Name: Rhugaved Narmade
Net ID: rrn210004
CS6375 Assignment 2

1 Collaborative Filtering
Google Collab Code:
https://colab.research.google.com/drive/12J4-t0B829YQjM9ZrLx5ch_cGYYSqXy4?usp=sharing

Code Explanation:

1. We read the Training and Testing files into dataframes
2. Looking at the data, we have 28978 unique users and 1821 unique movies in the train data. Not every pair of users and movies has a rating.
3. We convert the train dataframe to a numpy array of shape (1821, 28978), where rows index movies and columns index users and values are ratings. For non rated, we put 0
4. In our weight calculaton and Prediction equations, v(i,j) - mean(v(i)) value is used everywhere, so we calculate and store it.
5. We also calculate (v(i,j) - mean(v(i)))^2 and store it as it's used in weight calculation. Performing steps 4 and 5 help us in looking up the values for every operation instead of calculating them everytime. This saves a lot of time as v(i,j) - mean(v(i)) and (v(i,j) - mean(v(i)))^2 are used for every element in calculation for each element pair weight
6. Next , we calculate weights for every user pair. We use numpy matrices and use vectorized operations which help in reduction in calculation time. Instead of a higher order polynomial time using loops which takes hours, using vectorized operations does the work in ~15 mins
7. Next, for prediction, we need k, which we calculate and store beforehand.
8. The predict function too does vectorized operations to save time and returns an output array which is actually a prediction for every (user, movie) pair (28978, 1821). Thus the train data as well as test data is a subset of the output matrix of predict.
9. We then calculate the Absolute Error and Mean Squared Error

Training Errors:

Mean Absolute Error: 0.7311922948753752
Mean Squared Error: 2.0225304484670485

Testing Errors:

Mean Absolute Error: 0.7836825201643344
Mean Squared Error: 3.729596269279923

2 Neural Networks, K-nearest neighbors and SVMs

Google Collab Link:
https://colab.research.google.com/drive/1KqZ4QhSi9vclVTg_BhKwbqwQr3--Bx8n?usp=sharing

1. SVM:
    a. Different parameters used:
        C (Regularization Parameter) = [0.1, 1, 10]
        kernel Function = ["linear", "poly", "rbf", "sigmoid"]
    b. After Prediction on Test Set:

1) Model: SVM, Parameters: kernel: linear, Reg parameter C: 0.1, accuracy: 0.8246, Absolute Error: 0.5946, Mean Squared Error: 2.6294

2) Model: SVM, Parameters: kernel: linear, Reg parameter C: 1, accuracy: 0.7485, Absolute Error: 0.8713, Mean Squared Error: 3.9459

3) Model: SVM, Parameters: kernel: linear, Reg parameter C: 10, accuracy: 0.7481, Absolute Error: 0.8741, Mean Squared Error: 3.9597

4) Model: SVM, Parameters: kernel: poly, Reg parameter C: 0.1, accuracy: 0.3469, Absolute Error: 3.036, Mean Squared Error: 17.6612

5) Model: SVM, Parameters: kernel: poly, Reg parameter C: 1, accuracy: 0.8455, Absolute Error: 0.5299, Mean Squared Error: 2.2545

6) Model: SVM, Parameters: kernel: poly, Reg parameter C: 10, accuracy: 0.975, Absolute Error: 0.0977, Mean Squared Error: 0.4857

7) Model: SVM, Parameters: kernel: rbf, Reg parameter C: 0.1, accuracy: 0.8723, Absolute Error: 0.465, Mean Squared Error: 2.1346

8) Model: SVM, Parameters: kernel: rbf, Reg parameter C: 1, accuracy: 0.9764, Absolute Error: 0.094, Mean Squared Error: 0.47

9) Model: SVM, Parameters: kernel: rbf, Reg parameter C: 10, accuracy: 0.9822, Absolute Error: 0.07, Mean Squared Error: 0.3512

10) Model: SVM, Parameters: kernel: sigmoid, Reg parameter C: 0.1, accuracy: 0.3979, Absolute Error: 2.455, Mean Squared Error: 12.6086

11) Model: SVM, Parameters: kernel: sigmoid, Reg parameter C: 1, accuracy: 0.547, Absolute Error: 1.8111, Mean Squared Error: 8.8109

12) Model: SVM, Parameters: kernel: sigmoid, Reg parameter C: 10, accuracy: 0.6069, Absolute Error: 1.492, Mean Squared Error: 7.0926

2.2 Neural Networks

a. Different Parameters used:
   hidden_layers = [10, 100]
   activations = ["logistic", "tanh", "relu"]
   optimizers = ["sgd", "adam"]
   reg_parameters = [0.001, 0.1]
   lr_parameters = [0.001, 0.01, 0.1]

b. Total 72 models were tested, out of which, some models are described below. Others can be found in the code file.

1) Model: NN, Parameters: No. hidden layers: 10, Activation: logistic, Optimizer: sgd, Reg Para: 0.1, Learning Rate: 0.1, accuracy: 0.9302, Absolute Error: 0.2649, Mean Squared Error: 1.2871

2) Model: NN, Parameters: No. hidden layers: 10, Activation: logistic, Optimizer: adam, Reg Para: 0.001, Learning Rate: 0.1, accuracy: 0.8983, Absolute Error: 0.3374, Mean Squared Error: 1.501

3) Model: NN, Parameters: No. hidden layers: 10, Activation: tanh, Optimizer: sgd, Reg Para: 0.001, Learning Rate: 0.01, accuracy: 0.9341, Absolute Error: 0.2356, Mean Squared Error: 1.0946

4) Model: NN, Parameters: No. hidden layers: 10, Activation: tanh, Optimizer: sgd, Reg Para: 0.1, Learning Rate: 0.1, accuracy: 0.9366, Absolute Error: 0.2309, Mean Squared Error: 1.0831

5) Model: NN, Parameters: No. hidden layers: 10, Activation: tanh, Optimizer: adam, Reg Para: 0.001, Learning Rate: 0.1, accuracy: 0.8842, Absolute Error: 0.434, Mean Squared Error: 2.1624

6) Model: NN, Parameters: No. hidden layers: 10, Activation: tanh, Optimizer: adam, Reg Para: 0.1, Learning Rate: 0.1, accuracy: 0.8609, Absolute Error: 0.5261, Mean Squared Error: 2.5817

7) Model: NN, Parameters: No. hidden layers: 10, Activation: relu, Optimizer: sgd, Reg Para: 0.1, Learning Rate: 0.01, accuracy: 0.9403, Absolute Error: 0.2252, Mean Squared Error: 1.0952

8) Model: NN, Parameters: No. hidden layers: 10, Activation: relu, Optimizer: adam, Reg Para: 0.001, Learning Rate: 0.1, accuracy: 0.6445, Absolute Error: 1.2721, Mean Squared Error: 5.5275

9) Model: NN, Parameters: No. hidden layers: 10, Activation: relu, Optimizer: adam, Reg Para: 0.1, Learning Rate: 0.1, accuracy: 0.3384, Absolute Error: 2.3188, Mean Squared Error: 10.3578

10) Model: NN, Parameters: No. hidden layers: 10, Activation: relu, Optimizer: adam, Reg Para: 0.1, Learning Rate: 0.001, accuracy: 0.9432, Absolute Error: 0.2069, Mean Squared Error: 1.0117

11) Model: NN, Parameters: No. hidden layers: 100, Activation: logistic, Optimizer: sgd, Reg Para: 0.001, Learning Rate: 0.001, accuracy: 0.913, Absolute Error: 0.3289, Mean Squared Error: 1.6079

12) Model: NN, Parameters: No. hidden layers: 100, Activation: logistic, Optimizer: sgd, Reg Para: 0.001, Learning Rate: 0.1, accuracy: 0.9796, Absolute Error: 0.0809, Mean Squared Error: 0.4065

13) Model: NN, Parameters: No. hidden layers: 100, Activation: logistic, Optimizer: adam, Reg Para: 0.001, Learning Rate: 0.1, accuracy: 0.9073, Absolute Error: 0.337, Mean Squared Error: 1.5726

14) Model: NN, Parameters: No. hidden layers: 100, Activation: logistic, Optimizer: adam, Reg Para: 0.1, Learning Rate: 0.1, accuracy: 0.8955, Absolute Error: 0.3851, Mean Squared Error: 1.8103

15) Model: NN, Parameters: No. hidden layers: 100, Activation: tanh, Optimizer: sgd, Reg Para: 0.001, Learning Rate: 0.01, accuracy: 0.9769, Absolute Error: 0.0916, Mean Squared Error: 0.4586

16) Model: NN, Parameters: No. hidden layers: 100, Activation: tanh, Optimizer: adam, Reg Para: 0.001, Learning Rate: 0.1, accuracy: 0.8665, Absolute Error: 0.5062, Mean Squared Error: 2.4822

17) Model: NN, Parameters: No. hidden layers: 100, Activation: tanh, Optimizer: adam, Reg Para: 0.1, Learning Rate: 0.01, accuracy: 0.9674, Absolute Error: 0.1289, Mean Squared Error: 0.6699

18) Model: NN, Parameters: No. hidden layers: 100, Activation: tanh, Optimizer: adam, Reg Para: 0.1, Learning Rate: 0.1, accuracy: 0.8026, Absolute Error: 0.6766, Mean Squared Error: 2.9618

19) Model: NN, Parameters: No. hidden layers: 100, Activation: relu, Optimizer: sgd, Reg Para: 0.1, Learning Rate: 0.1, accuracy: 0.9804, Absolute Error: 0.0805, Mean Squared Error: 0.4183

20) Model: NN, Parameters: No. hidden layers: 100, Activation: relu, Optimizer: adam, Reg Para: 0.001, Learning Rate: 0.001, accuracy: 0.98, Absolute Error: 0.0784, Mean Squared Error: 0.3914

21) Model: NN, Parameters: No. hidden layers: 100, Activation: relu, Optimizer: adam, Reg Para: 0.001, Learning Rate: 0.1, accuracy: 0.8994, Absolute Error: 0.3958, Mean Squared Error: 1.9856

22) Model: NN, Parameters: No. hidden layers: 100, Activation: relu, Optimizer: adam, Reg Para: 0.1, Learning Rate: 0.01, accuracy: 0.9679, Absolute Error: 0.1184, Mean Squared Error: 0.5768

23) Model: NN, Parameters: No. hidden layers: 100, Activation: relu, Optimizer: adam, Reg Para: 0.1, Learning Rate: 0.1, accuracy: 0.8898, Absolute Error: 0.412, Mean Squared Error: 1.9904

2.3 KNN:

Different parameters used:

n_neighbors = [3, 5, 7]
weights = ["uniform", "distance"]
algorithm = ["ball_tree", "kd_tree", "brute"]
&
n_neighbors = [1, 3, 5, 7, 9]
weights = ["uniform", "distance"]
algorithm = ["auto"]

The ‘algorithm’ parameter didn’t change the values of the output.

After Prediction on Test Set:

1) Model: KNN, Parameters: Nearest Neighbours: 1, Weight Function: uniform, Algorithm: auto, accuracy: 0.962, Absolute Error: 0.142, Mean Squared Error: 0.65

2) Model: KNN, Parameters: Nearest Neighbours: 1, Weight Function: distance, Algorithm: auto, accuracy: 0.962, Absolute Error: 0.142, Mean Squared Error: 0.65

3) Model: KNN, Parameters: Nearest Neighbours: 3, Weight Function: uniform, Algorithm: auto, accuracy: 0.962, Absolute Error: 0.149, Mean Squared Error: 0.715

4) Model: KNN, Parameters: Nearest Neighbours: 3, Weight Function: distance, Algorithm: auto, accuracy: 0.965, Absolute Error: 0.139, Mean Squared Error: 0.661

5) Model: KNN, Parameters: Nearest Neighbours: 5, Weight Function: uniform, Algorithm: auto, accuracy: 0.961, Absolute Error: 0.154, Mean Squared Error: 0.734

6) Model: KNN, Parameters: Nearest Neighbours: 5, Weight Function: distance, Algorithm: auto, accuracy: 0.964, Absolute Error: 0.144, Mean Squared Error: 0.69

7) Model: KNN, Parameters: Nearest Neighbours: 7, Weight Function: uniform, Algorithm: auto, accuracy: 0.962, Absolute Error: 0.153, Mean Squared Error: 0.749

8) Model: KNN, Parameters: Nearest Neighbours: 7, Weight Function: distance, Algorithm: auto, accuracy: 0.964, Absolute Error: 0.142, Mean Squared Error: 0.686

9) Model: KNN, Parameters: Nearest Neighbours: 9, Weight Function: uniform, Algorithm: auto, accuracy: 0.952, Absolute Error: 0.188, Mean Squared Error: 0.902

10) Model: KNN, Parameters: Nearest Neighbours: 9, Weight Function: distance, Algorithm: auto, accuracy: 0.957, Absolute Error: 0.17, Mean Squared Error: 0.824