1. (a) i.

$$L(H_{t-1}+h_t) = L(H_t) = \log(1+\exp(-y_i H_t(x_i)))$$

taylor approximation $L(H_t) \approx L(H_{t-1}) + \frac{dL(H_{t-1})}{dH_{t-1}} h_t + \frac{1}{2} \cdot \frac{d^2L(H_{t-1})}{dH_{t-1}^2} h_t^2$

$$\frac{dL(H_{t-1})}{dH_{t-1}} = \frac{-y_i e(-y_i H_{t-1}(x_i))}{1+\exp(-y_i H_{t-1}(x_i))} = \frac{-y_i}{\exp(y_i H_{t-1}(x_i))+1}$$

$$= y_i\left(1 - \frac{\exp(y_i H_{t-1}(x_i))}{\exp(y_i H_{t-1}(x_i))+1}\right)$$

$$= -y_i\left(1 - \frac{1}{1+\exp(-y_i H_{t-1}(x_i))}\right)$$

$$\sigma(x) = \frac{1}{1+\exp(-x)} \qquad = -y_i(1-\sigma(y_i H_{t-1}(x_i)))$$

$$\boxed{= -y_i(1-p_i^{(t-1)})} \text{ for } y_i = 1$$

$$\frac{d^2L(H_{t-1})}{dH_{t-1}^2} = \frac{-y_i \nabla_H(\exp(-y_i H_{t-1}(x_i)))}{(1+\exp(-y_i H_{t-1}(x_i)))^2}$$

$$= \frac{y_i^2 \exp(-y_i H_{t-1}(x_i))}{(1+\exp(-y_i H_{t-1}(x_i)))^2}$$

$$= y_i^2 \cdot \frac{1}{1+\exp(-y_i H_{t-1}(x_i))} \cdot \left(1 - \frac{1}{1+\exp(-y_i(H_{t-1}(x_i)}\right)$$

$$= y_i^2 \cdot \sigma(y_i H_{t-1}(x_i)) \cdot (1-\sigma(y_i H_{t-1}(x_i)))$$

$$\boxed{= y_i^2 \cdot p_i^{(t-1)} \cdot (1-p_i^{(t-1)})}$$

$$\boxed{L(H_t) \approx L(H_{t-1}) - y_i(1-p_i^{(t-1)}) h_t + \frac{1}{2} y_i^2 p_i^{(t-1)}(1-p_i^{(t-1)}) h_t^2}$$

iii. $L(H_t) \approx L(H_{t-1}) - y_i(1-p_i^{(t-1)})h_t + \frac{1}{2}y_i^2 p_i^{(t-1)}(1-p_i^{(t-1)})h_t^2$

$$= L(H_{t-1}) - p_i^{(t-1)}(1-p_i^{(t-1)})\left[\frac{y_i h_t}{p_i^{(t-1)}} - \frac{1}{2}y_i^2 h_t^2\right]$$

$$= L(H_{t-1}) - p_i^{(t-1)}(1-p_i^{(t-1)})\left[\frac{y_i(1-p_i^{(t-1)})}{p_i^{(t-1)}(1-p_i^{(t-1)})}h_t - \frac{1}{2}h_t^2\right]$$

$$= L(H_{t-1}) - \frac{1}{2}p_i^{(t-1)}(1-p_i^{(t-1)})\left[2\frac{y_i - y_i p_i^{(t-1)}}{p_i^{(t-1)}(1-p_i^{(t-1)})}h_t - h_t^2\right]$$

since $\frac{y_i - y_i p_i^{(t-1)}}{p_i^{(t-1)}(1-p^{(t-1)})} = \frac{\frac{y_i}{2}+\frac{1}{2}-p_i^{(t-1)}}{p_i^{(t-1)}(1-p_i^{(t-1)})}$ when $y_i = 1$

$$= L(H_{t-1}) - \frac{1}{2}w_i^{(t)}(2z_i^{(t)}h_t - h_t^2)$$

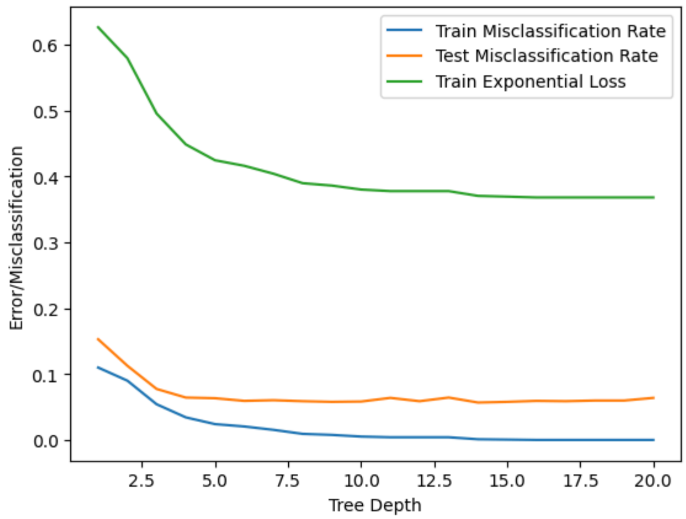
since $(z_i^{(t)})^2$ not depend on $h(x_i)$

$\min L(H_t) \approx L(H_{t-1}) + \frac{1}{2}w_i^{(t)}((z_i^{(t)})^2 - 2z_i^{(t)}h_t + h_t^2)$

$$\approx w_i^{(t)}(z_i^{(t)} - h_t)^2$$

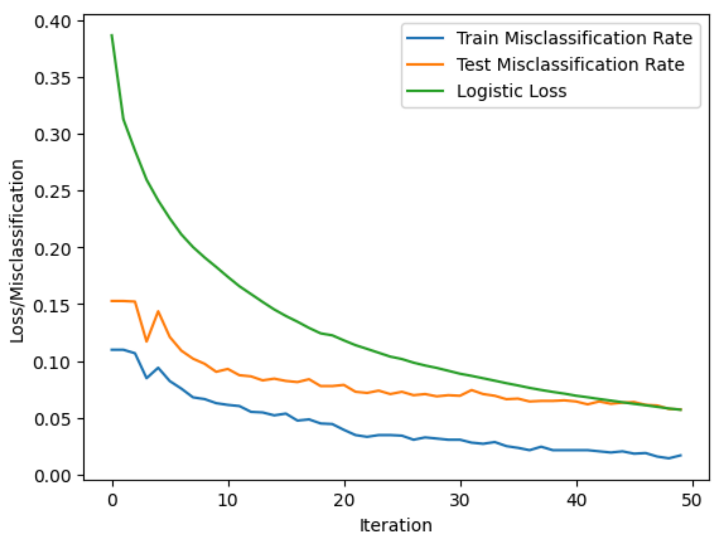
(b)

I. Deep Decision Tree



Minimum Train Misclassification Rate: 0.0
Minimum Test Misclassification Rate: 0.05675539929683576
Minimum Train Exponential Loss: 0.3678794411714424

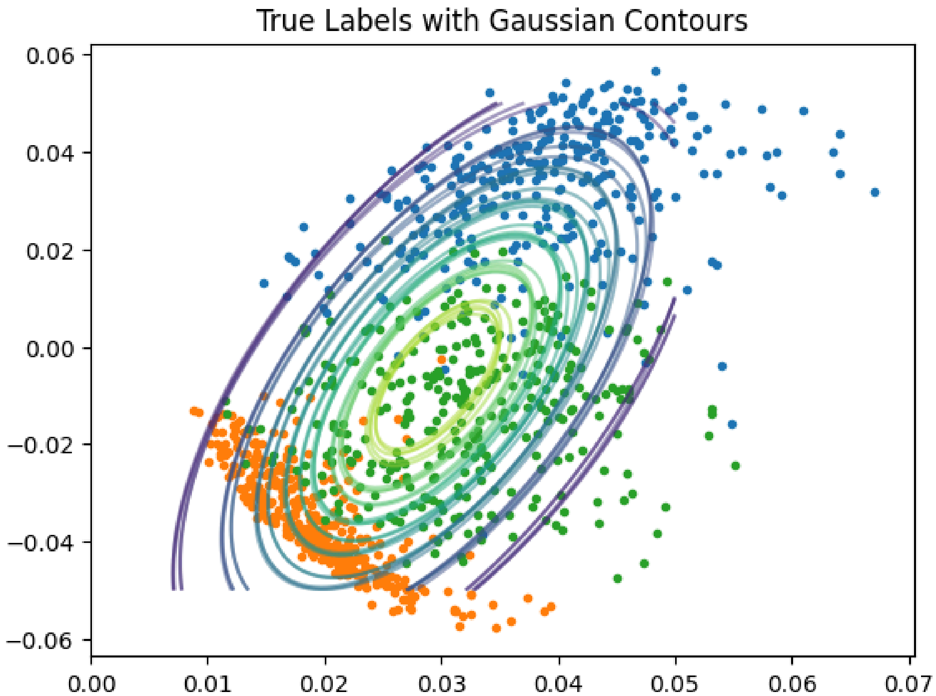
II. Boosted Decision Stump

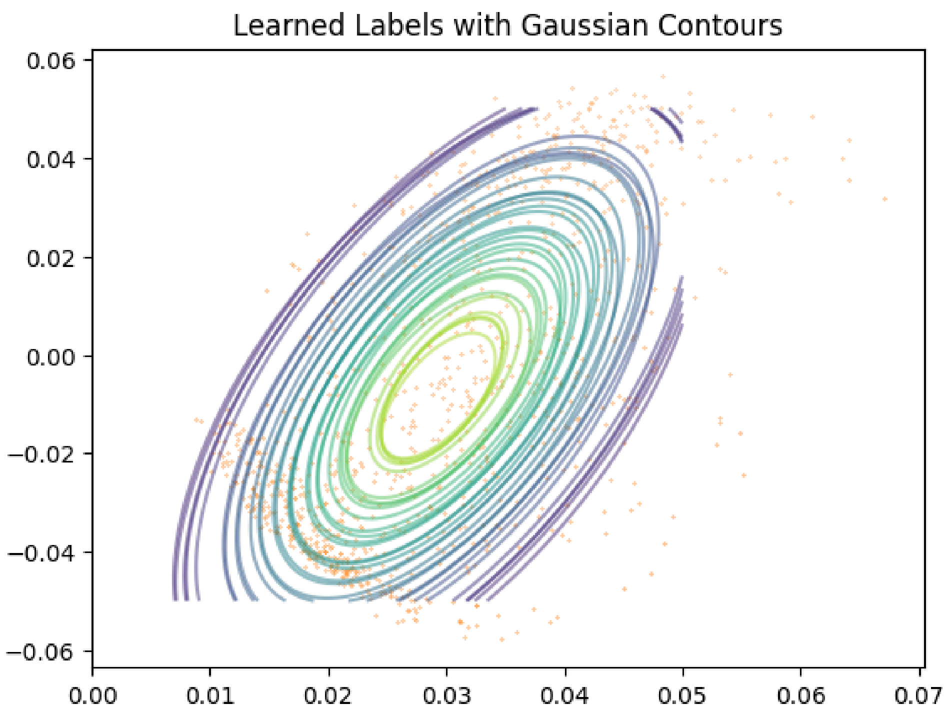


Minimum Train Misclassification Rate: 0.014300306435137897
Minimum Test Misclassification Rate: 0.05725765946760422
Minimum Train Logistic Loss: 0.057035361288887906

The training misclassification rate of the deep decision tree is 0, which indicates that it is overfitting, so it may lead to higher test error. Compared with the deep decision tree running time of 5 seconds, the running time of LogitBoost with decision stumps is 2 seconds, which indicates that LogitBoost can reduce the training loss and misclassification rate faster, and LogitBoost may have better generalization ability.

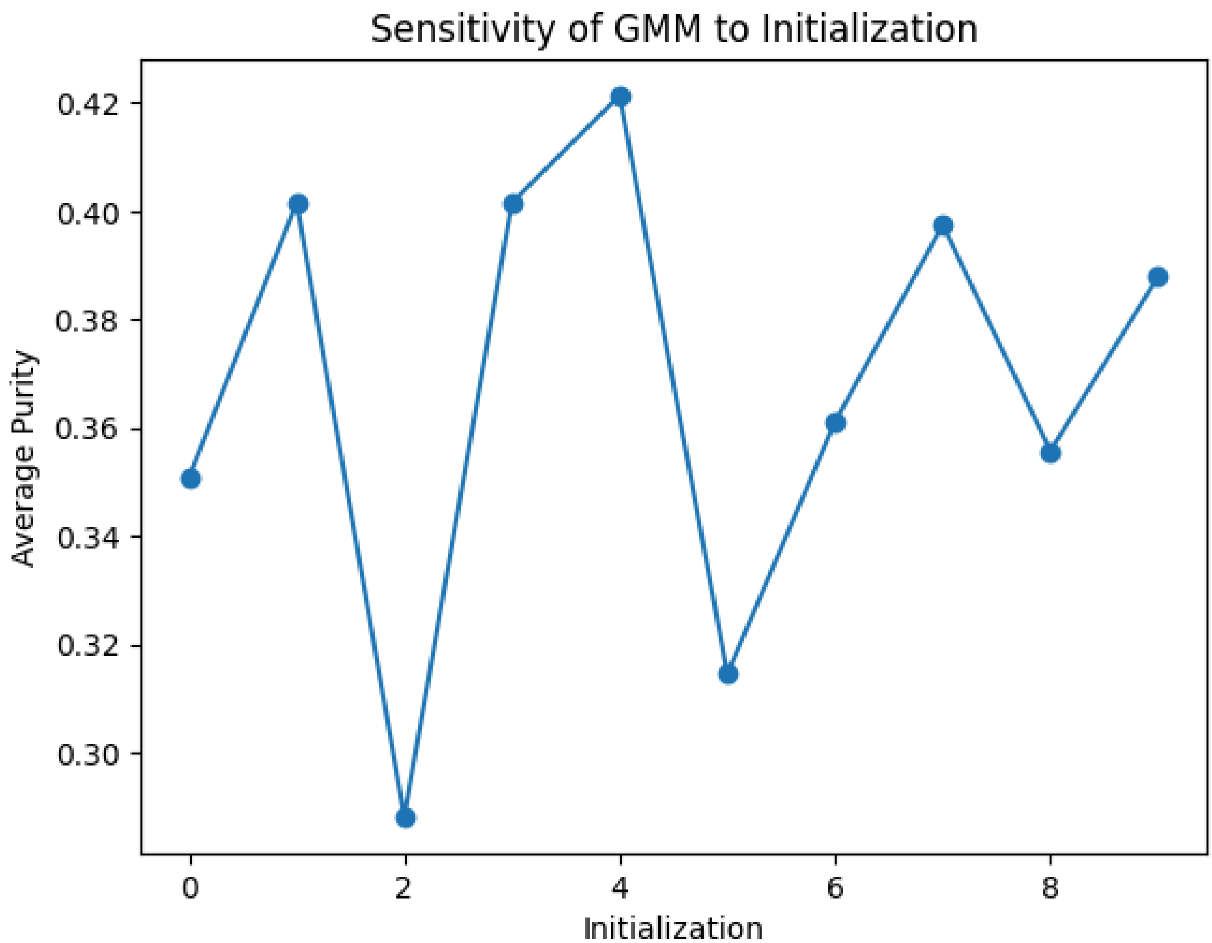
2. GMM problem
(a).





The Gaussian contours are not completely aligned with the true labels, probably because the true labels are not considered during the learning process of the GMM clustering. The Gaussian contours appear to overlap probably because the labels are not well separated in 2D space.

(b).



I. Is GMM sensitive to initialization?
GMM is sensitive to initialization, and the changes in the purity plot reflect this sensitivity.

II. Can it be used to cluster handwritten digits?
GMM can have difficulty with high-dimensional data due to class overlap. Some clusters have high purity, but overall results are mixed.

III. Improvements:
Dimensionality reduction: separate lower-dimensional clusters of digits.
Regularization: prevent overfitting in high dimensions.
Initialization: more robust initialization.

3. Movie recommendations

I. global_avg = np.mean(ratings_df['rating'])

II. Train RMSE: 1.0453378956423331, Validation RMSE: 1.0384954134276096, Test RMSE: 1.043776737697598

III. ratings(u,m) = global_average + average_user(u).
Train RMSE User: 0.9303541467440325,
Validation RMSE User: 0.9453702230795641,
Test RMSE User: 0.9506944370582363

ratings(u,m) = global_average + average_movie(u).
Train RMSE Movie: 0.8298431428616648,
Validation RMSE Movie: 1.018877291556982,
Test RMSE Movie: 1.026990470728207

IV. ratings(u,m) = global_average + 1/2 * average_user(u)+ 1/2 * average_movie(m).
Train RMSE Combine: 0.8125245349156025,
Validation RMSE Combine: 0.9139580565016393,
Test RMSE Combine: 0.9201928948677766

V. $$f(U,V) = \frac{1}{2}\sum_{i,j\in\Omega}\left(u_i^T v_j + c + \frac{\overline{m_j}}{2} + \frac{\overline{w_i}}{2} - R_{i,j}\right)^2$$

Gradients:

$$\bigtriangledown_{ui} = \sum_{j\in\Omega}\left(u_i^T v_j + c + \frac{\overline{m_j}}{2} + \frac{\overline{w_i}}{2} - R_{i,j}\right) \cdot v_j$$

$$\bigtriangledown_{vi} = \sum_{i\in\Omega}\left(u_i^T v_j + c + \frac{\overline{m_j}}{2} + \frac{\overline{w_i}}{2} - R_{i,j}\right) \cdot u_j$$

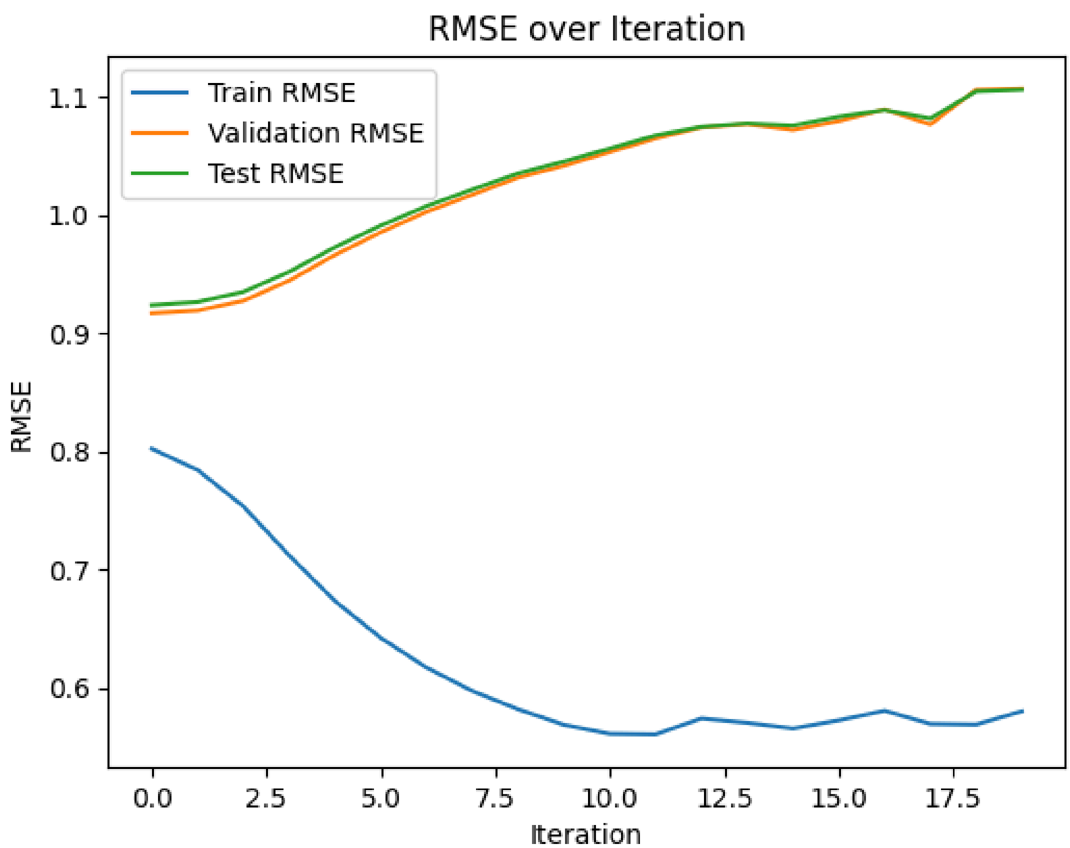
VI.

```
def get_grad_ui(U,V,i):
    j_nonzero = R.col[R.row == i]
    gradient = np.zeros(U.shape[1])
    for j in j_nonzero:
        r_ij = R.data[np.logical_and(R.row == i, R.col == j)][0]
        gradient += (np.dot(U[i], V[j]) + global_avg + 0.5 * movie_avg[j] + 0.5 * user_avg[i] - r_ij) * V[j]
    return gradient

def get_grad_vj(U,V,j):
    i_nonzero = R.row[R.col == j]
    gradient = np.zeros(V.shape[1])
    for i in i_nonzero:
        r_ij = R.data[np.logical_and(R.row == i, R.col == j)][0]
        gradient += (np.dot(U[i], V[j]) + global_avg + 0.5 * movie_avg[j] + 0.5 * user_avg[i] - r_ij) * U[i]
    return gradient
```

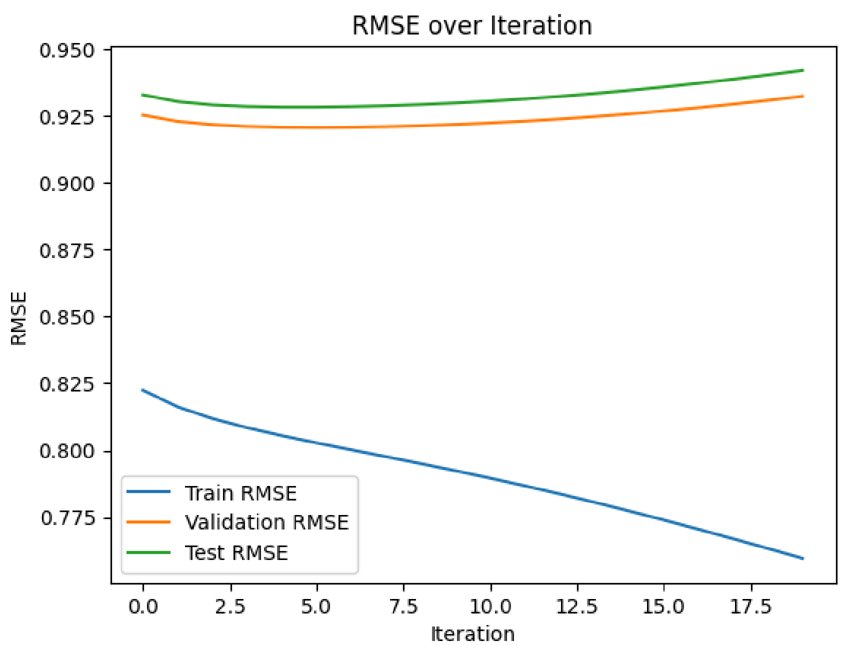
VII. r = 5, step size = 0.05, maxiter = 20.

Iteration 1: Train RMSE = 0.8020, Validation RMSE = 0.9167, Test RMSE = 0.9236
Iteration 2: Train RMSE = 0.7841, Validation RMSE = 0.9190, Test RMSE = 0.9263
Iteration 3: Train RMSE = 0.7531, Validation RMSE = 0.9273, Test RMSE = 0.9348
Iteration 4: Train RMSE = 0.7117, Validation RMSE = 0.9443, Test RMSE = 0.9518
Iteration 5: Train RMSE = 0.6731, Validation RMSE = 0.9661, Test RMSE = 0.9726
Iteration 6: Train RMSE = 0.6420, Validation RMSE = 0.9852, Test RMSE = 0.9909
Iteration 7: Train RMSE = 0.6168, Validation RMSE = 1.0025, Test RMSE = 1.0074
Iteration 8: Train RMSE = 0.5971, Validation RMSE = 1.0170, Test RMSE = 1.0214
Iteration 9: Train RMSE = 0.5816, Validation RMSE = 1.0315, Test RMSE = 1.0350
Iteration 10: Train RMSE = 0.5685, Validation RMSE = 1.0415, Test RMSE = 1.0451
Iteration 11: Train RMSE = 0.5610, Validation RMSE = 1.0532, Test RMSE = 1.0561
Iteration 12: Train RMSE = 0.5605, Validation RMSE = 1.0646, Test RMSE = 1.0671
Iteration 13: Train RMSE = 0.5739, Validation RMSE = 1.0739, Test RMSE = 1.0744
Iteration 14: Train RMSE = 0.5702, Validation RMSE = 1.0767, Test RMSE = 1.0773
Iteration 15: Train RMSE = 0.5654, Validation RMSE = 1.0718, Test RMSE = 1.0755
Iteration 16: Train RMSE = 0.5723, Validation RMSE = 1.0793, Test RMSE = 1.0829
Iteration 17: Train RMSE = 0.5804, Validation RMSE = 1.0890, Test RMSE = 1.0885
Iteration 18: Train RMSE = 0.5693, Validation RMSE = 1.0764, Test RMSE = 1.0814
Iteration 19: Train RMSE = 0.5688, Validation RMSE = 1.1058, Test RMSE = 1.1044
Iteration 20: Train RMSE = 0.5800, Validation RMSE = 1.1068, Test RMSE = 1.1058



VIII. r = 3, step size = 0.01, maxiter = 20.

Iteration 1: Train RMSE = 0.8224, Validation RMSE = 0.9253, Test RMSE = 0.9328
Iteration 2: Train RMSE = 0.8162, Validation RMSE = 0.9229, Test RMSE = 0.9303
Iteration 3: Train RMSE = 0.8118, Validation RMSE = 0.9217, Test RMSE = 0.9291
Iteration 4: Train RMSE = 0.8084, Validation RMSE = 0.9210, Test RMSE = 0.9285
Iteration 5: Train RMSE = 0.8055, Validation RMSE = 0.9207, Test RMSE = 0.9283
Iteration 6: Train RMSE = 0.8027, Validation RMSE = 0.9206, Test RMSE = 0.9283
Iteration 7: Train RMSE = 0.8002, Validation RMSE = 0.9207, Test RMSE = 0.9284
Iteration 8: Train RMSE = 0.7976, Validation RMSE = 0.9209, Test RMSE = 0.9288
Iteration 9: Train RMSE = 0.7950, Validation RMSE = 0.9213, Test RMSE = 0.9292
Iteration 10: Train RMSE = 0.7923, Validation RMSE = 0.9217, Test RMSE = 0.9298
Iteration 11: Train RMSE = 0.7896, Validation RMSE = 0.9223, Test RMSE = 0.9305
Iteration 12: Train RMSE = 0.7867, Validation RMSE = 0.9230, Test RMSE = 0.9313
Iteration 13: Train RMSE = 0.7837, Validation RMSE = 0.9238, Test RMSE = 0.9323
Iteration 14: Train RMSE = 0.7806, Validation RMSE = 0.9247, Test RMSE = 0.9333
Iteration 15: Train RMSE = 0.7773, Validation RMSE = 0.9257, Test RMSE = 0.9345
Iteration 16: Train RMSE = 0.7739, Validation RMSE = 0.9268, Test RMSE = 0.9358
Iteration 17: Train RMSE = 0.7705, Validation RMSE = 0.9280, Test RMSE = 0.9372
Iteration 18: Train RMSE = 0.7669, Validation RMSE = 0.9294, Test RMSE = 0.9387
Iteration 19: Train RMSE = 0.7633, Validation RMSE = 0.9308, Test RMSE = 0.9402
Iteration 20: Train RMSE = 0.7596, Validation RMSE = 0.9323, Test RMSE = 0.9419



I think the best values for r, step size, maxiter are: r = 3, step size = 0.01, maxiter = 20. This is because the information is not visible in the previous figure, but in this figure:

1. The training RMSE decreases slowly after 2 iterations, which indicates possible overfitting.
2. The validation and test RMSE decrease initially, but start to increase after 6 iterations. This indicates the beginning of overfitting.