PRACTICAL DATA SCIENCE

ASSIGNMENT 3: Virtual Presentation and interview

Practical Data Science with Python
COSC 2670/2738

I certify that this is all my own original work. If I took any parts from elsewhere, then they were non-essential parts of the assignment, and they are clearly attributed in our submission. I will show I agree to this honour code by typing "Yes": YES

Mohammed Usman E Ghani

RMIT University

S3901999

07/06/2022

CONTENTS IN THIS PRESENTATION

THE DATA SCIENCE PROBLEM

COLLABORATIVE FILTERING

- It is a technique that is used to filter items that a user might like based on reactions by other similar users. What it does is that it searches a large group of people and finds a smaller set of users with tastes similar to a particular user.

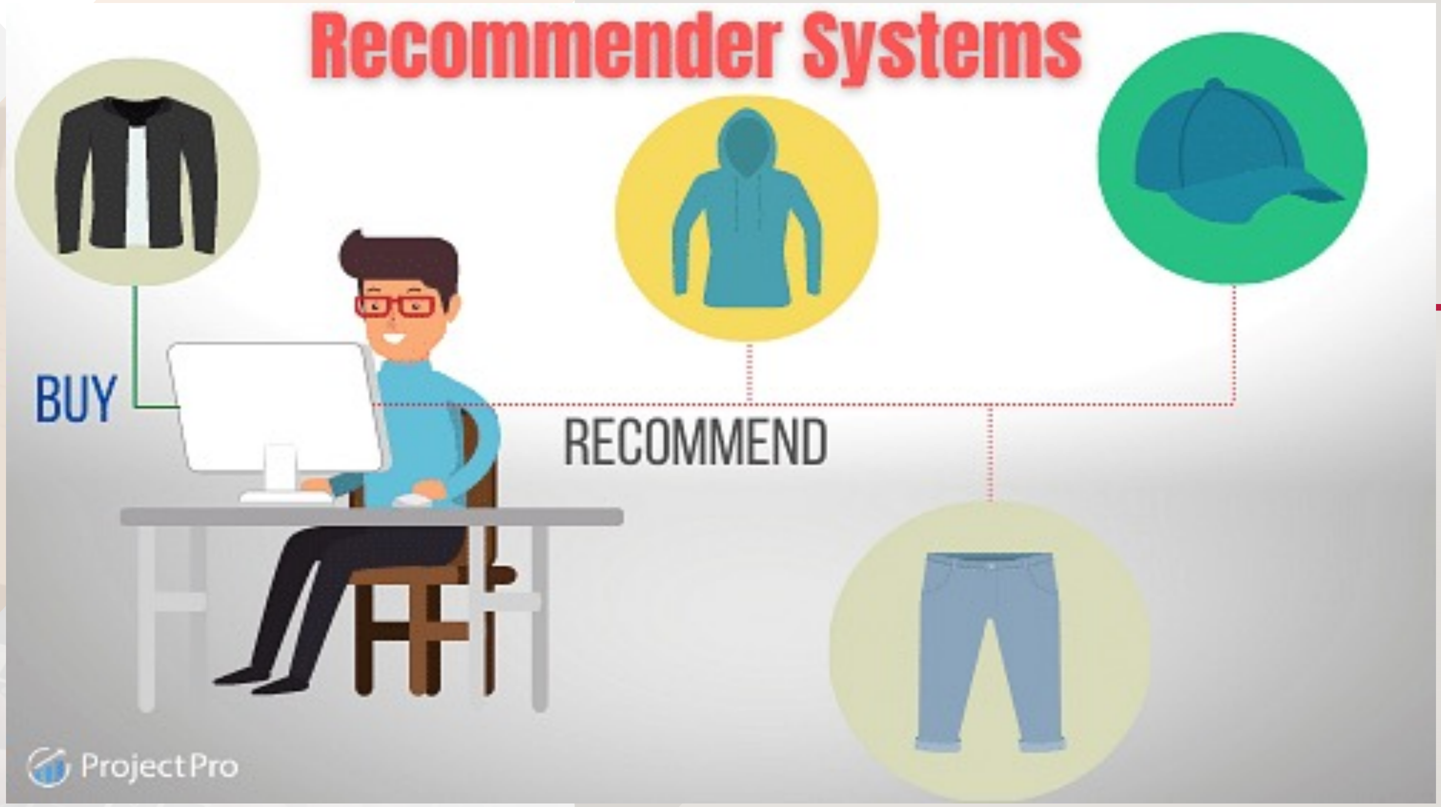


Figure 1: Recommender system visualization Grover, P. (2017).

- Collaborative filtering (CF) is a technique used by many recommender systems.
- It involves predicting the users' choices and offer relevant suggestions to users

SOME COMPANIES THAT USE COLLABORATIVE FILTERING

- Collaborative filtering uses rating information from all other users to provide predictions for a user-item interaction and, thereby, whittles down the item choices for the users, from the complete item set.

LIMITATIONS AND CHALLENGES

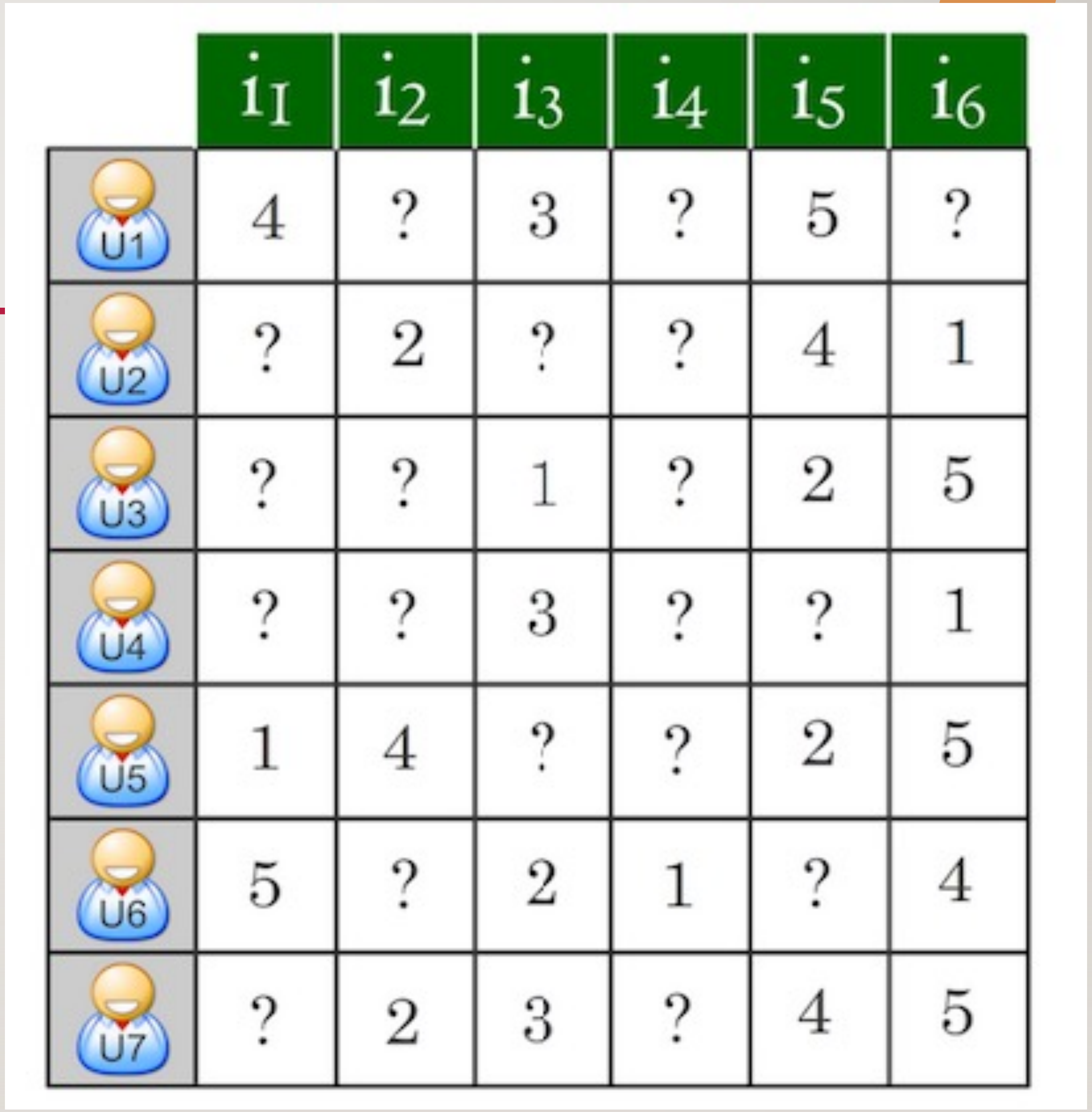
Collaborative filtering utilises the K-nearest neighbour like model which is considered to be a great baseline for developing recommender systems. It has also been proved to be a highly successful approach. However, there are some limitations to it and it still faces some challenges.

- The problem of sparsity (missing values)
- Estimating the similarity between users or items
- Scalability, most of the algorithms often suffer serious scalability problem.
- Dealing with new users with no useful data

SPARSITY PROBLEM

The sparsity problem refers to a situation in which the feedback data is sparse and inadequate to identify similarities in user's interests

It becomes difficult to find enough reliable and similar information, as a in general the active users would have only had provided ratings to a small portion of the items.



	i_1	i_2	i_3	i_4	i_5	i_6
U1	4	?	3	?	5	?
U2	?	2	?	?	4	1
U3	?	?	1	?	2	5
U4	?	?	3	?	?	1
U5	1	4	?	?	2	5
U6	5	?	2	1	?	4
U7	?	2	3	?	4	5

Figure 2: sparsity problem visual

COMMONLY USED SIMILARITY MEASURES

The vital component of collaborative filtering is to estimate the similarity between users or items. The performance of a recommender system depends on how accurately it is able to give the predictions. The system's this ability depends on similarity measure that is being used to find similar users.

Some of the commonly used similarity measure in recommended systems are Cosine, Pearson correlation coefficient, Vector space similarity and the Euclidean method.

QUICK FACTS

Netflix, one of the most popular streaming service's recommendation consists of algorithms that display content to the subscribers based on each individual subscriber's profile. The recommendation engine filters out more than 3000 items at a time with the help of clustering (recommendations) on the basis of subscriber's preference.

RELATED SOLUTIONS

Several solutions have been proposed to address the limitations and challenges of the current collaborative filtering model.

- Reducing the dimensionality of the recommender system (Singular Value Decomposition method)

- Fusing all ratings with a predictive value for a recommendation

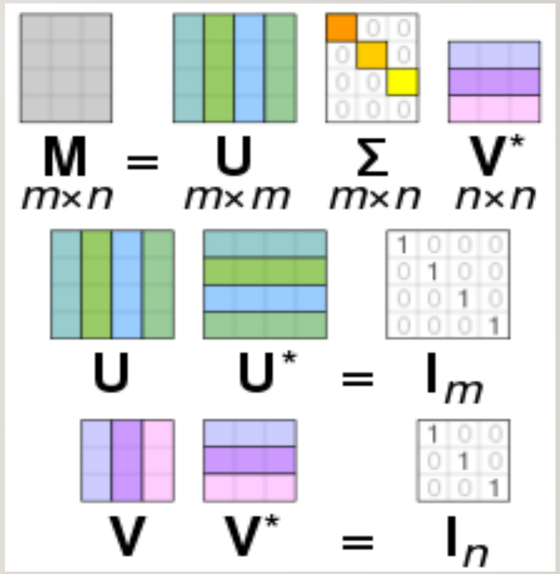


Figure 3: Fusing all ratings with a predictive value for a recommendation Grover, P. (2017).

“A NEW COLLABORATIVE FILTERING APPROACH UTILIZING ITEM’S POPULARITY”

A research has been conducted in which an overview of collaborative filtering recommender system and the major collaborative filtering challenges have been addressed.

A new approach is suggested in which it has been claimed that it can outperform other collaborative filtering algorithms that currently exist.

All of this has been backed up by experimental evidence.

“A NEW COLLABORATIVE FILTERING APPROACH UTILIZING ITEM’S POPULARITY”

A new missing data making up strategy

Suggested in order to smoothen the the sparsity problem before user’s similarity computation.

- utilises item’s category information

-more accurate recommendations

"A NEW COLLABORATIVE FILTERING APPROACH UTILIZING ITEM'S POPULARITY"

Defining item's popularity weight

- Neglect the items' difference – utilizing item's popularity weight in similarity computation

- The existing algorithms related to collaborative filtering do not think of the different significance weights of items with different.

“A NEW COLLABORATIVE FILTERING APPROACH UTILIZING ITEM’S POPULARITY”

Alleviate the new user problem

- Alleviated using ComRV, discussed further in the next slides

- Provide a composite recommendation value which combines the popularity and mean rating value of item.

MAKING UP RATINGS DATA

$$T(a,u) = T(a) \cup T(u)$$

$$N(a) = T(a,u) - T(a)$$

- Increases the number of ratings commonly rated by the users

DEFINE THE POPULARITY SIGNIFICANCE WEIGHT OF ITEM

$$w_t = \log(m/P_t)$$

- The more popular items reflect lower significance whereas the less popular items reflect higher significance

COMPUTING THE SIMILARITY OF USERS

$$sim(a,u) = \frac{\sum_{t \in T(a,u)} w_t^2 (R_{a,t} - \overline{R_a})(R_{u,t} - \overline{R_u})}{\sqrt{\sum_{t \in T(a,u)} w_t^2 (R_{a,t} - \overline{R_a})^2} \sqrt{\sum_{t \in T(a,u)} w_t^2 (R_{u,t} - \overline{R_u})^2}}$$

- The similarity between user a and user u is then given by this correlation formula
- This is when popularity significance come into play

SORTING THE OBTAINED RESULTS

$$KNN(a) = \{knn_1, knn_2, \cdots, knn_k\}$$

- Sorting the results in descending order and forming a neighbour set

PREDICTIONS AND RECOMMENDATIONS

$$P(a,i) = \overline{R_a} + \frac{\sum_{u \in KNN(a)} sim(a,u) \cdot (R_{u,i} - \overline{R_u})}{\sum_{u \in KNN(a)} sim(a,u)}$$

- Using the neighbours’ ratings data on the items, the predictions are computed

NEW USERS' RECOMMENDATIONS

$$ComRV(t) = P_t \cdot \overline{R_t}$$

- Obtaining top-n popular items which are also rated well after which a composite recommendation value such as above is defined

ANALYSING RESULTS

The proposed approach significantly improves the recommendation quality of collaborative filtering, and outperforms UPCC (user based), IPCC(item based) and EMDP. It can also be inferred that more accurate predictions are obtained with the larger numbers of training users and active users' rated items.

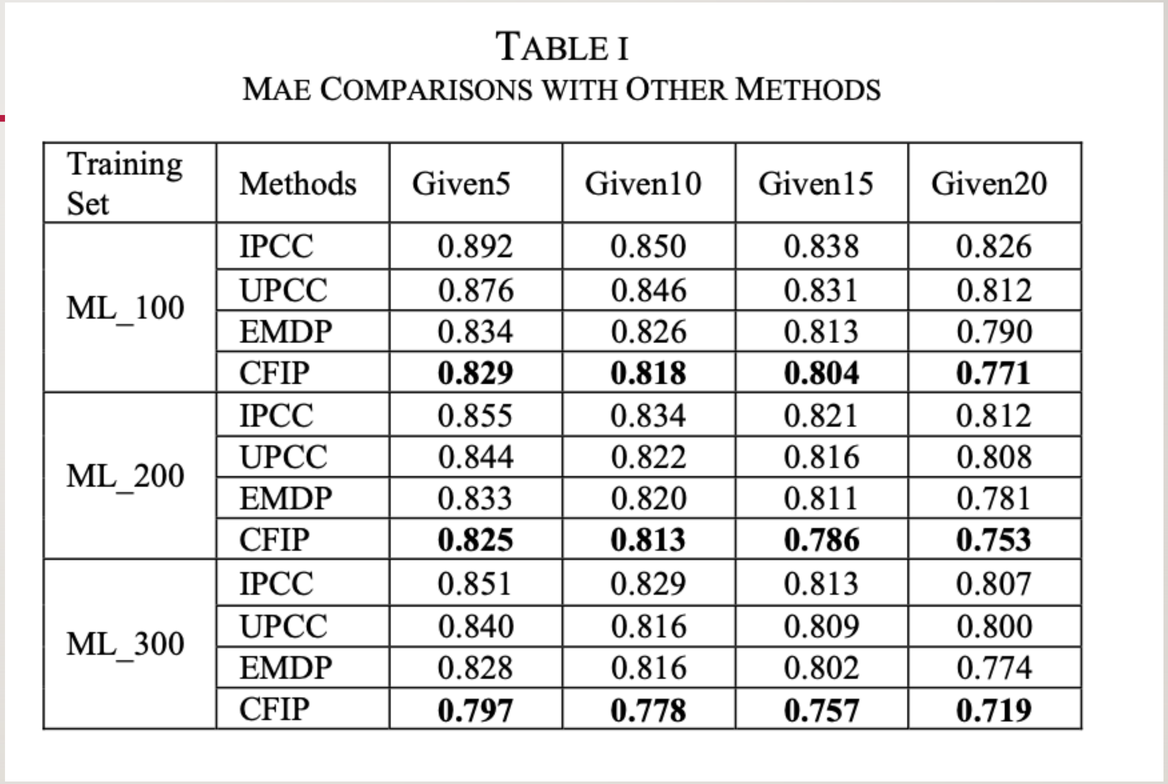
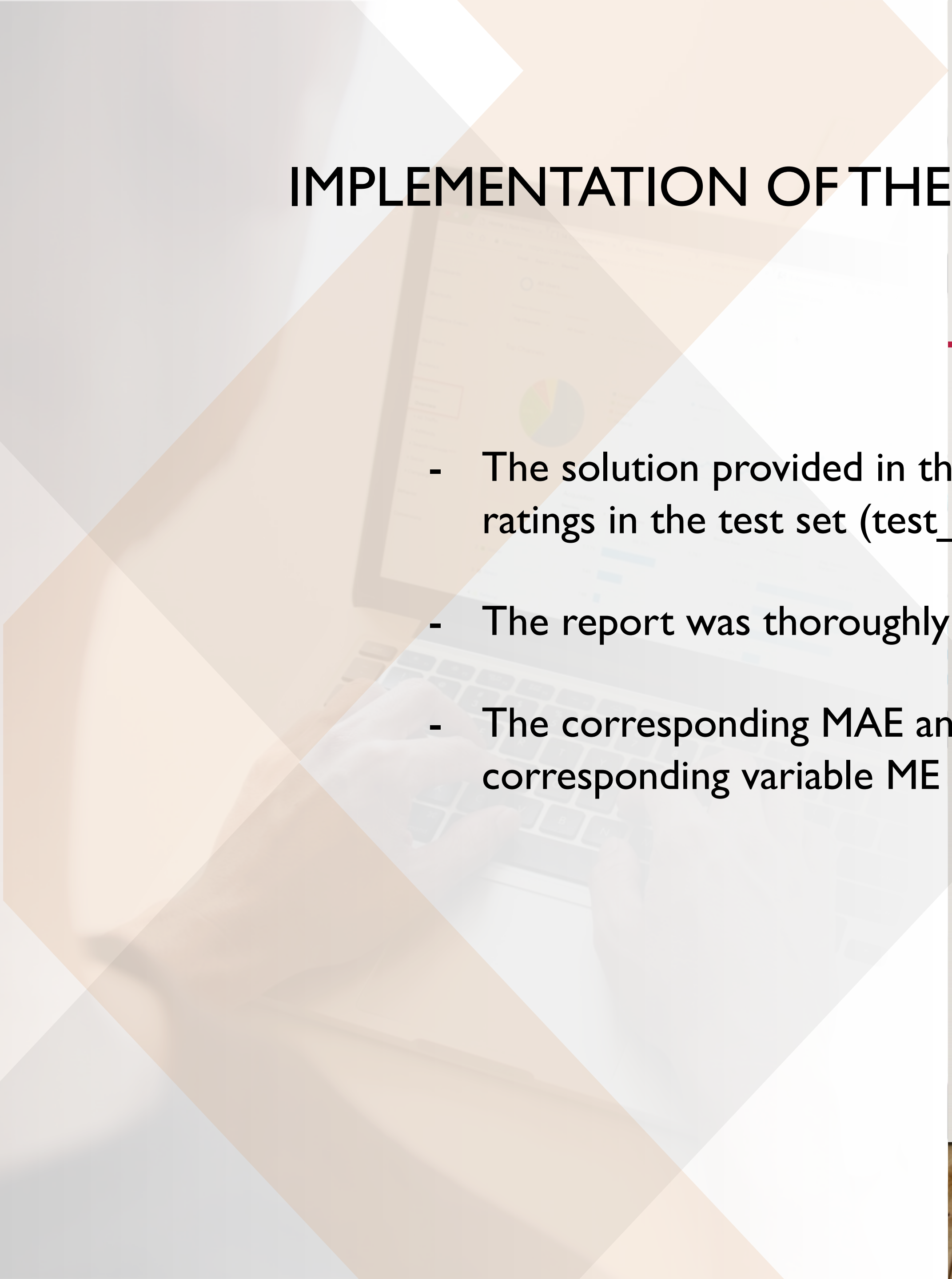


TABLE I
MAE COMPARISONS WITH OTHER METHODS

Training Set	Methods	Given5	Given10	Given15	Given20
ML_100	IPCC	0.892	0.850	0.838	0.826
	UPCC	0.876	0.846	0.831	0.812
	EMDP	0.834	0.826	0.813	0.790
	CFIP	**0.829**	**0.818**	**0.804**	**0.771**
ML_200	IPCC	0.855	0.834	0.821	0.812
	UPCC	0.844	0.822	0.816	0.808
	EMDP	0.833	0.820	0.811	0.781
	CFIP	**0.825**	**0.813**	**0.786**	**0.753**
ML_300	IPCC	0.851	0.829	0.813	0.807
	UPCC	0.840	0.816	0.809	0.800
	EMDP	0.828	0.816	0.802	0.774
	CFIP	**0.797**	**0.778**	**0.757**	**0.719**



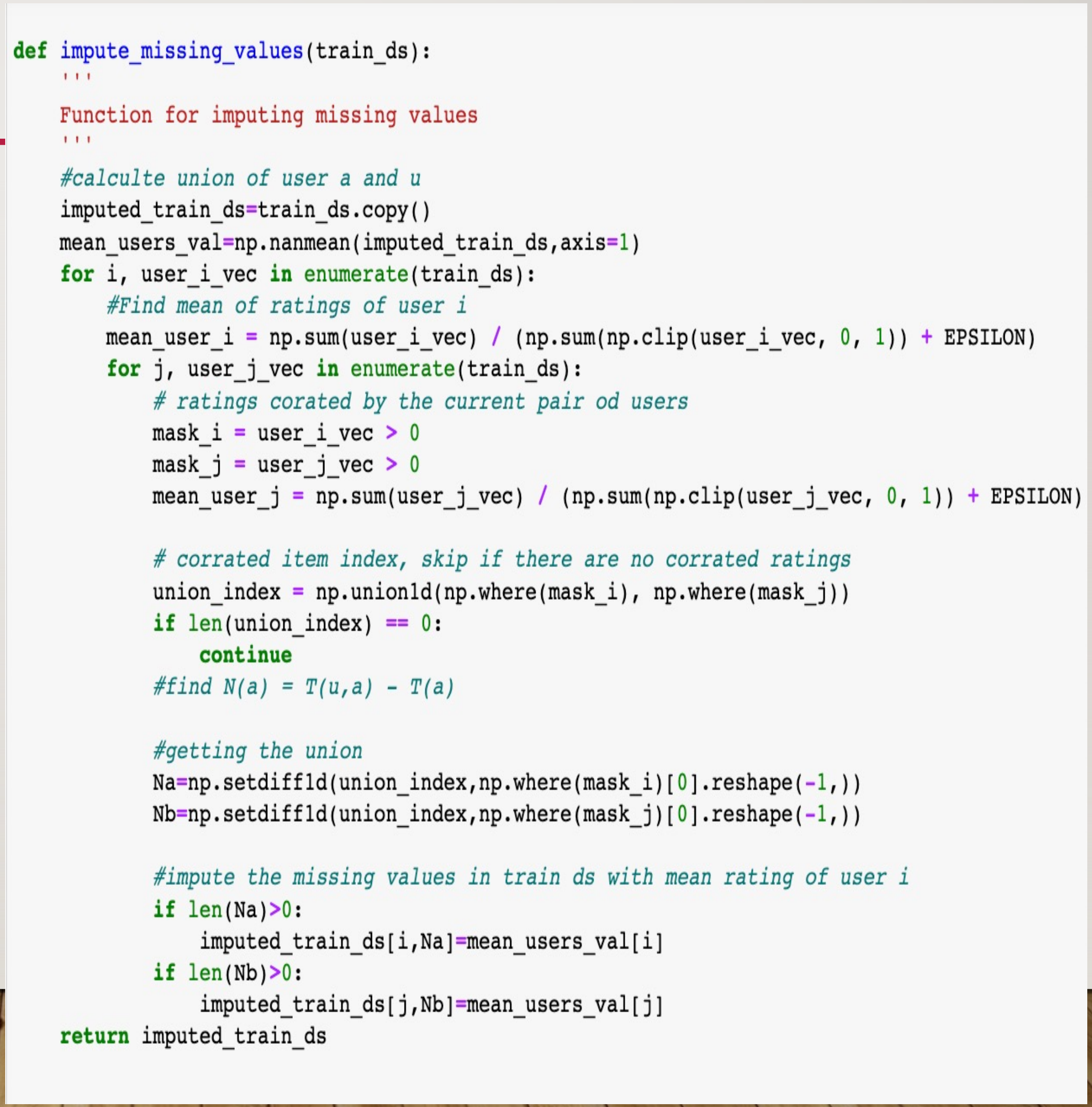
IMPLEMENTATION OF THE NEW APPROACH

- The solution provided in the given report was implemented and evaluated by predicting the ratings in the test set (test_ds).
- The report was thoroughly studied to understand the underlying idea of the proposed solution.
- The corresponding MAE and RMSE of your implementation were stored into the defined corresponding variable ME and RMSE.

IMPLEMENTATION OF THE NEW APPROACH

The missing values were imputed according to the proposed “a novel ratings data making up strategy” in order to address the problem of sparsity

This led to the following calculation being even more well-founded and the result being be more accurate

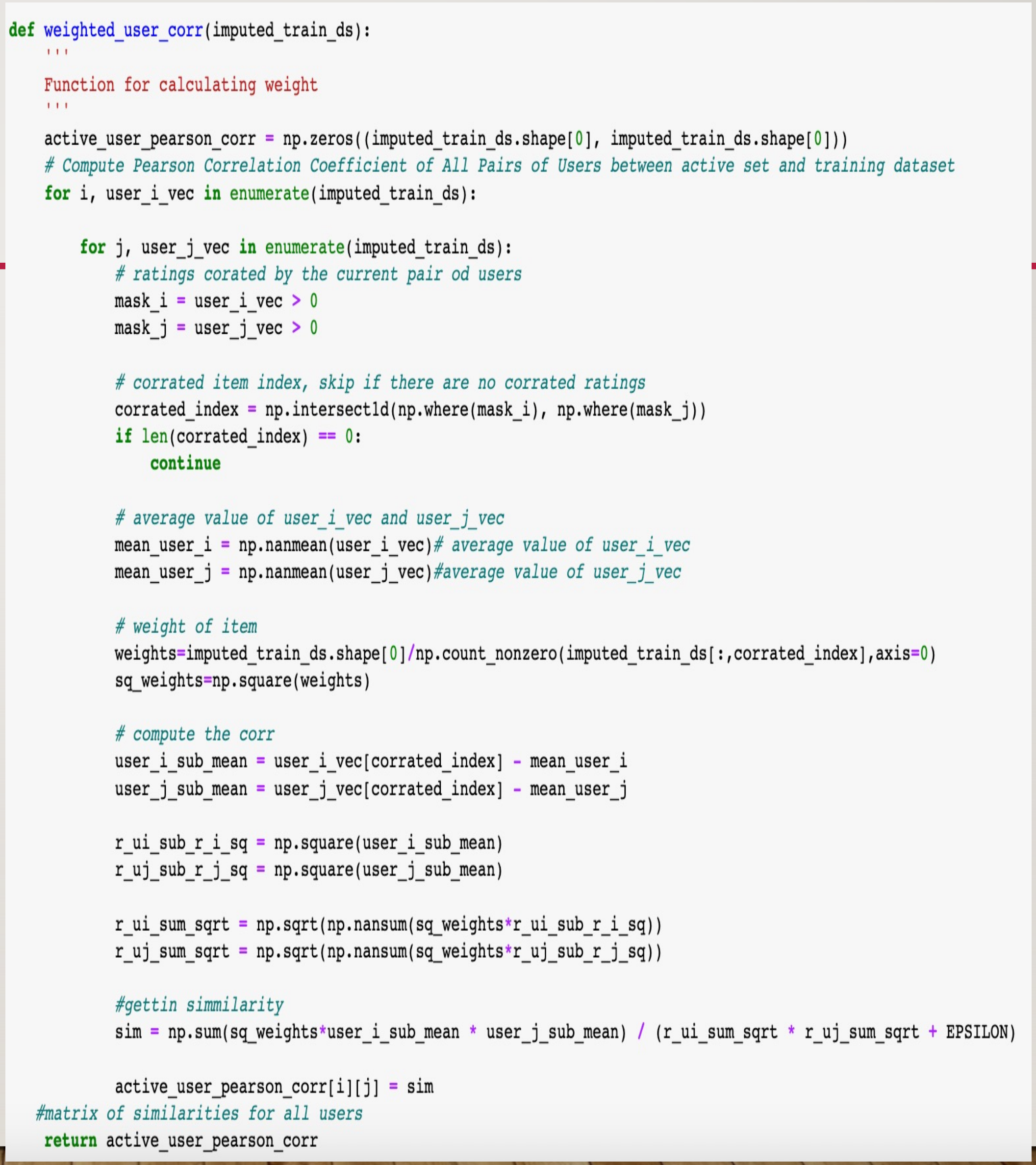


```python
def impute_missing_values(train_ds):
    '''
    Function for imputing missing values
    '''
    #calculte union of user a and u
    imputed_train_ds=train_ds.copy()
    mean_users_val=np.nanmean(imputed_train_ds,axis=1)
    for i, user_i_vec in enumerate(train_ds):
        #Find mean of ratings of user i
        mean_user_i = np.sum(user_i_vec) / (np.sum(np.clip(user_i_vec, 0, 1)) + EPSILON)
        for j, user_j_vec in enumerate(train_ds):
            # ratings corated by the current pair od users
            mask_i = user_i_vec > 0
            mask_j = user_j_vec > 0
            mean_user_j = np.sum(user_j_vec) / (np.sum(np.clip(user_j_vec, 0, 1)) + EPSILON)

            # corrated item index, skip if there are no corrated ratings
            union_index = np.union1d(np.where(mask_i), np.where(mask_j))
            if len(union_index) == 0:
                continue
            #find N(a) = T(u,a) - T(a)

            #getting the union
            Na=np.setdiff1d(union_index,np.where(mask_i)[0].reshape(-1,))
            Nb=np.setdiff1d(union_index,np.where(mask_j)[0].reshape(-1,))

            #impute the missing values in train ds with mean rating of user i
            if len(Na)>0:
                imputed_train_ds[i,Na]=mean_users_val[i]
            if len(Nb)>0:
                imputed_train_ds[j,Nb]=mean_users_val[j]
    return imputed_train_ds
```

ITEM'S POPULARITY WEIGHT

Utilizing the weight computation formula, the weight of the item was calculated

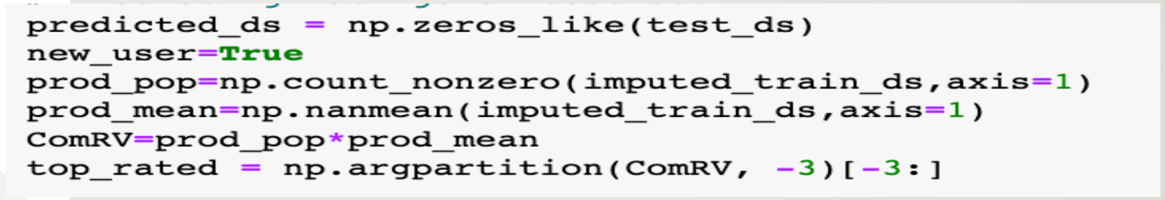
This was useful in capturing similarity for the less common items



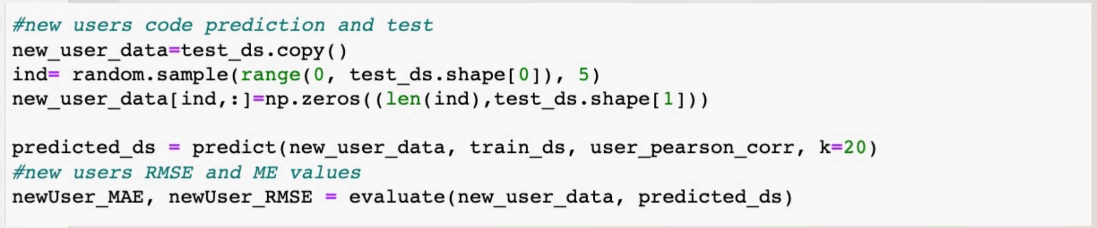
```
def weighted_user_corr(imputed_train_ds):
    '''
    Function for calculating weight
    '''
    active_user_pearson_corr = np.zeros((imputed_train_ds.shape[0], imputed_train_ds.shape[0]))
    # Compute Pearson Correlation Coefficient of All Pairs of Users between active set and training dataset
    for i, user_i_vec in enumerate(imputed_train_ds):

        for j, user_j_vec in enumerate(imputed_train_ds):
            # ratings corated by the current pair od users
            mask_i = user_i_vec > 0
            mask_j = user_j_vec > 0

            # corrated item index, skip if there are no corrated ratings
            corrated_index = np.intersect1d(np.where(mask_i), np.where(mask_j))
            if len(corrated_index) == 0:
                continue

            # average value of user_i_vec and user_j_vec
            mean_user_i = np.nanmean(user_i_vec)# average value of user_i_vec
            mean_user_j = np.nanmean(user_j_vec)#average value of user_j_vec

            # weight of item
            weights=imputed_train_ds.shape[0]/np.count_nonzero(imputed_train_ds[:,corrated_index],axis=0)
            sq_weights=np.square(weights)

            # compute the corr
            user_i_sub_mean = user_i_vec[corrated_index] - mean_user_i
            user_j_sub_mean = user_j_vec[corrated_index] - mean_user_j

            r_ui_sub_r_i_sq = np.square(user_i_sub_mean)
            r_uj_sub_r_j_sq = np.square(user_j_sub_mean)

            r_ui_sum_sqrt = np.sqrt(np.nansum(sq_weights*r_ui_sub_r_i_sq))
            r_uj_sum_sqrt = np.sqrt(np.nansum(sq_weights*r_uj_sub_r_j_sq))

            #gettin simmilarity
            sim = np.sum(sq_weights*user_i_sub_mean * user_j_sub_mean) / (r_ui_sum_sqrt * r_uj_sum_sqrt + EPSILON)

            active_user_pearson_corr[i][j] = sim
    #matrix of similarities for all users
    return active_user_pearson_corr
```

NEW USER PROBLEM

Since it is difficult to recommend items to new users due to the absence of any ratings data, Top-N most popular items that are also rated well were obtained with the ComRV formula.



```
predicted_ds = np.zeros_like(test_ds)
new_user=True
prod_pop=np.count_nonzero(imputed_train_ds,axis=1)
prod_mean=np.nanmean(imputed_train_ds,axis=1)
ComRV=prod_pop*prod_mean
top_rated = np.argpartition(ComRV, -3)[-3:]
```



```
#new users code prediction and test
new_user_data=test_ds.copy()
ind= random.sample(range(0, test_ds.shape[0]), 5)
new_user_data[ind,:]=np.zeros((len(ind),test_ds.shape[1]))

predicted_ds = predict(new_user_data, train_ds, user_pearson_corr, k=20)
#new users RMSE and ME values
newUser_MAE, newUser_RMSE = evaluate(new_user_data, predicted_ds)
```

HOW IT'S SOLVING THE PROBLEM

From a business point of view, recommendation engines can significantly increase revenue and improve CTRs (click-through rate).

Recommendations can easily drive a notable amount of engagement from the users.

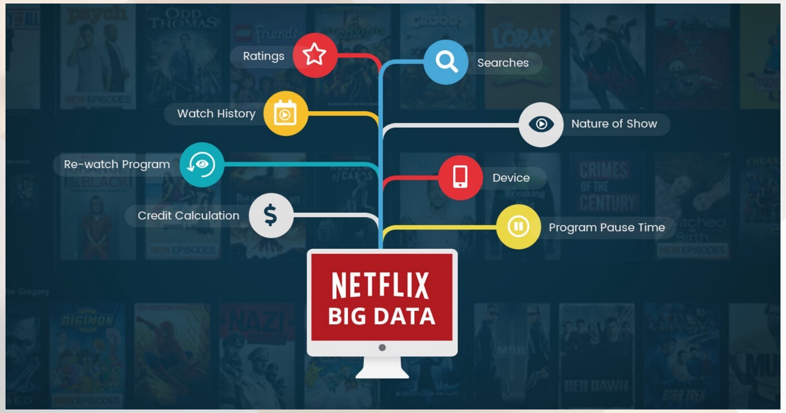


Figure 4: Netflix recommendation system Grover, P. (2017).

Netflix has managed to achieved 80% of its stream time through personalization

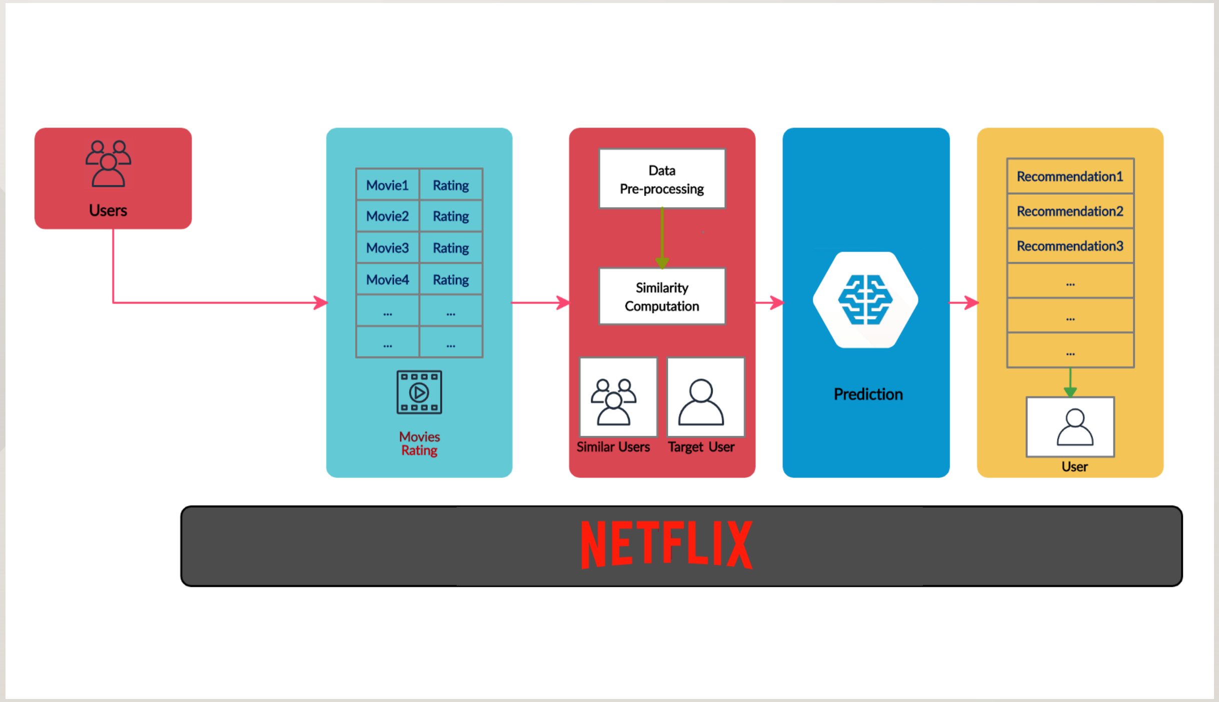


Figure 5 Netflix User-item recommendation system Grover, P. (2017).

REFERENCES

Grover, P. (2017). *Various Implementations of Collaborative Filtering*. [online] Towards Data Science. Available at: https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0.

Python, R. (n.d.). *Build a Recommendation Engine With Collaborative Filtering – Real Python*. [online] realpython.com
Available at: https://realpython.com/build-recommendation-engine-collaborative-filtering/#:~:text=Collaborative%20filtering%20is%20a%20technique.

THANK YOU