Machine Learning Project 3 Report

Suraj Jayakumar-sxj240002

1) Collaborative filtering

```
print("Mean Absolute Error (MAE):", mae)
print("Root Mean Squared Error (RMSE):", rmse)
```

```
Mean Absolute Error (MAE): 0.7883677257380779
Root Mean Squared Error (RMSE): 0.9856831548353928
```

2) SVM parameter tuning

```
Error rate for kernel linear and C 0.1: 0.0528
Error rate for kernel linear and C 1: 0.0596
Error rate for kernel linear and C 10: 0.069
Error rate for kernel linear and C 100: 0.0742
Error rate for kernel rbf and C 0.1: 0.0405
Error rate for kernel rbf, C 1 and gamma 0.01: 0.0231
```

```
Error rate for kernel rbf and C 10: 0.0163
Error rate for kernel rbf and C 100: 0.0167
Error rate for kernel poly and C 0.1: 0.0428
Error rate for kernel poly and C 1: 0.0229
Error rate for kernel poly and C 10: 0.0214
Error rate for kernel poly and C 100: 0.0213
```

```
[33]: from sklearn.neighbors import KNeighborsClassifier
      import numpy as np
```

It is observed that:

a) Gaussian kernels yield the lowest error rate followed by polynomial followed by linear. This is because rbf (gaussian) and polynomial kernels yield a nonlinear decision boundary that does a better job separating the datapoints especially if it is not linearly separable and prone to noise. Linear kernel tends to underfit.

b) Small C values have a lower error rate because the model allows a softer margin with more tolerance for classification errors which leads to the reduction of overfitting. Larger C values in this case lead to higher classification errors since the MNIST dataset is noisy and hence more prone to overfitting.

c) Smaller gamma values produce a smoother, broader decision boundary leading to a more generalized model while large gamma values results in a tighter, more complex decision boundary and is prone to overfitting and higher sensitivity to noise in the data hence leading to a higher error rate.

3) KNN parameter tuning

```
Error rate for parameters {'n_neighbors': 3, 'weights': 'uniform'}: 0.0295
Error rate for parameters {'n_neighbors': 10, 'weights': 'uniform'}: 0.0335
Error rate for parameters {'n_neighbors': 20, 'weights': 'uniform'}: 0.0375
Error rate for parameters {'n_neighbors': 3, 'weights': 'distance', 'metric': 'cityblock'}: 0.036
Error rate for parameters {'n_neighbors': 3, 'weights': 'distance', 'metric': 'euclidean'}: 0.0283
Error rate for parameters {'n_neighbors': 3, 'weights': 'distance', 'metric': 'cosine'}: 0.0258
Error rate for parameters {'n_neighbors': 3, 'weights': 'distance', 'algorithm': 'brute'}: 0.0283
Error rate for parameters {'n_neighbors': 3, 'weights': 'distance', 'algorithm': 'ball_tree'}: 0.0283
```

[ ]:

```
Error rate for parameters {'n_neighbors': 1, 'weights': 'uniform'}: 0.0309
Error rate for parameters {'n_neighbors': 50, 'weights': 'uniform'}: 0.0466
```

It is observed that:

a) Error increases with increase in number of neighbours considered.
Low values (1-3): Can lead to high variance and overfitting, capturing noise in the data.
Moderate values (10): Often give balanced accuracy with less sensitivity to noise.
High values (20-50): Result in smoother decision boundaries, which may reduce overfitting but increase bias, potentially underfitting the model.
In the case of this dataset the error rate is lowest for 3 nearest neighbours

b) Uniform weighting vs distance weighting: Distance weighting improves the performance of the classifier when the dataset is unevenly spaced, giving more influence to the closer neighbours and reducing the impact of distant ones. Cosine distance weighting yields the lowest error rate for this dataset.

c) Algorithms like ball tree or kd tree are more efficient and take less time to fit than brute force especially on larger datasets like MNIST although they do not affect error rate