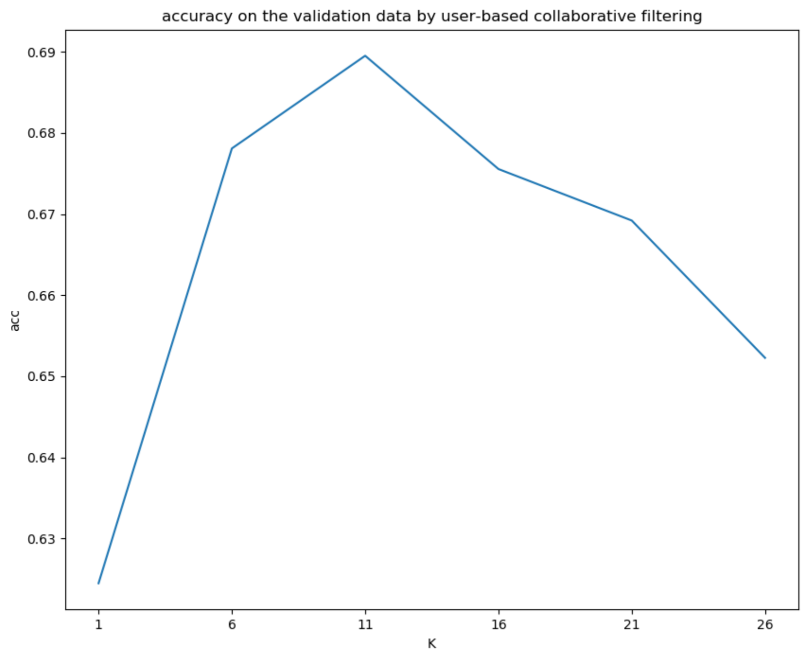
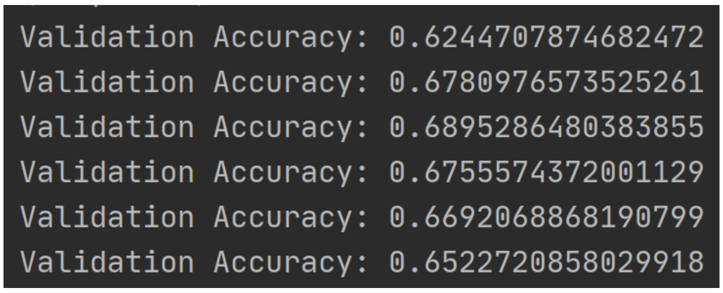
1 a

plot and report the accuracy on the validation data.





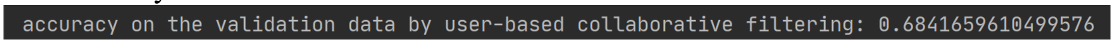
```
Validation Accuracy: 0.6244707874682472
Validation Accuracy: 0.6780976573525261
Validation Accuracy: 0.6895286480383855
Validation Accuracy: 0.6755574372001129
Validation Accuracy: 0.6692068868190799
Validation Accuracy: 0.6522720858029918
```

1 b

Report the chosen k $*$ and the final test accuracy

K = 11

test accuracy:

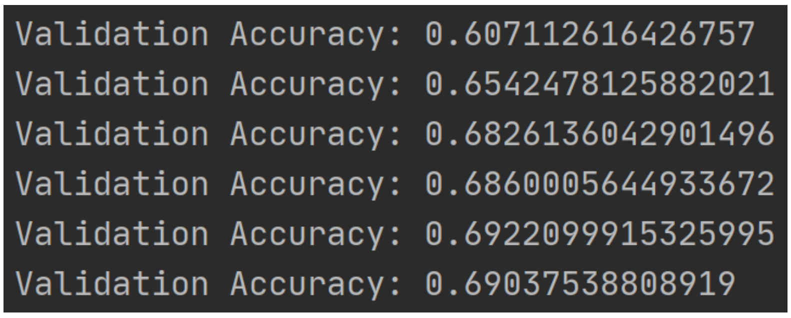


```
accuracy on the validation data by user-based collaborative filtering: 0.6841659610499576
```

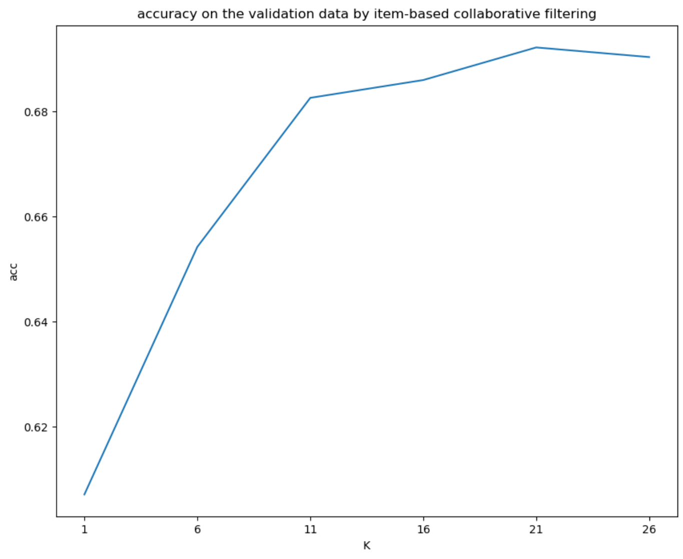
1 c

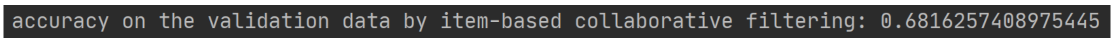
State the underlying assumption on item-based collaborative filtering.

Answer: Assuming that each user's interests are limited to a few aspects, so if two items belong to one user's interest list, then these two items may belong to a limited number of areas, and if two items belong to many users' interests List, then they may belong to the same field and therefore have a great degree of similarity.



```
Validation Accuracy: 0.607112616426757
Validation Accuracy: 0.6542478125882021
Validation Accuracy: 0.6826136042901496
Validation Accuracy: 0.6860005644933672
Validation Accuracy: 0.6922099915325995
Validation Accuracy: 0.69037538808919
```





```
accuracy on the validation data by item-based collaborative filtering: 0.6816257408975445
```

1 d

Answer: user-based collaborative filtering performs better

1 e

Answer: 1. When the data becomes very large, the matrix calculation is very expensive

2. For example, sometimes item A has been seen by many users, so recommendation based on item A may be inaccurate. Instead, recommendations should be made based on the special concerns of some users.

2a log-likelihood log

$$\mathrm{L} = \prod p\left(C \middle| \theta, \beta\right)$$

$$L = \prod_{i,j} p_{i,j}^{C_{i,j}} \left(1 - p_{i,j}\right)^{1-C_{i,j}}$$

$$-ln(L) = -\left(\sum_{i,j} C_{i,j} ln(p_{i,j}) + \left(1 - C_{i,j}\right) ln\left(1 - p_{i,j}\right) \right)$$

$$\text{Where } p_{i,j} = \frac{exp\left(\theta_i - \beta_j\right)}{1 + exp\left(\theta_i - \beta_j\right)}$$

the derivative of the log-likelihood

$$\frac{\partial\left(-ln(L)\right)}{\theta} = -\sum_{i,j} \left(\frac{C_{i,j}}{p_{i,j}} - \frac{1 - C_{i,j}}{1 - p_{i,j}}\right) \frac{\partial\left(p_{i,j}\right)}{\theta}$$

$$= -\sum_{i,j} \left(\frac{C_{i,j}}{p_{i,j}} + \frac{1 - C_{i,j}}{1 - p_{i,j}}\right)\left(p_{i,j}\right)\left(1 - \left(p_{i,j}\right)\right)$$

$$= -\sum_{i,j} C_{i,j} \left(1 - p_{i,j}\right) - (1 - C_{i,j}) p_{i,j}$$

$$= -\sum_{i,j} C_{i,j} - p_{i,j}$$

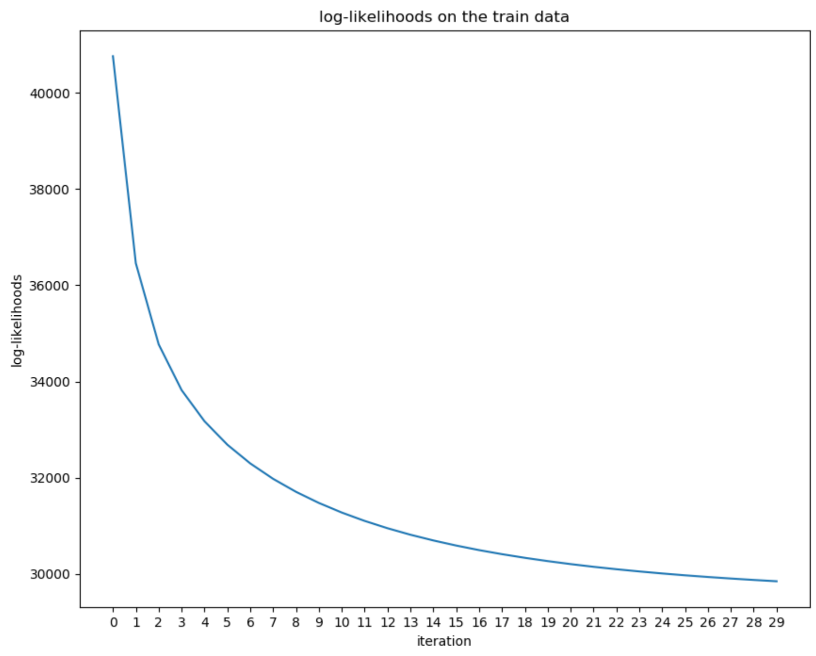
$$\frac{\partial\left(-ln(L)\right)}{\beta} = -\sum_{i,j} \left(\frac{C_{i,j}}{p_{i,j}} - \frac{1 - C_{i,j}}{1 - p_{i,j}}\right) \frac{\partial\left(p_{i,j}\right)}{\beta}$$

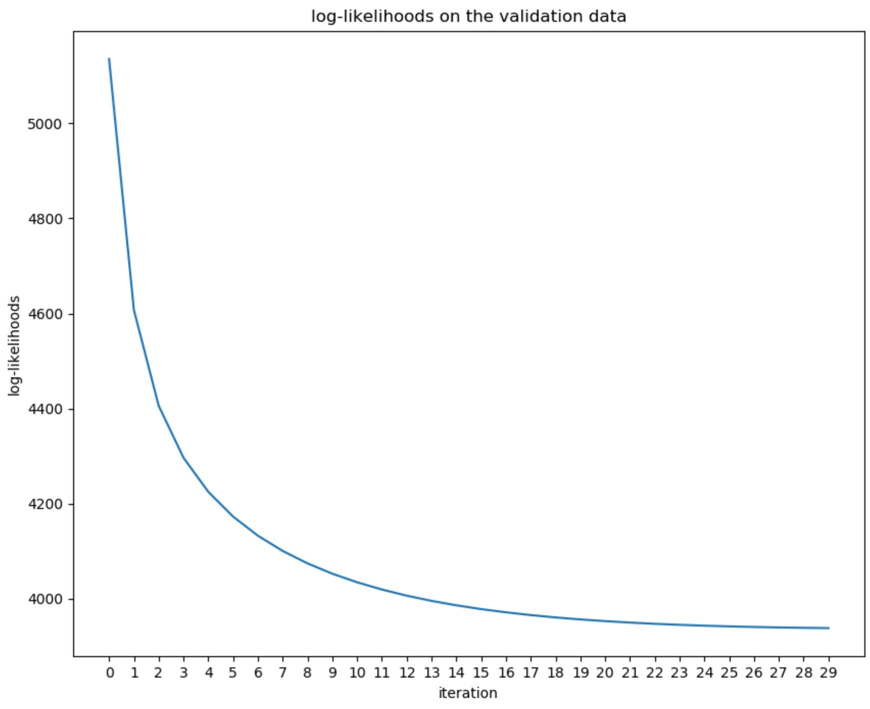
$$= -\sum_{i,j} \left(\frac{C_{i,j}}{p_{i,j}} + \frac{1 - C_{i,j}}{1 - p_{i,j}}\right)\left(p_{i,j}\right)\left(1 - \left(p_{i,j}\right)\right) * -1$$

$$= \sum_{i,j} C_{i,j} \left(1 - p_{i,j}\right) - (1 - C_{i,j}) p_{i,j}$$

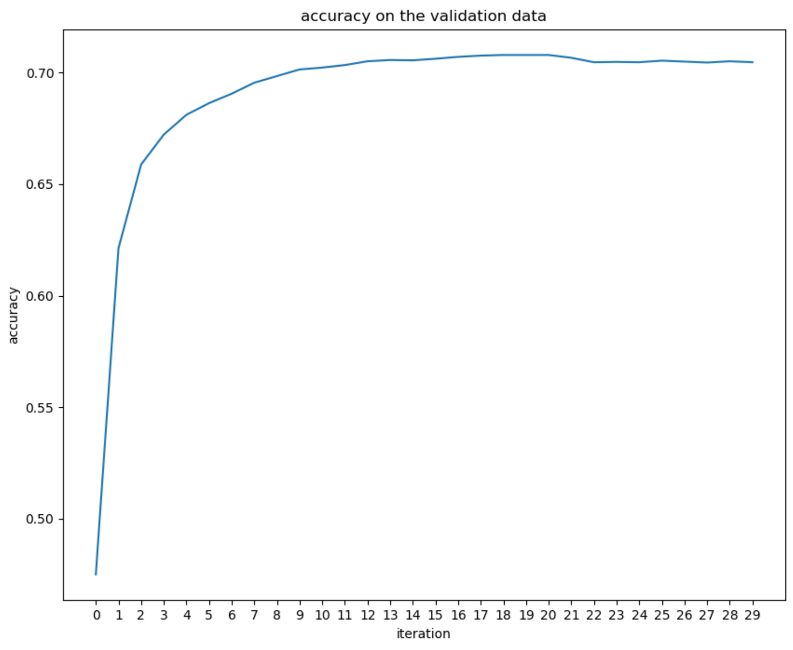
$$= \sum_{i,j} C_{i,j} - p_{i,j}$$

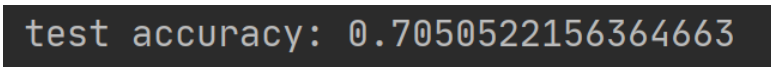
2 b
Learning rate = 0.01
Iterations = 30





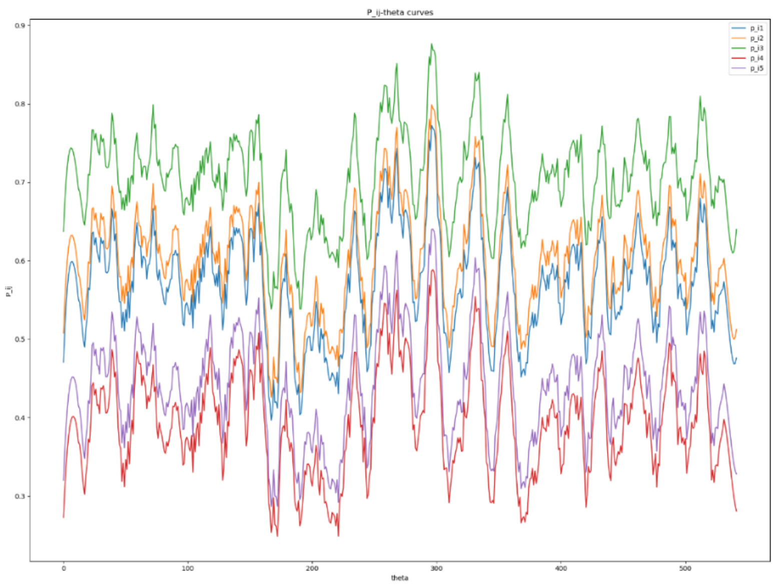
2 c





```
test accuracy: 0.7050522156364663
```

2 d



Answer: The curve in the figure fluctuates with theta value, but from the height of the curve, we can see the difficulty of the question. The higher the height, the more students can make the question and the simpler the question.

3(i) a

Report final validation and test performance for SVD with the chosen k

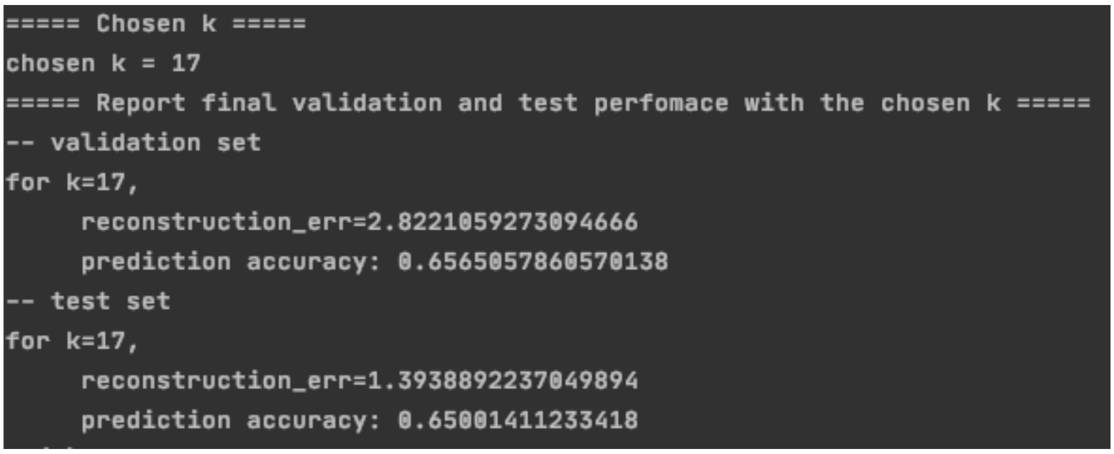
Answer:

for validation set with chosen k=17,

    reconstruction_err: approx. 2.82
    prediction accuracy: approx. 0.66

for test set with chosen k=17,

    reconstruction_err: approx. 1.39
    prediction accuracy: approx. 0.65



```
===== Chosen k =====
chosen k = 17
===== Report final validation and test perfomace with the chosen k =====
-- validation set
for k=17,
     reconstruction_err=2.8221059273094666
     prediction accuracy: 0.6565057860570138
-- test set
for k=17,
     reconstruction_err=1.3938892237049894
     prediction accuracy: 0.65001411233418
```

3(i) b

Limitation of perform SVD:

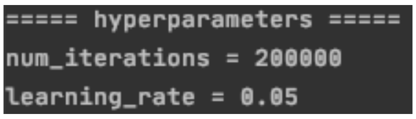
Answer:

We basically treated the missing entries as the mean value of the possibility of correctness for each question according to all users. We only predict the with evaluation on question, but the correctness prediction on a single question for a single user is not evaluated horizontally, that is personally.

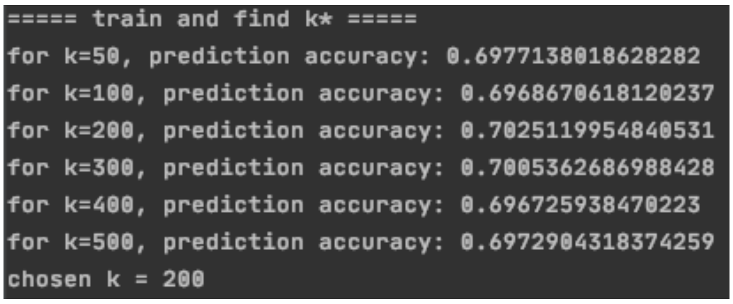
3(i) c

See function 'als' and 'update_u_z' in part_a/matrix_factorization.

3(i) d

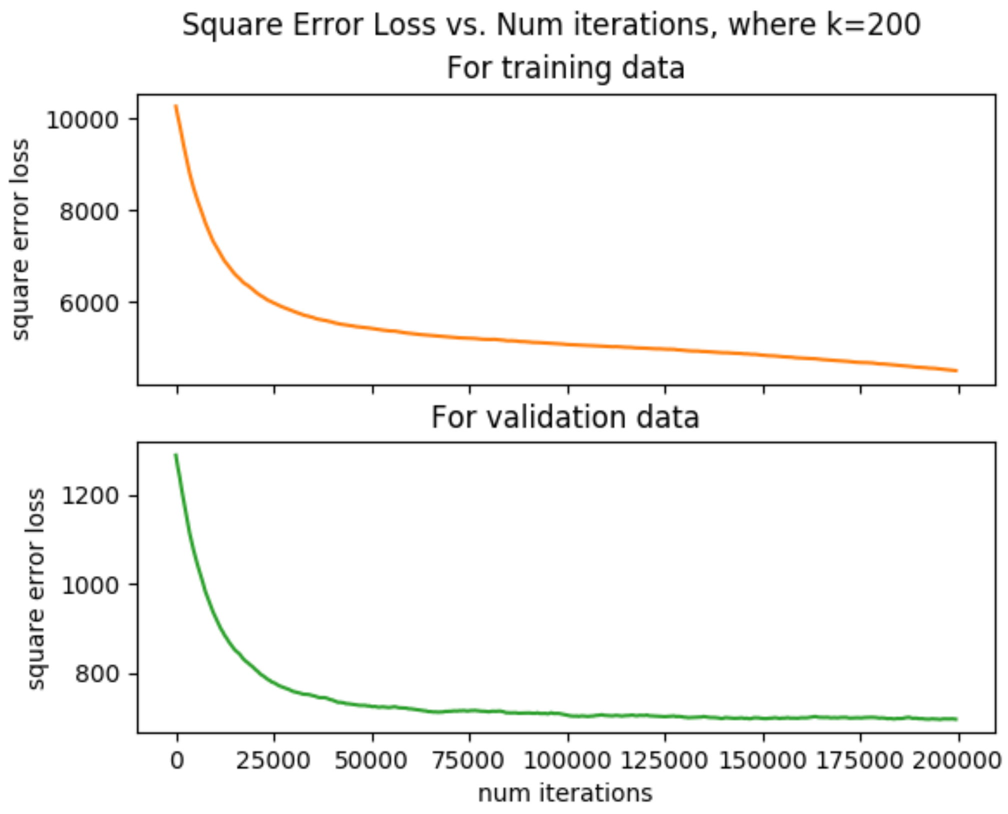


```
===== hyperparameters =====
num_iterations = 200000
learning_rate = 0.05
```



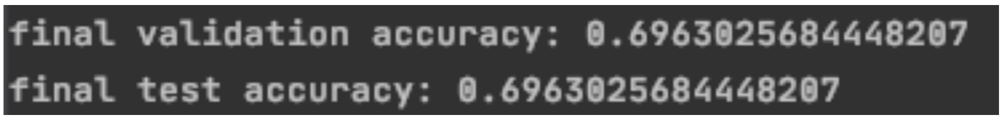
```
===== train and find k* =====
for k=50, prediction accuracy: 0.6977138018628282
for k=100, prediction accuracy: 0.6968670618120237
for k=200, prediction accuracy: 0.7025119954840531
for k=300, prediction accuracy: 0.7005362686988428
for k=400, prediction accuracy: 0.696725938470223
for k=500, prediction accuracy: 0.6972904318374259
chosen k = 200
```

3(i) e



Report how training and validation squared error loss change as a func of iterations:
Answer: Both training and validation squared error loss first decrease exponentially, then converge to a constant range of loss as the number of iterations increases.

Report final validation accuracy and test accuracy:



```
final validation accuracy: 0.6963025684448207
final test accuracy: 0.6963025684448207
```

3(i) f
For ALS, you trained the model as a regression problem; your loss function was a squared-error-loss. How would you modify the loss function if you would like to train the model as a binary classification problem? Describe your modified loss function.

Answer: If training the model as a binary classification problem, we may use the cross entropy loss with logistic activation function as the loss function.

Using logistic cross entropy can help with converting the a discrete 1-0 loss in binary classification problem into a continuous probability problem, and then classify with threshold.

$$y = \sigma\left(u_n^T z_m\right) = \frac{1}{1 + e^{-u_n^T z_m}}$$

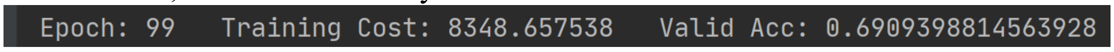
$$L_{CE} = -C_{nm} \log y - (1 - C_{nm}) \log (1 - y)$$

3(ii) a

1. ALS is mainly used for sparse matrices in recommendation systems, while neural networks are more widely applicable, not only recommendation systems.
2. ALS calculates sparse matrices, while the input of neural networks is generally text or images, which are not sparse matrices.
3. The ALS solution method is to fix one value first, use the least square method to solve the other value, and then alternate until the ALS converges; the neural network uses the gradient descent method to solve the optimal solution of the network.

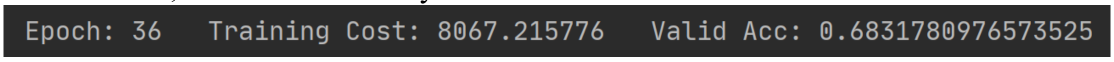
3(ii) b

When k = 10, validation accuracy:



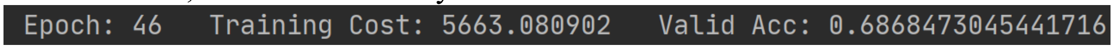
```
Epoch: 99   Training Cost: 8348.657538   Valid Acc: 0.6909398814563928
```

When k = 50, validation accuracy:



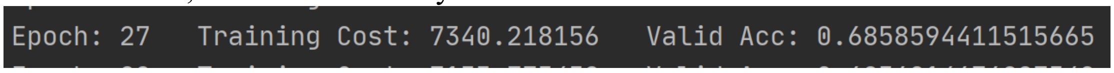
```
Epoch: 36   Training Cost: 8067.215776   Valid Acc: 0.6831780976573525
```

When k = 100, validation accuracy:



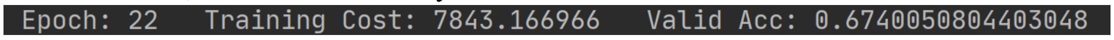
```
Epoch: 46   Training Cost: 5663.080902   Valid Acc: 0.6868473045441716
```

When k = 200, validation accuracy:



```
Epoch: 27   Training Cost: 7340.218156   Valid Acc: 0.6858594411515665
```

When k = 500, validation accuracy:

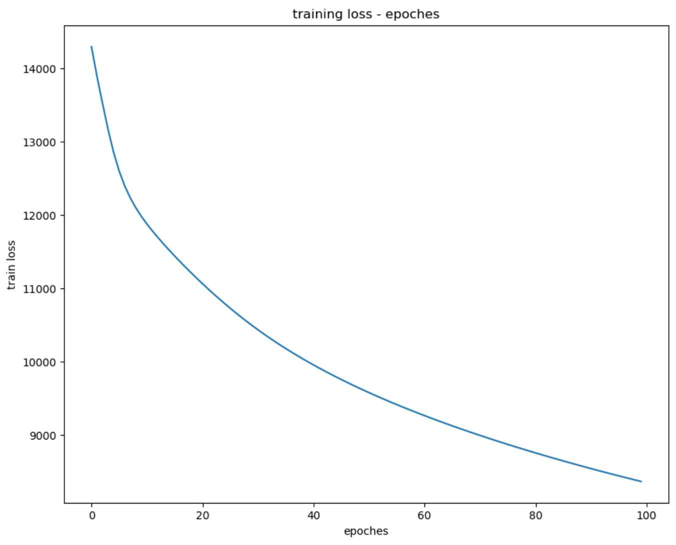


```
Epoch: 22   Training Cost: 7843.166966   Valid Acc: 0.6740050804403048
```

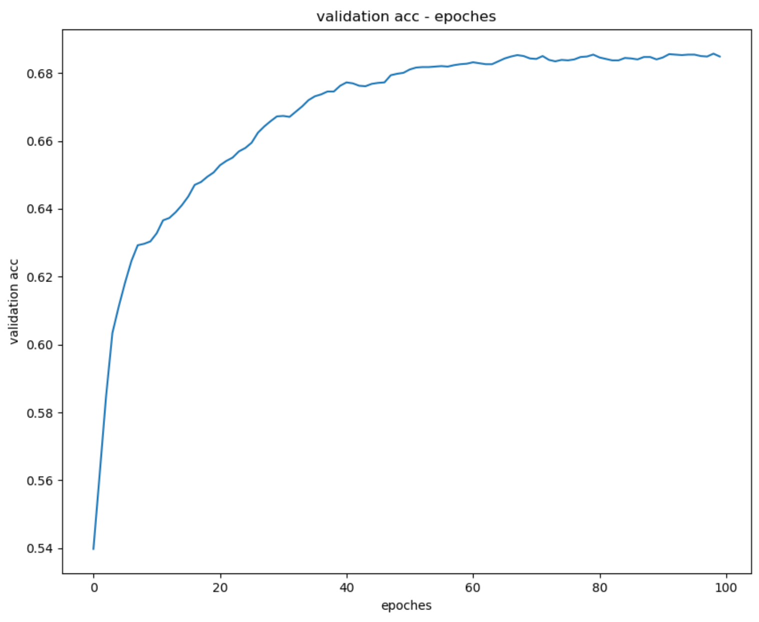
When k=10, learning rate=0.01, num_epoches=100, model has highest validation accuracy

3(ii)c

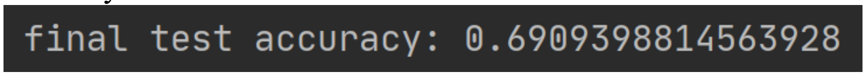
Training loss



Validation accuracy



Final test accuracy:

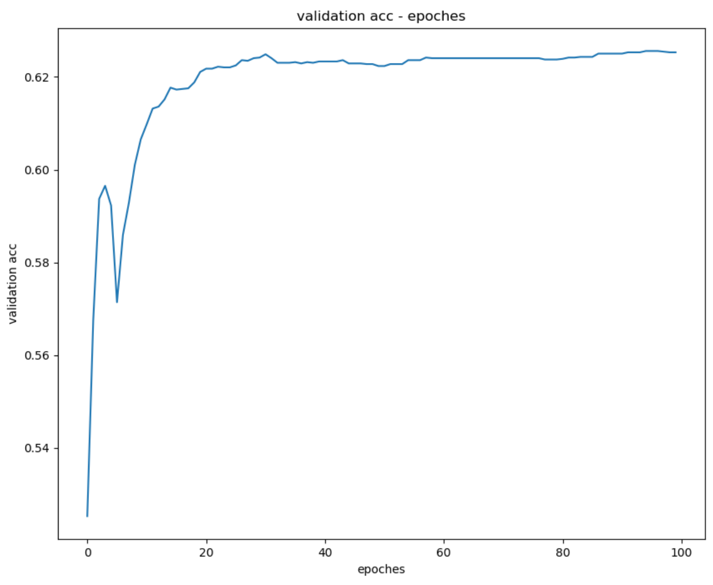


```
final test accuracy: 0.6909398814563928
```

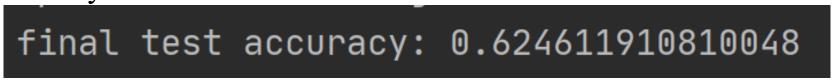
3(ii)d

1. Lambda = 0.001:

Validation accuracy



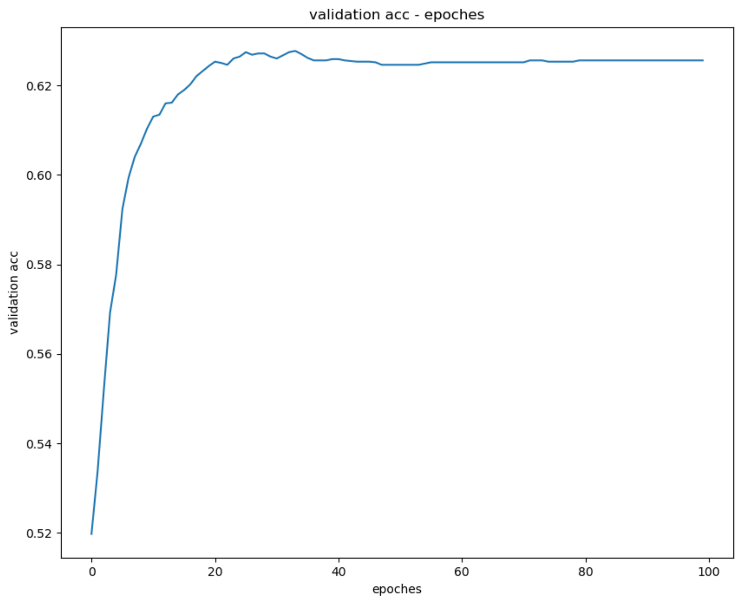
Final test accuracy:



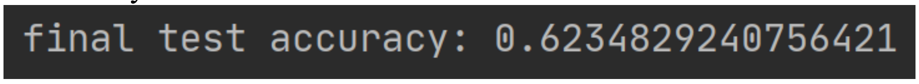
```
final test accuracy: 0.6246119108100048
```

2. Lambda = 0.01:

Validation accuracy



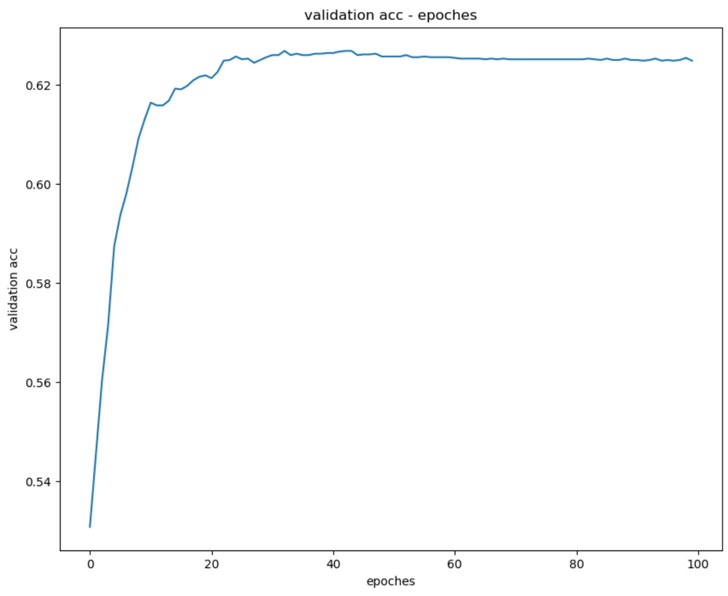
Final test accuracy:



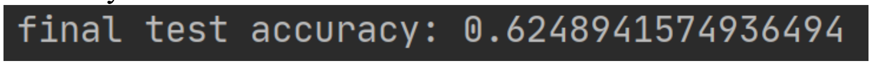
```
final test accuracy: 0.6234829240756421
```

3. Lambda = 0.1:

Validation accuracy



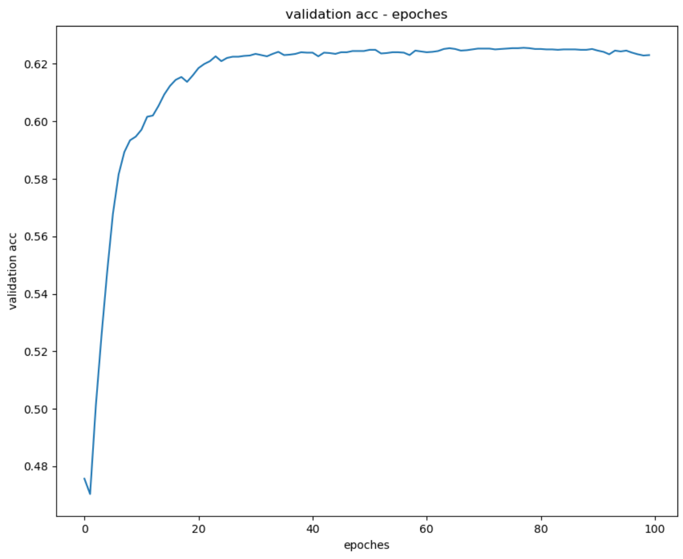
Final test accuracy:



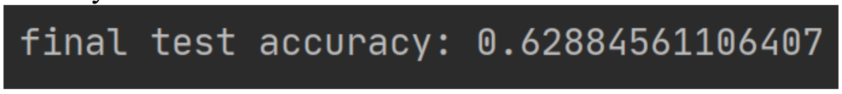
```
final test accuracy: 0.6248941574936494
```

4. Lambda = 1:

Validation accuracy



Final test accuracy:



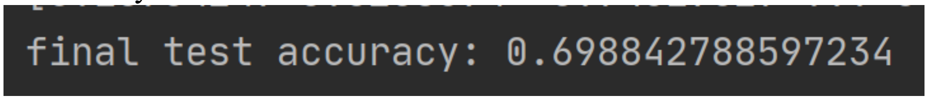
```
final test accuracy: 0.62884561106407
```

Answer:

when lambda=0.1, final test accuracy is 0.6248; model without the regularization penalty perform better.

4

Final test accuracy:



```
final test accuracy: 0.698842788597234
```

Answer:

Firstly training the knn, item response, and neural network models separately, and then calculate the predicted probability of the test set data in the trained model, then average the predicted probabilities of the three models, and finally get the predicted probability of the ensemble model, and then calculate Calculate the accuracy of the model. The accuracy of using the ensemble model is higher than that of using the basic model alone, because the classification principle of each basic model is different, and the ensemble model combines the advantages of each basic model.