Practical Data Science- Assignment 3: Advanced Data Modelling

IDENTIFIERS

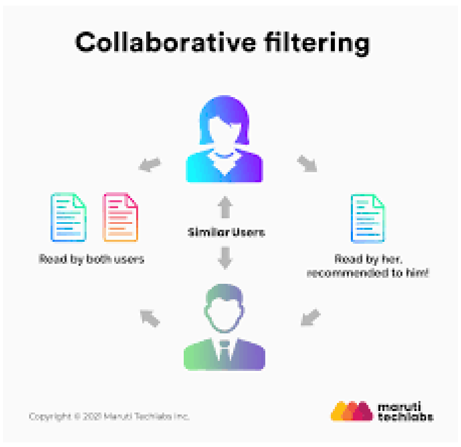
- Anuj Sharma
- Student ID- s3820112

OVERVIEW

The aim of this report was to investigate the process and working of the recommendation engine and explore various python libraries. In the report I have demonstrated three algorithms in the report to achieve this task. With the help of the 'Surprise' library I have chosen KNNBasic,KNNBaseline and KNNWithMeans as my algorithms here for the demonstration of a Beer recommendation engine.

MODEL DESCRIPTION

The Integrated Development Environment (IDE) used for this study is Jupyter Notebook, although the final submission is done as a .py script file. Packages such as pandas, numpy, scikit-learn and Surprise were used to conduct the analysis. The pandas numpy packages were mainly used in Data preparation and Data exploration. The algorithm used here from Surprise library works on Collaborative Filtering.This Collaborative Model works on the similarity index-based technique. For example in the neighbourhood based approach a number of users are selected which resembles our test user and the predictions on behalf of the test user is made by judging and analysing the choices of similar users on the subject.


Collaborative filtering
Read by both users
Similar Users
Read by her, recommended to him!
Copyright © 2021 Maruti Techlabs Inc.
maruti techlabs

My 3 Algorithms (all KNN Based) today, that displays this feature are-

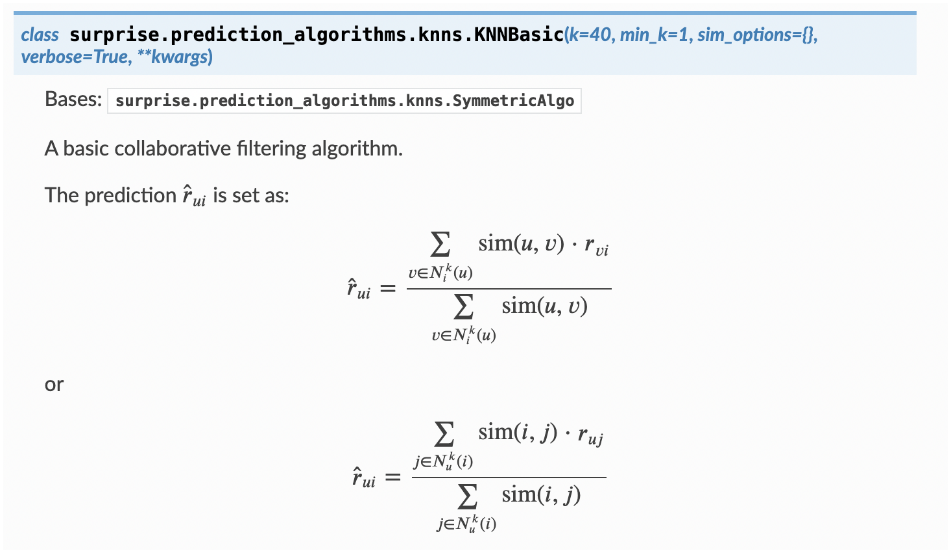
1. KNNBasic
   basic collaborative filtering algorithm.
2. KNNBaseline
   basic collaborative filtering algorithm and also takes account of Baseline rating
3. KNNWithMeans
   basic collaborative filtering algorithm and also takes account of mean rating

KNN stands for K-nearest neighbours is an algorithm designed for clustering similar/nearest neighbours of the test user where k can be any integral value denoting the radius in which we had to find the similar users.
The main module parameters were sim options as {'name': 'pearson_baseline', 'user_based': False}
user based parameter was set to false to explore item based collaborative filtering.

EXPERIMENTS

The data analysis started with cleaning the data. I used the pandas library to import and visualize the dataset in the form of a data set. After cleaning the dataset and ensuring there are no wrong or null values entered in the dataset. We removed all unnecessary data columns and then left with our final dataset of 3 columns containing "RowID", "ReviewerID", "Label". Then split the dataset into a training set and testing set in order to train my model.

Algorithm1- KNNBasic



class **surprise.prediction_algorithms.knns.KNNBasic**(*k=40, min_k=1, sim_options={}, verbose=True, **kwargs*)

Bases: `surprise.prediction_algorithms.knns.SymmetricAlgo`

A basic collaborative filtering algorithm.

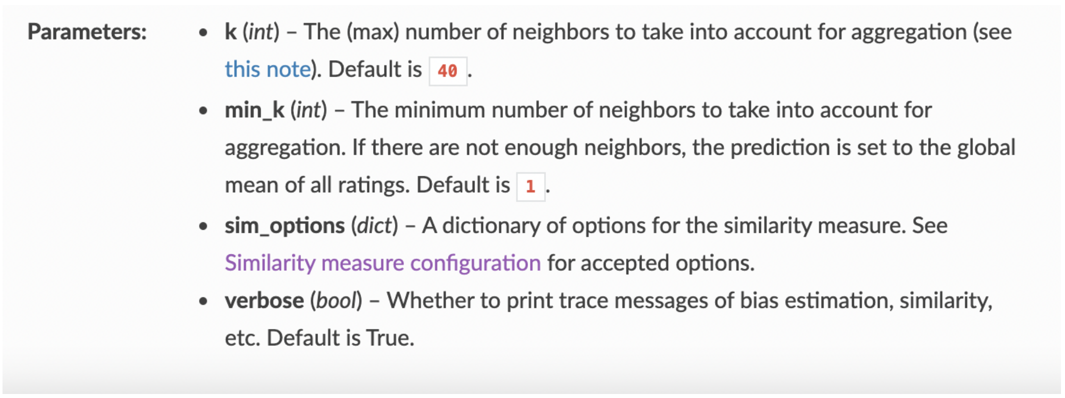
The prediction $\hat{r}_{ui}$ is set as:

$$\hat{r}_{ui} = \frac{\sum\limits_{v \in N_i^k(u)} \text{sim}(u, v) \cdot r_{vi}}{\sum\limits_{v \in N_i^k(u)} \text{sim}(u, v)}$$

or

$$\hat{r}_{ui} = \frac{\sum\limits_{j \in N_u^k(i)} \text{sim}(i, j) \cdot r_{uj}}{\sum\limits_{j \in N_u^k(i)} \text{sim}(i, j)}$$

KNNBasic algorithm is the most basic algorithm for the display of my collaborative model. The parameter description is provided in the image below and the algorithm parameters used in the assignment were
sim_options = {'name': 'pearson_baseline', 'user_based': False}
algo = KNNBasic(sim_options=sim_options)

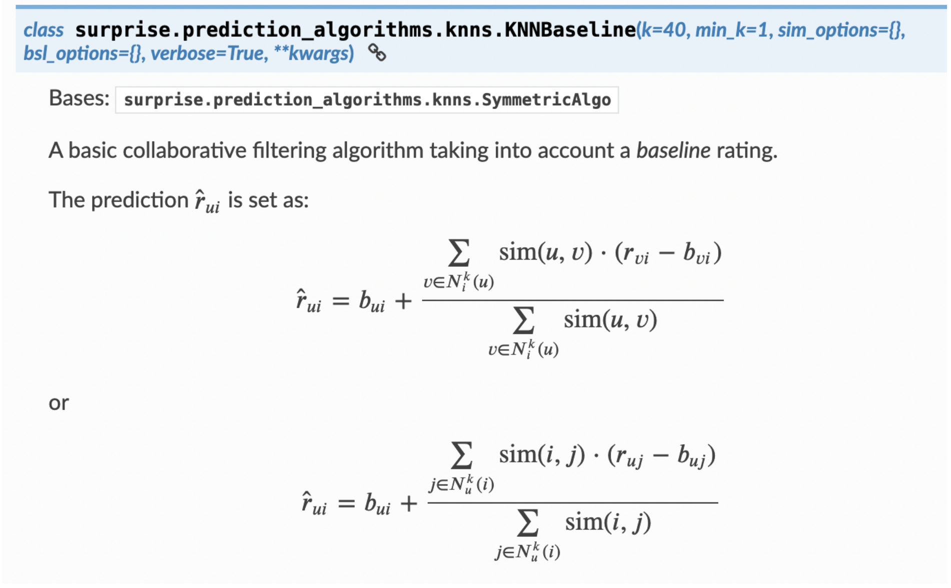


Parameters:
- **k** (*int*) – The (max) number of neighbors to take into account for aggregation (see this note). Default is `40`.
- **min_k** (*int*) – The minimum number of neighbors to take into account for aggregation. If there are not enough neighbors, the prediction is set to the global mean of all ratings. Default is `1`.
- **sim_options** (*dict*) – A dictionary of options for the similarity measure. See Similarity measure configuration for accepted options.
- **verbose** (*bool*) – Whether to print trace messages of bias estimation, similarity, etc. Default is True.

I used cross validation on the algorithm since the library provides the features to ensure the training is done with minimum 'MAE' (Mean Absolute Error).

Training Results {'**test_mae**': array([0.54014763, 0.5408268 , 0.53945904]),
'**fit_time**': (4.698112964630127, 4.9257590770721436, 4.8337321281433105),
'**test_time**': (1.5645389556884766, 1.4001169204711914,
1.534059762954712)}

Algorithm2- KNNBaseline



class **surprise.prediction_algorithms.knns.KNNBaseline**(*k=40, min_k=1, sim_options={}, bsl_options={}, verbose=True, **kwargs*)

Bases: `surprise.prediction_algorithms.knns.SymmetricAlgo`

A basic collaborative filtering algorithm taking into account a *baseline* rating.

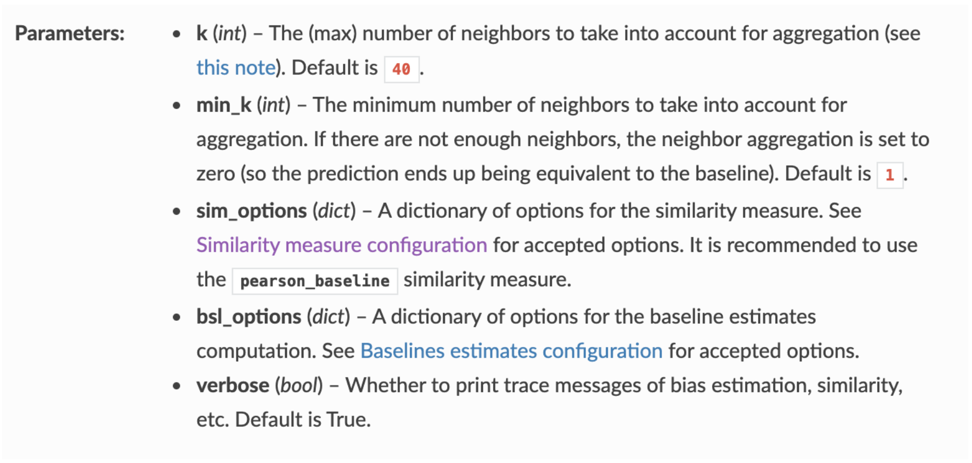
The prediction $\hat{r}_{ui}$ is set as:

$$\hat{r}_{ui} = b_{ui} + \frac{\sum\limits_{v \in N_i^k(u)} \text{sim}(u, v) \cdot (r_{vi} - b_{vi})}{\sum\limits_{v \in N_i^k(u)} \text{sim}(u, v)}$$

or

$$\hat{r}_{ui} = b_{ui} + \frac{\sum\limits_{j \in N_u^k(i)} \text{sim}(i, j) \cdot (r_{uj} - b_{uj})}{\sum\limits_{j \in N_u^k(i)} \text{sim}(i, j)}$$

KNNBaseline algorithm is somehow similar to the basic in terms of collaborative filtering but it also takes account of the baseline rating of the user the parameters used here were {'name': 'pearson_baseline', 'user_based': False}

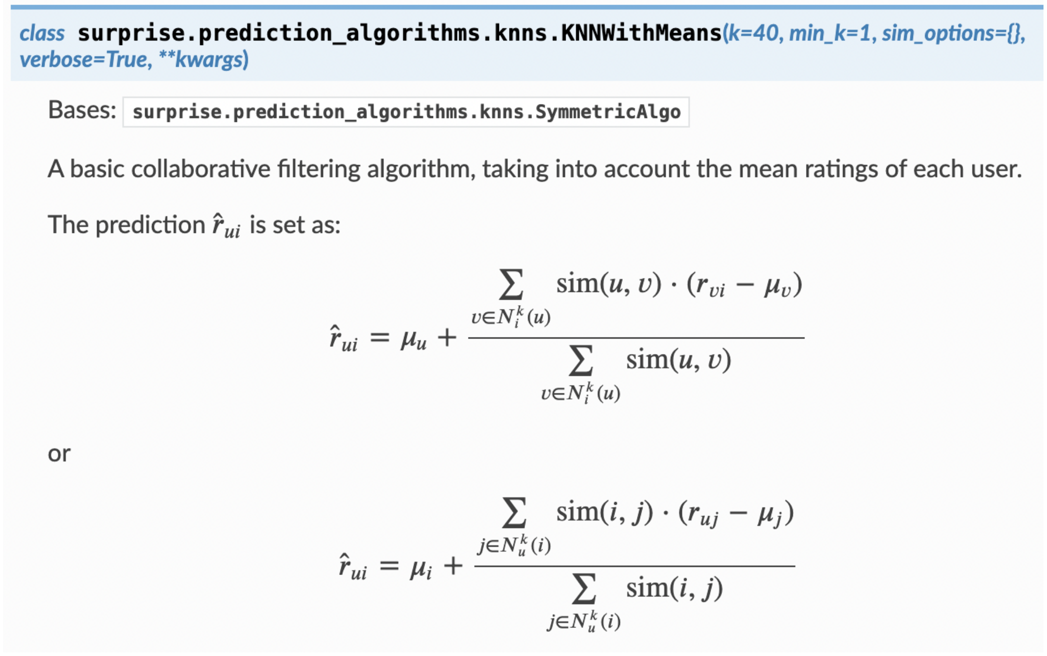


Parameters:	• **k** (*int*) – The (max) number of neighbors to take into account for aggregation (see this note). Default is `40`. • **min_k** (*int*) – The minimum number of neighbors to take into account for aggregation. If there are not enough neighbors, the neighbor aggregation is set to zero (so the prediction ends up being equivalent to the baseline). Default is `1`. • **sim_options** (*dict*) – A dictionary of options for the similarity measure. See Similarity measure configuration for accepted options. It is recommended to use the `pearson_baseline` similarity measure. • **bsl_options** (*dict*) – A dictionary of options for the baseline estimates computation. See Baselines estimates configuration for accepted options. • **verbose** (*bool*) – Whether to print trace messages of bias estimation, similarity, etc. Default is True.

I used crossed validation here to with parameters as (algo1, data, measures=['MAE'], cv=3) since algorithms are judged on the basis of the mean absolute error in the predictions

Training results-{'**test_mae**': array([0.5120793 , 0.51205332, 0.51250819]),
'**fit_time**': (4.870250940322876, 5.19868278503418, 5.044945955276489),
'**test_time**': (1.6627602577209473, 1.2265021800994873,
1.2797818183898926)}

Algorithm3- KNNWithMeans



class **surprise.prediction_algorithms.knns.KNNWithMeans**(*k=40, min_k=1, sim_options={}, verbose=True, **kwargs*)

Bases: `surprise.prediction_algorithms.knns.SymmetricAlgo`

A basic collaborative filtering algorithm, taking into account the mean ratings of each user.

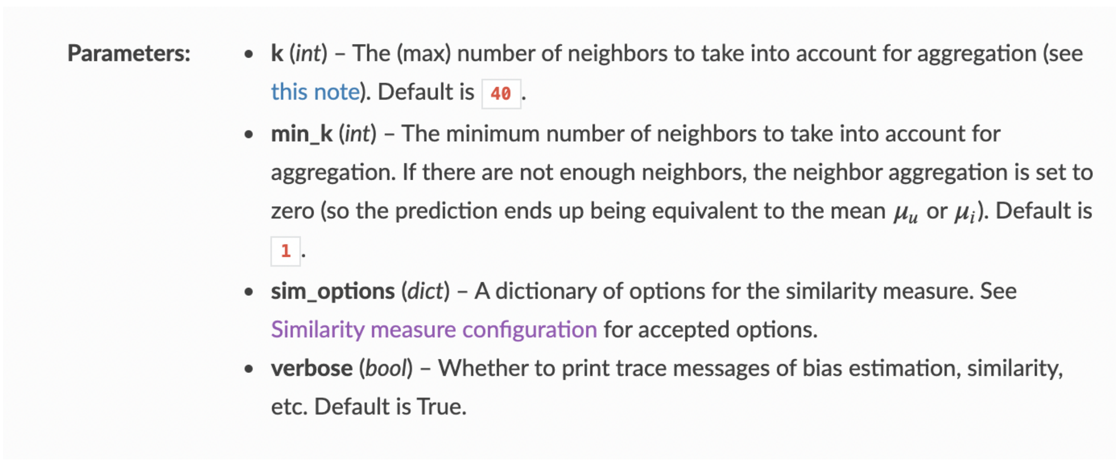
The prediction $\hat{r}_{ui}$ is set as:

$$\hat{r}_{ui} = \mu_u + \frac{\sum\limits_{v \in N_i^k(u)} \text{sim}(u, v) \cdot (r_{vi} - \mu_v)}{\sum\limits_{v \in N_i^k(u)} \text{sim}(u, v)}$$

or

$$\hat{r}_{ui} = \mu_i + \frac{\sum\limits_{j \in N_u^k(i)} \text{sim}(i, j) \cdot (r_{uj} - \mu_j)}{\sum\limits_{j \in N_u^k(i)} \text{sim}(i, j)}$$

The Third and my main algorithm is KNNWithMeans which includes the mean rating of all the users. The parameters used here are
sim_options = {'name': 'pearson_baseline', 'user_based': False}
k=40
min_k=1
verbose=true



Parameters:

- **k** (*int*) – The (max) number of neighbors to take into account for aggregation (see this note). Default is `40`.
- **min_k** (*int*) – The minimum number of neighbors to take into account for aggregation. If there are not enough neighbors, the neighbor aggregation is set to zero (so the prediction ends up being equivalent to the mean μ_u or μ_i). Default is `1`.
- **sim_options** (*dict*) – A dictionary of options for the similarity measure. See Similarity measure configuration for accepted options.
- **verbose** (*bool*) – Whether to print trace messages of bias estimation, similarity, etc. Default is True.

I used cross fold validation to tune in my training of dataset in accordance with least mean absolute error possible.

```
Training results-{'test_mae': array([0.54070731, 0.54034027, 0.53938591]),
                  'fit_time': (5.197546005249023, 5.300782203674316, 5.381285190582275),
                 'test_time': (1.6370491981506348, 1.1045310497283936,
                 1.621837854385376)}
```

LIBRARIES USED

1. Numpy- 1.21.2
2. Pandas- 1.3.3
3. Scikit-learn- 1.0
4. Surprise- 1.1.1

REFERENCES (if any)

1. Hug, N. (n.d.). *K-NN inspired algorithms* . k-NN inspired algorithms - Surprise 1 documentation. Retrieved November 1, 2021, from https://surprise.readthedocs.io/en/stable/knn_inspired.html
2. Hug, N. (n.d.). *Home*. Surprise. Retrieved November 1, 2021, from http://surpriselib.com/
3. Kurama, V. (n.d.). *A simple introduction to collaborative filtering*. Built In. Retrieved November 1, 2021, from https://builtin.com/data-science/collaborative-filtering-recommender-system
4. Li, S. (2019, September 26). *Building and testing recommender systems with surprise, step-by-step*. Medium. Retrieved November 2, 2021, from https://towardsdatascience.com/building-and-testing-recommender-systems-with-surprise-step-by-step-d4ba702ef80b