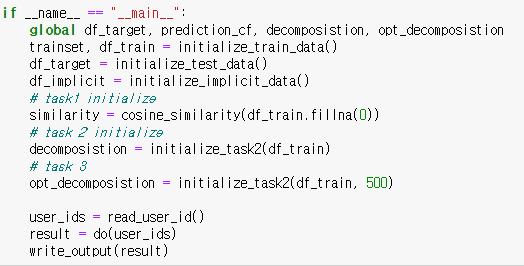
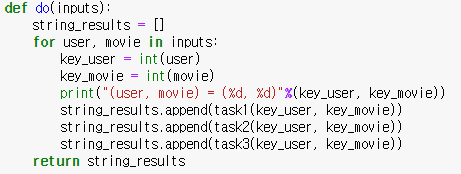
**0 main() / do() / data 초기화 / RSME**



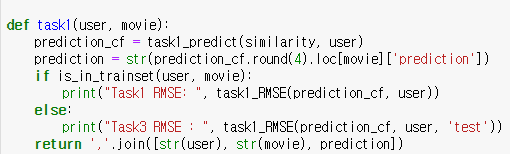
Trainset, Testset, task1의 similarity, task2(3)의 matrix factorization처럼 여러 번 사용되는 데이터들은 main에서 초기화해서 실행시간을 줄이고자 하였다.

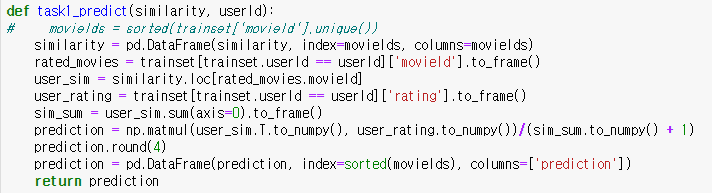


저번 과제와 동일한 구조로, main()으로부터 호출된 do()는 user와 movie에 대한 정보를 받아, task 1,2,3을 각각 실행한다.

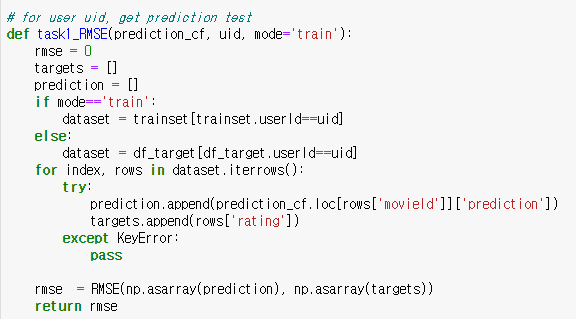


RMSE는 numpy를 사용해서 측정한다.

**1 Item-based Collaborative Filtering** 



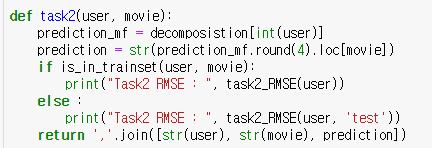
사용자가 평점을 남긴 영화들과 비슷한 영화들을 중심으로 similarity표를 통해서 예측되는 값을 유추한다.



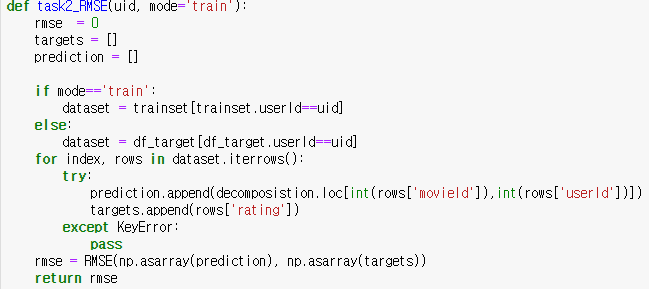
예측한 점수가 얼마나 정확한지 평가하기 위해 RMSE 를 사용한다.

과제의 경우에는 trainset에 있는 데이터로 RSME를 검증하지만, 모델이 얼마나 정확한지 확인하기 위해서 옵션으로 [‘train’, ‘test’]를 둬서 측정할 수 있도록 했다.

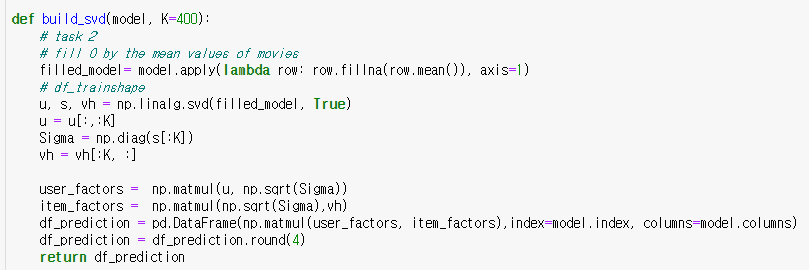
**2 Matrix Factorization**



미리 계산해 decomposition표 (matrix factorization)에서 해당 user와 movie에 대한 좌표를 찾으면 predict값을 받을 수 있다.



Task1과 마찬가지로 RMSE 계산에 option을 두었다.



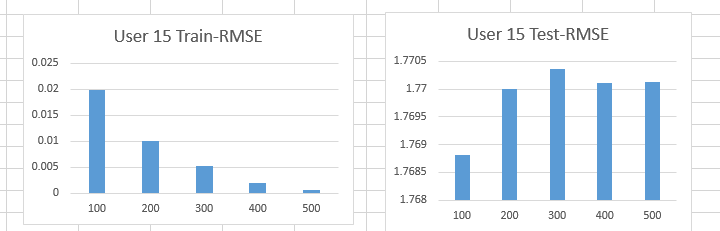
K의 기본 값을 400으로 셋팅하여 decomposed 된 matrix을 찾는 과정이다. ‘numpy.linalg.svd’을 사용해서 u, sigma, v까지는 구할 수 있지만, K개에 맞춰서 자르는 과정이 별도로 필요했다.

SVD 모델을 생성하는 단계로 factor의 사이즈에 맞게 User factor와 item factor를 자르는 작업이 익숙하지 않아 애를 먹었다. 이 과정에서 dot 연산과 @연산이 동일한 연산임을 배웠다.

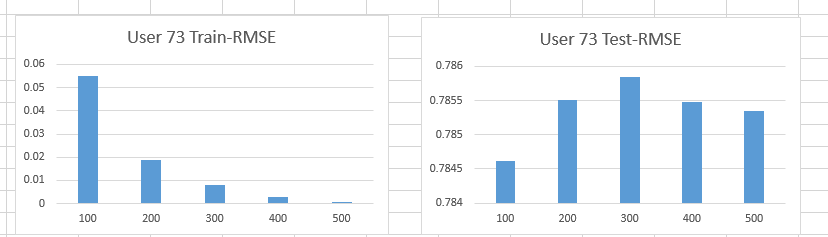
**3 모델 최적화  
[실험]**

2번 SVD 모델을 최적화하기 위해서 RMSE가 가장 작은 K를 찾았다. 기본적인 구조는 2번과 동일하며, 정확도를 위해서 test dataset에서 RMSE를 metric으로 사용해서 정확도를 측정했다. overfitting이 발생할 수 있기 때문에 방지하기 위해 최대 2 case에서 값을 찾았다.

User 15에 대해 train, test data로 측정한 결과는 아래와 같다.

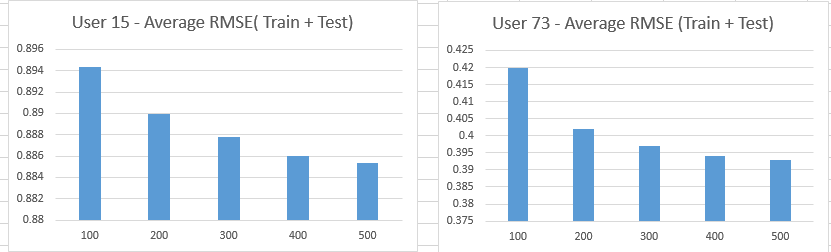


User 73에 대해 train, test data로 측정한 결과는 아래와 같다.

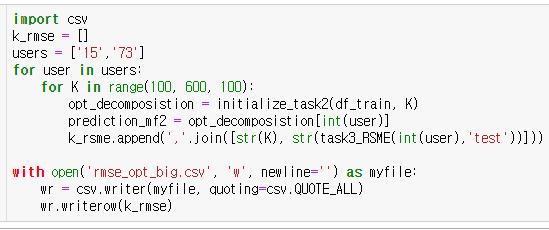


**두 경우에 대해 overfitting이 되는 것을 방지하기 위해**

**Train과 test data의 RMSE 평균이 가장 작은 500을 선택했다. K = 500**



**실험을 위한 코드는 아래와 같다.**



실제 모델 구현을 할 때는 K로 decomposed 된 matrix을 생성했다. (이후 구조는 2번과 동일 하기 때문에 생략)



**4 input의 RMSE 값**

Train data set을 사용해서 rmse를 구하면 다음과 같다.

