Fernando Colman

Fec190000

CS 6375

Project 2 Write-Up

# How To Run

Both programs are python notebooks meaning they can be ran without any setup on any program that accepts notebooks (Google Colab, Jupyter Notebooks, etc.) Furthermore, if the files can’t be accessed then I’ve also uploaded them to my Github at <https://github.com/FernandoColman/6375-projects/tree/main/project2>

# Part 1

The first thing I do is download the given txt files and then convert them into csv files for easy processing. For this part, I decided to split up the project into two main sections. The first section is dedicated to the helper functions. These functions are executed throughout the entire program and serve vital functions such as average ratings for active users and normal users, and ratings for a specific movie from a specific user. The second section is the two formulas that are referred to in the research paper. I have clearly labeled the two formulas with comments on the python notebook. Lastly, I iterate through the test dataset to compute the predictions given the first and second formulas. Due to the nature of this very large database, I limited the range for the algorithm just for the sake of time, however you can change it back to full values for analysis purposes.

# Part 2

## Testing Using Different Kernels/Penalization Parameter

**Linear**

C value: 1.0

Execution Time: 5m 32s

Accuracy: 0.9404

C value: 1.5

Execution Time: 6m 43s

Accuracy: 0.9387

C value: 0.5

Execution Time: 5ms 24s

Accuracy: 0.943

**RBF**

C value: 1.0

Execution Time: 6m 27s

Accuracy: 0.9792

C value: 1.5

Execution Time: 7m 11s

Accuracy: 0.9821

C value: 0.5

Execution Time: 7m 36s

Accuracy: 0.9759

**Polynomial**

C value: 1.0

Execution Time: 6ms 13s

Accuracy: 0.9771

C value: 1.5

Execution Time: 3ms 34s

Accuracy: 0.9778

C value: 0.5

Execution Time: 4m 47s Accuracy: 0.9731

**Sigmoid**

C value: 1.0

Execution Time: 9m 47s

Accuracy: 0.7759

## Testing Using Different MLP Hyperparameters

**RELU**

Hidden Layers: (256, 128, 64, 32)

Execution Time: 1m 59s

Accuracy: 0.9811

Hidden Layers: (100)

Execution Time: 1m 21s

Accuracy: 0.9791

**Using LBFGS Solver**

Hidden Layers: (256, 128, 64, 32)

Execution Time: 7m 44s

Accuracy: 0.9775

**Using SGD Solver**

Hidden Layers: (256, 128, 64, 32)

Execution Time: 9m 22s

Accuracy: 0.9764

**Identiy**

Hidden Layers: (256, 128, 64, 32)

Execution Time: 6m 42s

Accuracy: 0.9234

Hidden Layers: (100)

Execution Time: 2m 49s

Accuracy: 0.925

**Logistic**

Hidden Layers: (256, 128, 64, 32)

Execution Time: 4m 25s

Accuracy: 0.9785

Hidden Layers: (100)

Execution Time: 2m 1s

Accuracy: 0.9771

**Tanh**

Hidden Layers: (256, 128, 64, 32)

Execution Time: 3m 19s

Accuracy: 0.9824

Hidden Layers: (100)

Execution Time: 1m 42s

Accuracy: 0.9783

## Testing Using Different KNN Hyperparameters

**Neighbors = 3**

Algorithm: Ball Tree

Execution Time: 17m 45s

Accuracy: 0.9705

Algorithm: KD Tree

Execution Time: 20m 51s

Accuracy: 0.9705

Algorithm: Brute

Execution Time: 49s

Accuracy: 0.9705

**Neighbors = 5**

Algorithm: Ball Tree

Execution Time: 17m 40s

Accuracy: 0.9688

Algorithm: KD Tree

Execution Time: 20m 13s

Accuracy: 0.9678

Algorithm: Brute

Execution Time: 50s

Accuracy: 0.9829

**Neighbors = 7**

Algorithm: Ball Tree

Execution Time: 18m 32s

Accuracy: 0.9688

Algorithm: KD Tree

Execution Time: 22m 14s

Accuracy: 0.9690

Algorithm: Brute

Execution Time: 1m 17s

Accuracy: 0.9691

Algorithm: Brute

Weights: Distance

Execution Time: 1m 18s

Accuracy: 0.9691

## Final Report

In terms of error rate, the MLP Classifiers overall had a much better accuracy rate and relatively low execution times compared to the other classifiers. However, the RBF and Polynomial Kernels also performed well except with higher execution times. The very best combination of classifiers and hyperparameters in terms of accuracy was the KNN Classifier with 5 neighbors, the Brute algorithm, an execution time of 50s and an accuracy of 0.9829, however, the othe KNN instances had a really long execution time.