anime_df.reset_index()

anime_df=anime_df.iloc[:,1:]


#creating a list of top 15 most frequently occurring genres across the dataset
top_15_genre_list=anime3_df.value=anime3_df['genre'].value_counts().sort_values(ascending=
False).head(15).index

#create dummies for top 15 genre
for i in range(0,14):
    anime_df[top_15_genre_list[i]]=0
#populating the dummies for top 15 genre
for i in range(0,14):
```

```
    for j in range(0,len(anime_df)):
        anime_df[top_15_genre_list[i]][j]=np.where( top_15_genre_list[i] in anime_df['genre'][j]
,1,0)

#seperation of independent and dependent variables
X=anime_df.iloc[:,6:20].values
y=anime_df.iloc[:,5].values


#train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
#sc_y = StandardScaler()
#y_train = sc_y.fit_transform(y_train.reshape(-1,1))

#fitting classifier into training set
from sklearn.neighbors import KNeighborsRegressor
classifier=KNeighborsRegressor(n_neighbors=5,algorithm='auto',
metric='minkowski',p=2,weights='uniform')
classifier.fit(X_train,y_train)

#making prediction of ratings for test values
y_pred=classifier.predict(X_test)


'''for train set
y_pred_entire=classifier.predict(X_train)
y_diff_entire=y_pred_entire-y_train
MAEP=(y_diff_entire/y_train)*100
Avg_MAEP=sum(MAEP)/len(MAEP)'''
"""from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
"""

#sample object of FetchNeibours class for prediction and evaluations
new_pred=FetchNeighbors()
abc=new_pred.getNeighbors(10,y_pred_entire[15],anime_df)
abc2=new_pred.getNeighbors(10,y_test[15],anime_df_new)
```

```
#absolute deviation across test set
y_diff=abs(y_pred-y_test)
'''abc = filter(lambda x: x < 1.5, y_diff)
y_diff=list(abc)'''

#MAE across test set
sum(y_diff)/len(y_diff)

#MAEP
MAEP=(y_diff/y_test)*100
Avg_MAEP=sum(MAEP)/len(MAEP)

#function for calculating squares and roots of lists
def square(list):
    return [i ** 2 for i in list]
def square_root(list):
    return [i ** 0.5 for i in list]


#comparison of results between KNN regressor model and Item based collaborative filtering
model
sample=anime_df['name'][800:900]
sample_knn=pd.merge(sample,anime_df,on='name')
sample_knn_X=sample_knn.iloc[:,7:21].values
sample_knn_y=sample_knn.iloc[:,6].values
#feature scaling
sample_knn_X=sc_X.transform(sample_knn_X)

sample_pred=classifier.predict(sample_knn_X)

#calculation of accuracy
difference_sample=abs(sample_knn_y-sample_pred)
MAE_sample=sum(difference_sample)/len(difference_sample)
MAEP_knn=(difference_sample/sample_knn_y)*100
Avg_MAEP_knn=sum(MAEP_knn)/len(MAEP_knn)
MAE_list=pd.DataFrame(difference_sample)
MAEP_list=pd.DataFrame(MAEP_knn)
#merging dataframes for plotting
sample_results_knn=pd.merge(MAE_list,MAEP_list,left_index=True,right_index=True)
```

```
#FetchNeighbours class:

import pandas as pd
import numpy as np
class FetchNeighbors:

    def ___init__(self, name):
        self.name = name
    #creating distance variable
    from numpy import zeros
    anime_df_new=pd.DataFrame(anime_df)
    distance=zeros([len(anime_df)])
    anime_df_new=pd.DataFrame(anime_df)
    anime_df_new['distance']=np.nan
    values={'distance':0}
    anime_df_new.fillna(value=values,inplace=True)

   #getNeighbors function returns a list of content with least distance from predicted set
    def getNeighbors(self,n,pred,anime_df_new):
        neighbor_list=[]
        for i in range(0,len(anime_df_new['distance'])):
            anime_df_new['distance'][i]=abs(pred-anime_df_new['rating'][i])
        neighbor_list=anime_df_new.sort_values(ascending=True,by='distance').head(n)
        return neighbor_list

    #to predict for any tuple
    def predict_single(self,a,i):
       a=X[i]
       a=np.array(a)
       a=np.expand_dims(a,0)
       b=classifier.predict(a)
       return b
```

In item based collaborative filtering model:

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn  as sns


# Importing the dataset
anime_df=pd.read_csv('anime.csv')
rating_df=pd.read_csv('rating.csv')
```

```
#sample set
#sample=anime_df['name'][838:900]

#merge the dataframes
anime_rating_df=pd.merge(anime_df,rating_df, on='anime_id')

#visualization of data
anime_rating_df.groupby('name')['rating_x'].describe()



#mean and count dataframes
ratings_df_mean=anime_rating_df.groupby('name')['rating_y'].describe()['mean']
ratings_df_count=anime_rating_df.groupby('name')['rating_y'].describe()['count']

ratings_mean_count_df=pd.concat([ratings_df_mean,ratings_df_count],axis=1)


#userid movieid matrix
userid_anime_matrix=anime_rating_df.pivot_table(index='user_id',columns='name',values='ratin
g_y')

#recommendations for single user
u1=CollaborativeItemBased()
u1.accuracy(5,'Gintama',userid_anime_matrix)
u1.getSimilarItems(5,'Gintama',userid_anime_matrix)
getSimilarItems(5,'Gintama',userid_anime_matrix)

#importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn  as sns
import scipy as sp

class CollaborativeItemBased:
    def ___init__(self, name):
      self.name = name
    #getSimilarItems returns a list of recommended content
    def getSimilarItems(n,item,matrix):
      item_matrix=matrix
      item_matrix_correlation=pd.DataFrame(userid_anime_matrix.corrwith(
matrix[item]),columns=['Correlations'])
      item_matrix_correlation.dropna(inplace=True)
      item_matrix_correlation=item_matrix_correlation.join(ratings_mean_count_df['count'])
```

```
        recommended_content_list=
item_matrix_correlation[item_matrix_correlation['count']>100].sort_values('Correlations',ascend
ing=False).head(n+1)
        return  recommended_content_list


    #functions for calculating squares and roots of lists
    def square(list):
        return [i ** 2 for i in list]

    def square_root(list):
        return [i ** 0.5 for i in list]


    #Calculation of MAE for evaluation
    def meanAbsoluteError(n,item,matrix):
      rec_content_list=getSimilarItems(n,item,matrix)
      accuracy_list=pd.merge(anime_df,rec_content_list,on='name')
      item_rating=anime_df[anime_df.name==item]['rating']
      avg_rating=sum(accuracy_list['rating'])/len(accuracy_list['rating'])
      accuracy_score_mae=abs(item_rating-avg_rating).tolist()
      accuracy_score_mae=accuracy_score_mae[0]
      return accuracy_score_mae


   #Calculation of MAE for evaluation
   def meanAbsoluteErrorPercentage(n,item,matrix):
        rec_content_list=getSimilarItems(n,item,matrix)
        accuracy_list=pd.merge(anime_df,rec_content_list,on='name')
        item_rating=anime_df[anime_df.name==item]['rating']
        avg_rating=sum(accuracy_list['rating'])/len(accuracy_list['rating'])
        accuracy_score_mae=abs(item_rating-avg_rating)
        accuracy_score_maep=((accuracy_score_mae/item_rating)*100).tolist()
        accuracy_score_maep=accuracy_score_maep[0]
        return accuracy_score_maep


    #evaluation of model
    def getAccuracyList(n,item,matrix):
          rec_content_list=getSimilarItems(n,item,matrix)
          accuracy_list=pd.merge(anime_df,rec_content_list,on='name')
          accuracy_list.dropna(item)
          return accuracy_list
```

```
def getError(start_index,sample_size):
    accuracy_score_list=[]
    while sample_size!=0:

accuracy_score_list.append(accuracy(6,anime_df.index[start_index],userid_anime_matrix))
        start_index+=1
        sample_size-=1
    return accuracy_score_list

def sampling(i,list1,list2,list3):
    list1.append(meanAbsoluteError(5,i,userid_anime_matrix))
    list2.append(meanAbsoluteErrorPercentage(5,i,userid_anime_matrix))
    list3.append(i)

test1=[]
test2=[]
index1=[]



xyz=meanAbsoluteError(5,'Gintama',userid_anime_matrix)
sampling('Gintama',test1,test2,index1)


index1.pop(5)
test1.pop(5)
test2.pop(5)


#comparison of results
#meanAbsoluteErrorPercentage(5,'Gintama',userid_anime_matrix)
MAE_list_collaborative_item=[]
MAEP_list_collaborative_item=[]
Name_list=[]


sample=sample.tolist()
for i in sample:
     MAE_list_collaborative_item.append(meanAbsoluteError(5,i,userid_anime_matrix))
     MAEP_list_collaborative_item.append(
meanAbsoluteErrorPercentage(5,i,userid_anime_matrix))
     Name_list.append(i)


MAE_list_collaborative_item=pd.DataFrame( MAE_list_collaborative_item)
```

```
    MAEP_list_collaborative_item=pd.DataFrame(MAEP_list_collaborative_item)
    Name_list=pd.DataFrame(Name_list)


sample_result_collaborative_item=pd.merge(Name_list,MAE_list_collaborative_item,left_index
=True,right_index=True)
    sample_result_collaborative_item=pd.merge(
sample_result_collaborative_item,MAEP_list_collaborative_item,left_index=True,right_index=
True)

    new1= meanAbsoluteError(5,'Persona 3 the Movie 1: Spring of Birth',userid_anime_matrix)
    new2= meanAbsoluteErrorPercentage(5,'Persona 3 the Movie 1: Spring of
Birth',userid_anime_matrix)
```

For user-based collaborative filtering model:

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn  as sns
import scipy as sp


# Importing the dataset
anime_df=pd.read_csv('anime.csv')
rating_df=pd.read_csv('rating.csv')


#merge the dataframes
anime_rating_df=pd.merge(anime_df,rating_df, on='anime_id')

anime_rating_df.groupby('user_id')['rating_y'].describe()



#mean and count dataframes
ratings_df_mean=anime_rating_df.groupby('user_id')['rating_y'].describe()['mean']
ratings_df_count=anime_rating_df.groupby('user_id')['rating_y'].describe()['count']

#merging mean and count dataframes
ratings_mean_count_df=pd.concat([ratings_df_mean,ratings_df_count],axis=1)
```

```
#userid movieid matrix
userid_anime_matrix=anime_rating_df.pivot_table(index='name',columns='user_id',values='ratin
g_y')
#userid_anime_matrix=userid_anime_matrix.transpose()

"""
Created on Thu Apr  9 03:25:04 2020

@author: anadi
"""
#importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn  as sns
import scipy as sp

class Collaborative:
    def ___init__(self, name):
      self.name = name
    #getSimilarUsers returns a list of n users from matrix which have similar interests to the user
    def getSimilarUsers(n,user,matrix):
      user_matrix=matrix
      user_matrix_correlation=pd.DataFrame(user_matrix.corrwith(
user),columns=['Correlations'])
      user_matrix_correlation.dropna(inplace=True)
      user_matrix_correlation=user_matrix_correlation.join(ratings_mean_count_df['count'])
      neighbor_list=
user_matrix_correlation[user_matrix_correlation['count']>200].sort_values('Correlations',ascendi
ng=False).head(n)
      return neighbor_list

     #getSimilarUsers(100,userid_anime_matrix[44],userid_anime_matrix)


    '''def recommendAnime(number_recommended,n,user,matrix):
        neighbor_list=getSimilarUsers(n,user,matrix)
        #rec_list=anime_rating_df=pd.merge(rating_df,neighbor_list, on='user_id')
        rec_lis=pd.merge(rating_df,neighbor_list, on='user_id')
        rec_list['rec_score']=rec_list['rating']*rec_list['Correlations']

recommended_anime=rec_list.sort_values(by='rec_score',ascending=False).head(number_recom
mended)
        recommended_anime_name=pd.merge(
recommended_anime,anime_df,on='anime_id').drop(axis=1,columns=['episodes','user_id','type','
Correlations','members','genre'])
```

```
        return  recommended_anime_name
    '''
    #functions for calculating squares and roots of lists
    def square(list):
        return [i ** 2 for i in list]
    def square_root(list):
        return [i ** 0.5 for i in list]
    #recommendAnime returns a list containing recommended content based on recommendation
score
    def recommendAnime(number_recommended,n,user,matrix):
        neighbor_list=getSimilarUsers(n,user,matrix)
        rec_list=pd.merge(rating_df,neighbor_list, on='user_id')
        rec_list['rec_score']=rec_list['rating']*rec_list['Correlations']
        rec_list_count=rec_list.groupby('anime_id')['rec_score'].describe()['count']
        rec_list_mean=rec_list.groupby('anime_id')['rec_score'].describe()['mean']
        rec_list_mean_count=pd.merge(rec_list_count,rec_list_mean,on='anime_id').dropna()

recommend_anime=rec_list_mean_count.sort_values(by='count',ascending=False).head(100)

recommend_anime_final=recommend_anime.sort_values(by='mean',ascending=False).head(n)
        recommended_anime_name=pd.merge(
recommend_anime_final,anime_df,on='anime_id').drop(axis=1,columns=['episodes','type','memb
ers','genre'])
        return  recommended_anime_name

    #similar to recommendAnime but ruturns list of anime id
    def recommendAnimeList(number_recommended,n,user,matrix):
        neighbor_list=getSimilarUsers(n,user,matrix)
        rec_list=pd.merge(rating_df,neighbor_list, on='user_id')
        rec_list['rec_score']=rec_list['rating']*rec_list['Correlations']
        rec_list_count=rec_list.groupby('anime_id')['rec_score'].describe()['count']
        rec_list_mean=rec_list.groupby('anime_id')['rec_score'].describe()['mean']
        rec_list_mean_count=pd.merge(rec_list_count,rec_list_mean,on='anime_id').dropna()

recommend_anime=rec_list_mean_count.sort_values(by='count',ascending=False).head(100)

recommend_anime_final=recommend_anime.sort_values(by='mean',ascending=False).head(n)
        return  recommend_anime_final

    #calculation of MAE for evaluation
    def meanAbsoluteError(number_recommended,n,user,matrix):
        acc_list=recommendAnimeList(number_recommended,n,user,matrix)
        sim=pd.merge(acc_list,anime_df,on='anime_id')
        deviation_mean_rating=abs(sim['mean']-sim['rating'])
        mean_absolute_error=sum( deviation_mean_rating)/len( deviation_mean_rating)
        return mean_absolute_error
```

```
    #calculation of MAEP for evaluation
    def meanAbsoluteErrorPercentage(number_recommended,n,user,matrix):
        acc_list=recommendAnimeList(number_recommended,n,user,matrix)
        sim=pd.merge(acc_list,anime_df,on='anime_id')
        deviation_mean_rating=abs(sim['mean']-sim['rating'])
        mean_absolute_error=sum(deviation_mean_rating)/len(deviation_mean_rating)
        mean_absolute_error_percentage_list=(deviation_mean_rating/sim['mean'])*100

mean_absolute_error_percentage=sum(mean_absolute_error_percentage_list)/len(mean_absolute
_error_percentage_list)
        return mean_absolute_error_percentage


    #sampling for observations
    def sampling(i,list1,list2,list3):
        list1.append(meanAbsoluteError(10,10,userid_anime_matrix[i],userid_anime_matrix))

list2.append(meanAbsoluteErrorPercentage(10,10,userid_anime_matrix[i],userid_anime_matrix)
)
        list3.append(i)
```