**추천 시스템**

**-Latent Factor model-**

**결과 보고서**

202012252 지수영

1. **데이터**
2. **Train data (80664 rows x 6 columns)**

* userId : 610개
* title(영화) : 8949개

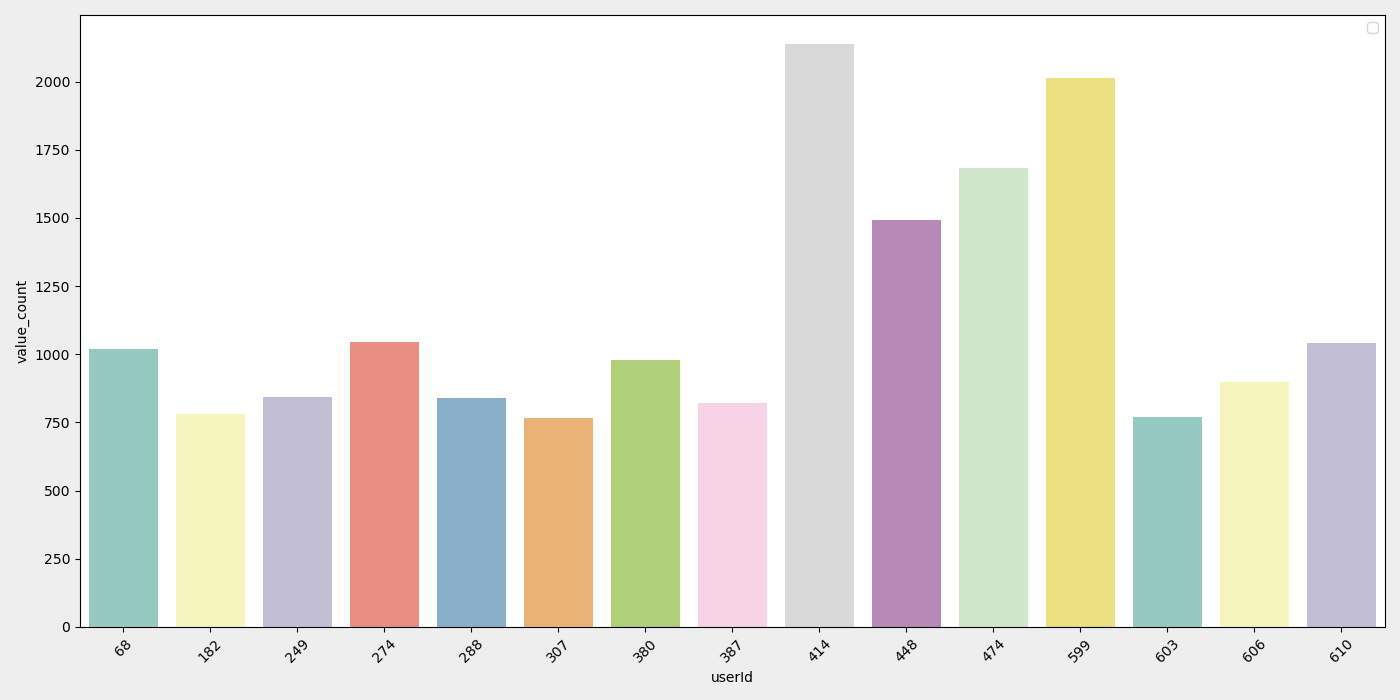
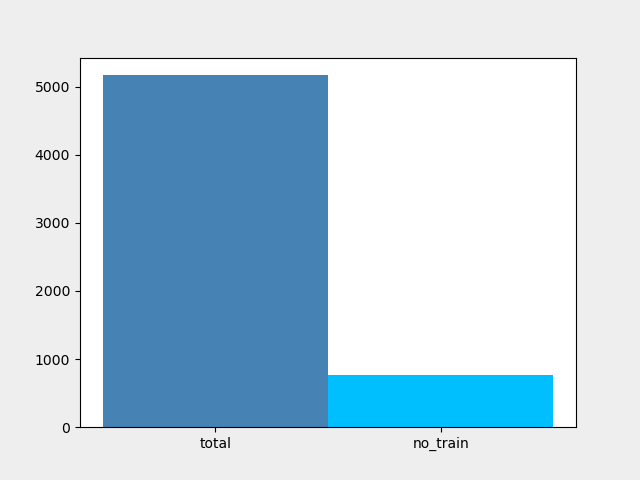


그림 1 ‘userId’와 평점이 있는 데이터 개수.

1. **Test data (20166 rows x 6 columns)**

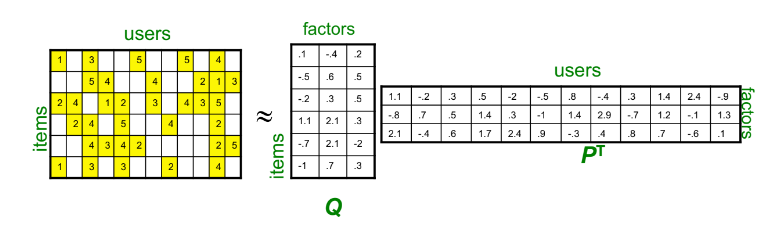
* userId : 610개
* title(영화) : 5168개 (train 데이터에 없는 영화 개수 : 770개)



1. **latent factor model**

* 사용자와 아이템을 잠재적인 차원(Factor)들을 이용해 나타낼 수 있다고 보는 모델
* Matrix Factorization을 이용해 구현한다.
* **Matrix Factorizaiton**

사용자(user)와 아이템(movie)을 같은 차원의 벡터들의 집합(Matrix)으로 표현한다.



Q[i]와 P[x]의 곱(yhat)과 R(items x users, y)행렬의 오차를 비교해 최소가 되는 값을 찾는다.

이를 반복하며 오차가 최소가 되는 Q, P를 찾는다.

R matrix의 빈 공간에 학습을 통해 만들어진 Q와 P의 곱을 넣는다.

* **경사하강법(Gradient Descent) - SGD**

오차를 계산하고 오차가 최소화 되도록 P, Q 값을 계속해서 업데이트 한다.

1. **목표**

* **RMSE 1.1 미만**
* Rmse : root mean square error(제곱한 에러의 평균)

1. **학습**

**Train**

1. Train 데이터 값을 pivot table로 변경해, 사용자별로 영화에 매긴 평점을 알아본다. (아래 표와 같이 null 값이 많다. 이 null 값은 matrix factorization을 통해 채워 넣을 것이다.)

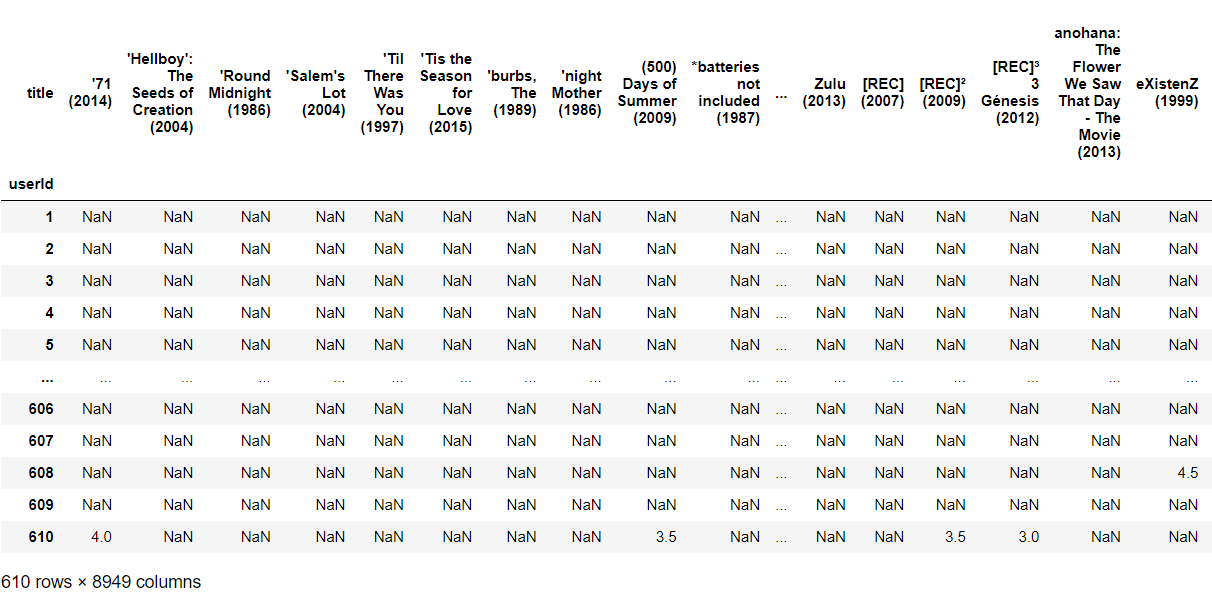
****

그림 2 피봇 테이블(train data)

1. 랜덤한 값을 가진 P와 Q행렬을 생성한다. (경사 하강법으로 오차를 줄여 나갈 것이기 때문에 아무 값이나 넣어준다.)
2. 경사하강법을 적용하여 값을 갱신하는 과정을 반복한다. (MSE값을 출력해 학습횟수에 따라 오차가 얼마나 줄어들었는 지 확인한다.)

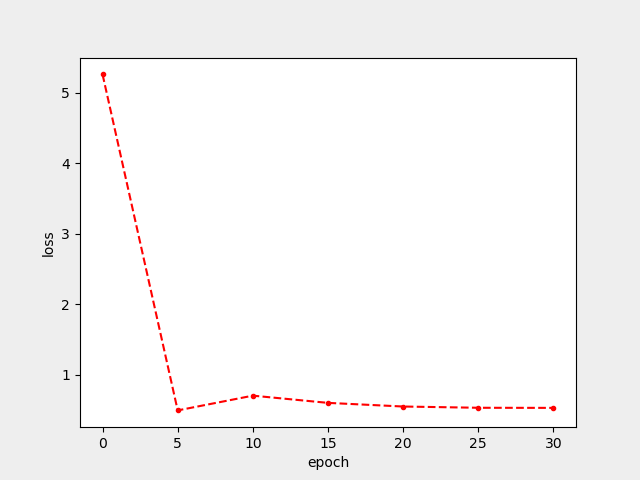
****

그림 3 epoch\_loss 그래프

**학습을 진행하면서 loss 값(mse)이 계속 줄어들고 있는 것을 확인할 수 있다.**

1. 최종적으로 갱신된 P, Q 행렬을 곱해서 ***y\_hat(R)***이라는 새로운 행렬을 만든다. Rating 값은 1~5점까지 0.5점 단위로 분포되어 있기 때문에 ***y\_hat(R)***값을 이 범위에 맞춘다. (X.3 ~ X.7 사이의 값은 X.5로 scaling하고 그 외의 값들은 반올림으로 처리했다. → 전체 행렬에 0.2를 더한 값을 0.5로 나누었을 때의 몫을 다시 0.5로 곱한다.)

Ex) **① 3.7 -> 3.7 + 0.2 = 3.9, 3.9//0.5 = 7, 7\*0.5 = 3.5**

**② 2.4 -> (2.4 + 0.2)//0.5 = 5, 5\*0.5 = 2.5**

**Test**

1. Test 데이터를 하나씩 읽어들이며 해당 영화가 Train 데이터에 있는 지 확인한다. 해당 영화의 평점이 ***y\_hat(R)***에 **존재하면 해당 rating값을 추가**하고, **없다면 사용자(userId)의 평균 평점을 추가**한다. (평균 평점을 넣은 것은 한 사람이 평점을 매길 때 비슷한 값으로 매긴다고 가정했기 때문이다.)
2. **결과**
3. **Rating**

* Submission을 제출했을 때 RMSE 값은 1.04383으로 나왔으며 목표했던 RMSE 1.1 미만의 값이 나왔다.
* Epoch를 100으로 설정하고 코드를 실행했을 때 loss 값은 떨어졌지만, Kaggle에 제출했을 때의 RMSE 값은 1.09로 epoch를 30번 반복했을 때 보다 더 높게 나왔다.

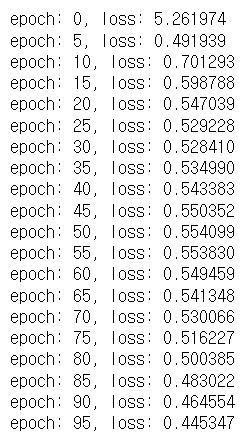


그림 4 epoch 100

* Epoch가 커질수록 loss값은 감소했지만 test데이터에 적용했을 때 성능이 더 안 좋아진 것으로 보아 ***과적합*** 문제가 발생했음을 알 수 있다.

1. **추천**

* Train 데이터의 전체 영화 정보를 정리해 movie\_info라는 데이터 프레임을 생성한다.

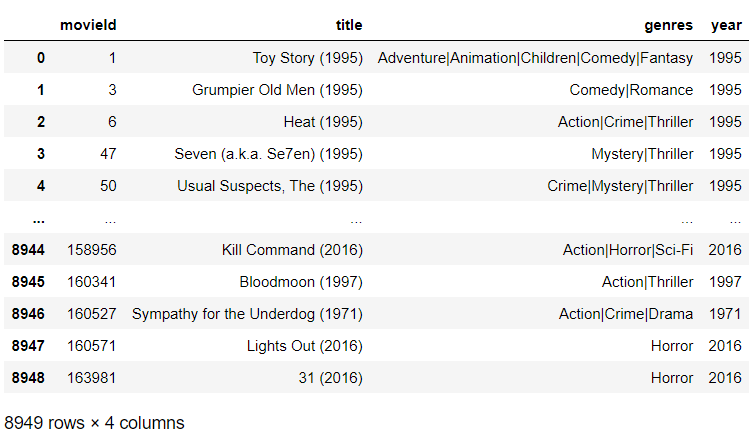


그림 5 영화 정보

* userId를 입력 받으면 해당 userId가 평점을 매긴 영화를 제외하고, 평점을 매기지 않은 영화 중에서 예측 평점이 높은 순으로 10개를 출력한다.

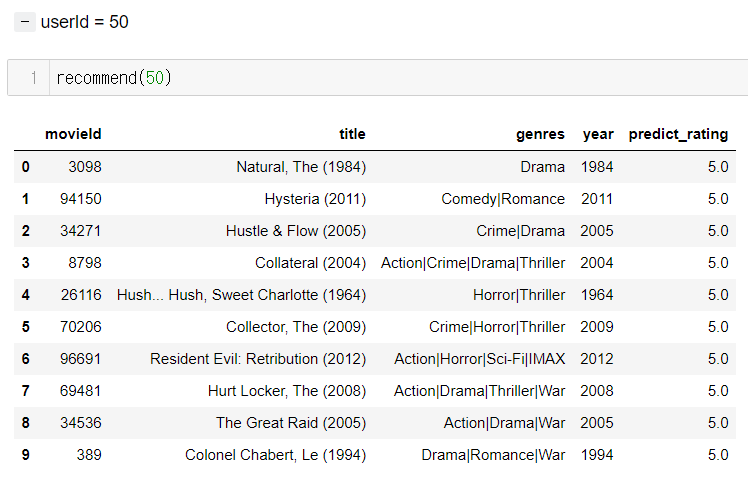


그림 6 userId = 50 예측 평점 상위 10개 영화 출력