PDS Assignment 1

Alan Gaugler

U885853

I will attach this document that includes all the required plots and some explanatory notes.

Both questions 1 and 2 have been written using a GUI. All the input is done in the GUI and the output plots are returned to the GUI. As the GUI has limited space, some output information is also displayed on the command line interface (CLI), some is just additional information about the status and output of running the programs. It also does show the output information that is required. The required part that must be saved to a file is also saved in files which are included here.

In both programs, I have made a radio button option to save time by loading the default assignment files for each category automatically. Typing them in manually does of course work however:

**Note that when typing in the file names manually ensure the radio button “Enter manually” is selected. It can be easy to overlook and cause confusion.**

In both programs, I tried to minimize the number of functions in the program itself and place them all in the Data Module, however in both cases the functions run\_main and save\_file had to be placed in the programs because they require input from GUI widgets and display output information on the GUI. To do this, they could not be placed in the IO module. They did call several functions to the IO module.

The IO module has several functions 13 functions, including the main cluster classification algorithms. Also, many functions within the IO module are shared by both programs to save code duplication. Some of these functions include: read\_file, write\_file and calc\_distance.

It was stated in the assignment instructions that no other programs could be imported apart from tkinter, however I have confirmed with Ram that the module “random” can be imported.

I have provided many comments in all three program files to explain my workings as clearly as possible, in some cases in the cluster classifier algorithms, there are several lists containing various information which may make the workings of the program hard to follow. Both programs are working as expected. And exception handling has been added, to make them reliable.

The deliverables are:

1. The code files – attached in the zip
2. Txt files specifying test data labels for Q1 and the final clusters for Q2 I was not certain of the exact output so have done them to the best of my knowledge of the descriptions.
3. GUI figures that you can generate to illustrate your output/working. I will paste them in this document, as it will be easier to see them rather than opening them one by one.

The programs should be working fine, but below are screen shots in case they for some reason do not run.

**Question 1. Nearest Neighbour Classifier**

Exception handling. The program will not crash if valid data files are loaded.

It will detect if:

1. Files are not found in the working directory (either not placed there or a typo)
2. the format of the file is invalid,
3. the number of dimensions in the files are inconsistent.
4. When saving a file, if that file name already exists, the program will not crash, it will ask to enter a new name.

All other specified marking criteria have been addressed.

The algorithm functions as required and will work for any number of dimensions if the input files are in the specified format and consistent.

The data samples were read in from the three files and were stored as three lists of tuples as was specified in the assignment. The tuples are the coordinates of each data point and the lists contain the data points for the red, blue and unknown points. In the program, these lists are called red\_data, blue\_data and unknown\_data. They were used as arguments into the nearest\_neighbour\_classifier algorithm.

Examples of the GUI display for errors and the required output for each datafile are shown below:

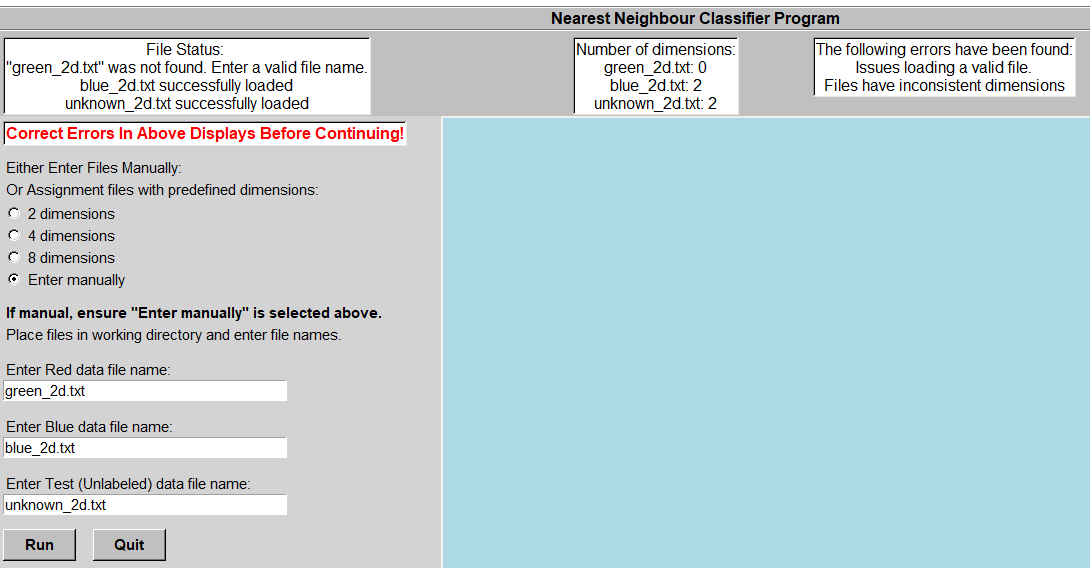


Figure 1: a file was not found.

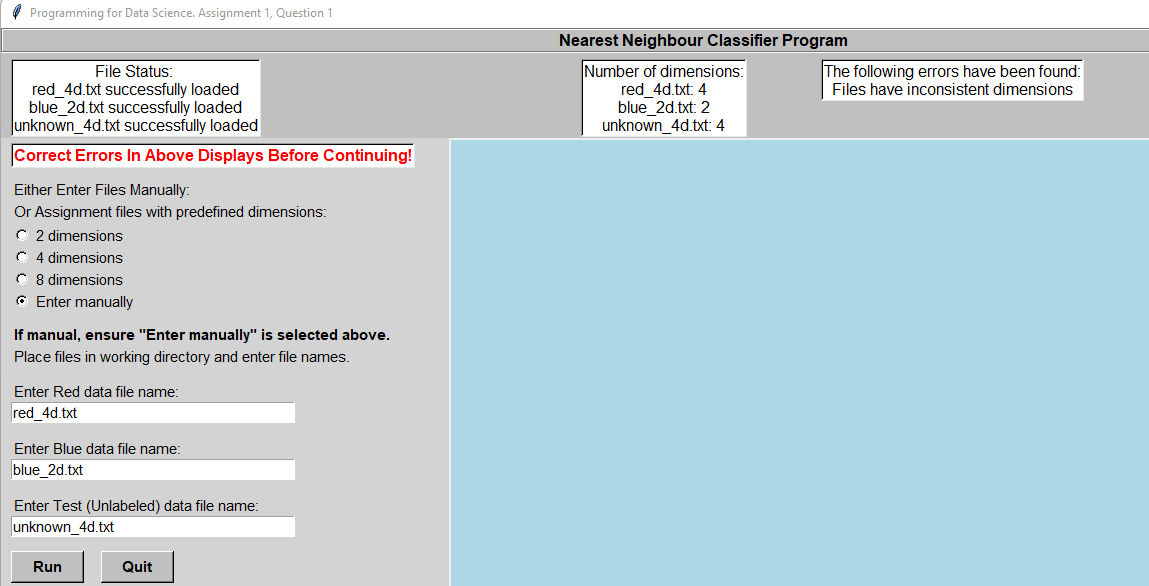


Figure 2 – Files have inconsistent dimensions.

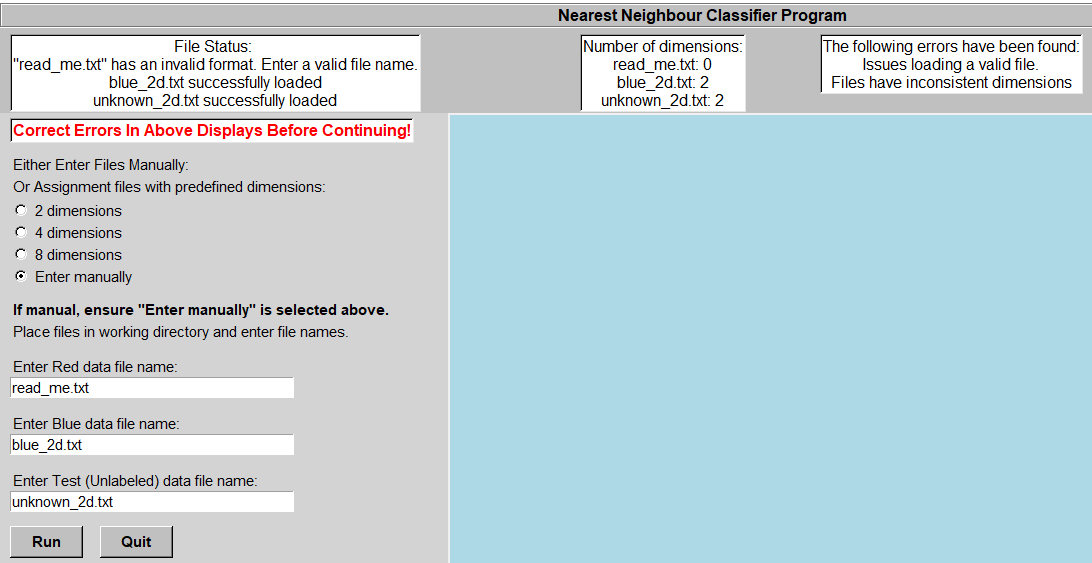


Figure 3: A file has an invalid format.

If the assignment files are placed in the working directory, they can automatically be detected and loaded. This is a shortcut. Lines are drawn between the unknown point and its closest red or blue data point. However, as most points are very close together in the 2D and 4D data sets, most lines are not visible or are very small.

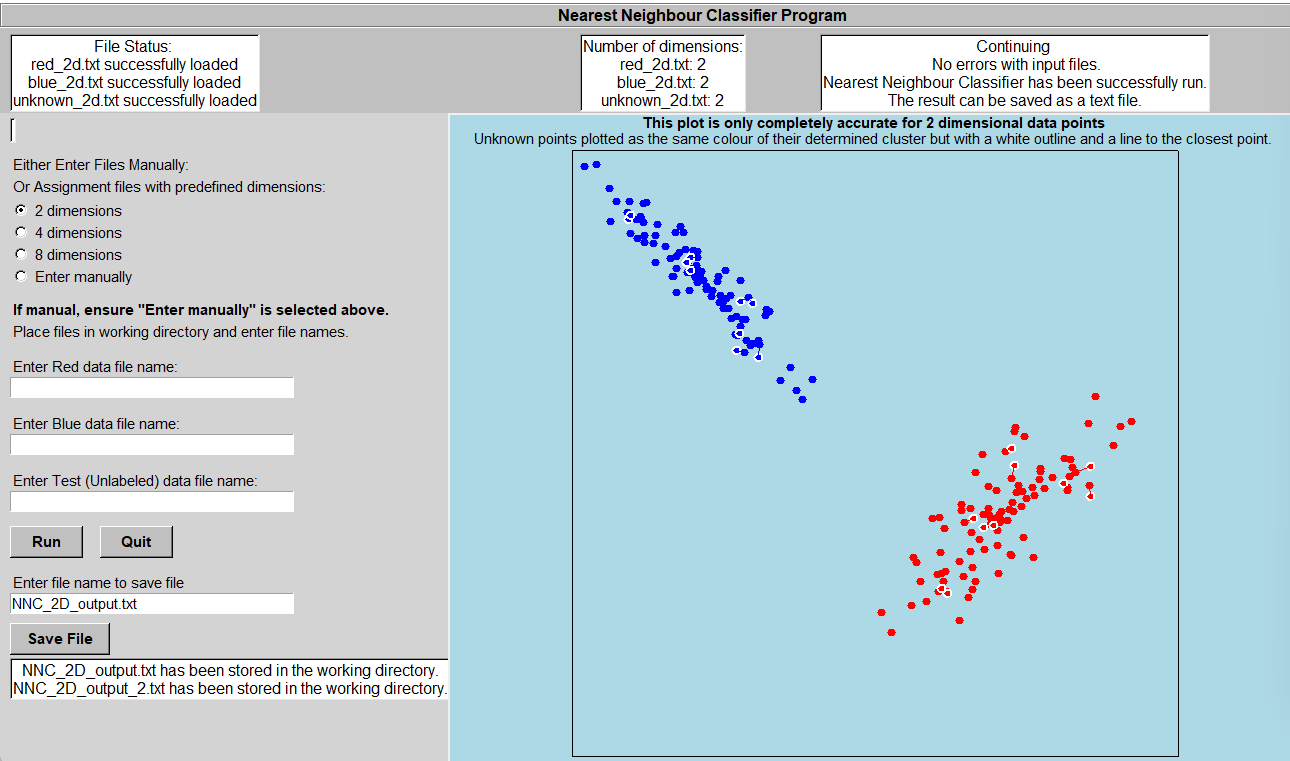


Figure 4: Output for all given assignment 2D files, loaded automatically. The status of saving the files is also shown.

Filenames: red\_2d.txt, blue\_2d.txt, unknown\_2d.txt

They can of course be loaded manually if placed in the working directory.

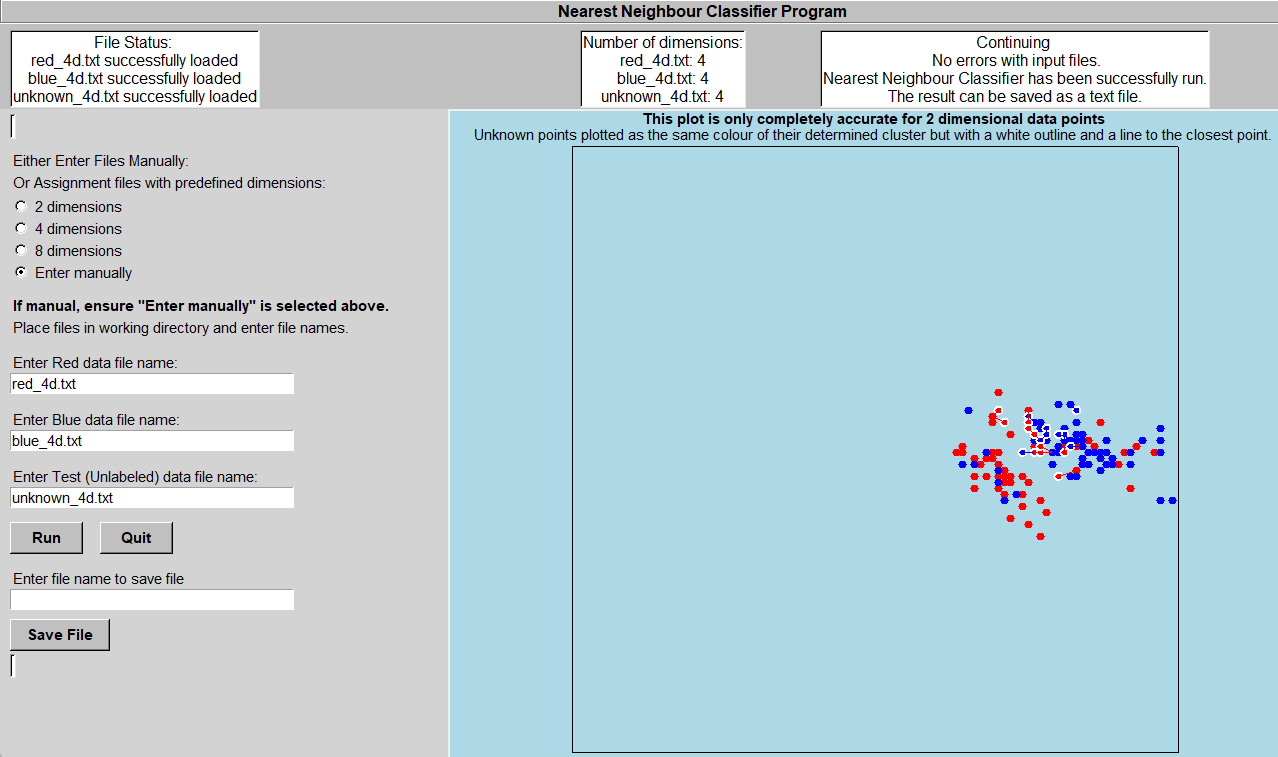


Figure 5: Output for all given assignment 4D files, entered manually.

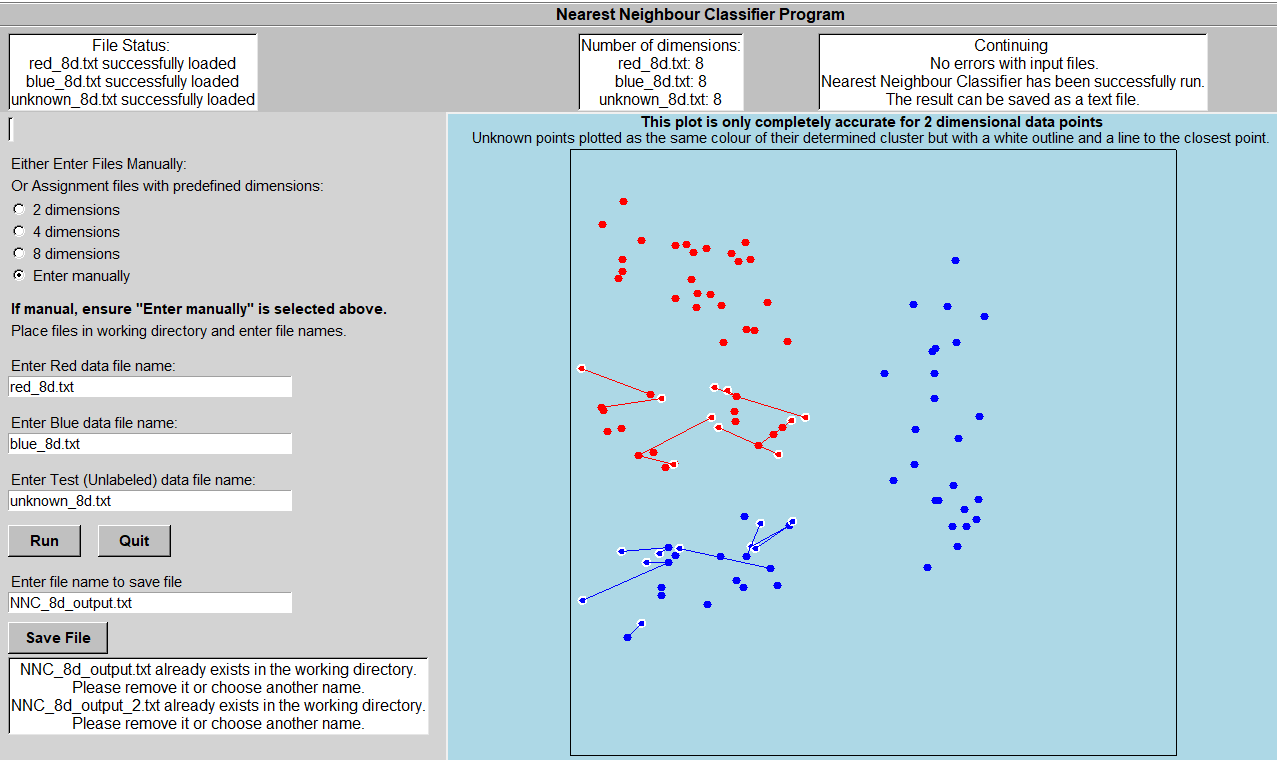


Figure 6: Output for all given assignment 8D files, entered manually.

Note when saving the filename and it already exists a message will be displayed.

Some required output information is also printed in the CLI. A screen shot is show below

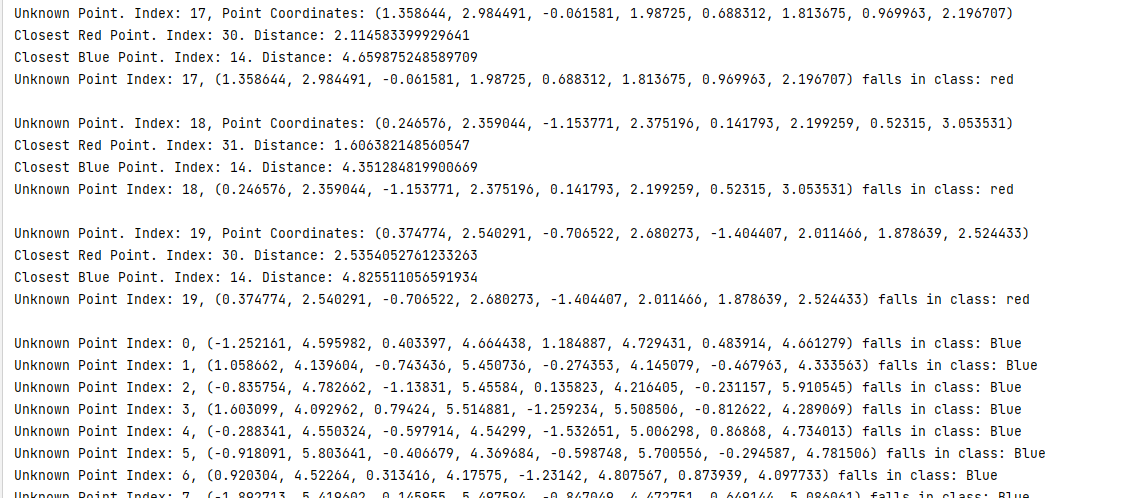


Figure 7. NNC Program output in the CLI.

**Question 2. K-Means Clustering**

Exception handling. The program will not crash if valid data files are loaded.

It will detect if:

1. Files are not found in the working directory (either not placed there or a typo).
2. the format of the file is invalid,
3. When saving a file, if that file name already exists, the program will not crash, it will ask to enter a new name.
4. Various incorrect inputs are checked for in the Entry slots. An incorrect filename, a file with the wrong format, typos, floats or other non integer entries or entries of the wrong integer values (<=1) may have been entered. These will be taken care of,

All other specified marking criteria have been addressed.

The algorithm functions as required and will work for any number of dimensions if the input file is in the specified format.

It will also work for any number of clusters too as will be demonstrated in the plots.

If the sum is less than the threshold then convergence will have been reached, however this program will continue to find a better convergence point. In the GUI an input is the number of iterations. The program will continue until this number of iterations is reached. When a convergence is reached its Sum of Squares Error (SSE) will be noted which is a total measure of the sum of the squared distance of all data points to their cluster centre. The smaller this number is, the better the cluster centres fit to the data. After convergence, the algorithm will reset random data points and continue the whole process over and over again until the specified number of iterations is reached. Additionally, very often it will occur that one or more random cluster centres have zero data points associated with them, this would cause a divide by zero error when calculating the distance of each dimension of each data point to its centre. If this occurs, the starting points of the cluster centres will be randomly chosen again until this error does not occur and convergence of the cluster centres can be successfully carried out.

The SSE is displayed in the upper status bar. The lower the score, the better the cluster centroids’ fit. Running more iterations will generally improve the chances of getting a better score.

The number of dimensions and the number of samples in the input dataset are mentioned in the central upper status text window.

The data samples have been returned from the IO module to the K-Means program as tuples which was required. They were used as arguments to the create\_K\_Means\_plot function, they are called list\_of\_best\_fit and best\_centroids.

The output information in the text files show the cluster centroid and all the datapoints within its cluster. The colour is mentioned so each cluster can easily be referenced by looking at the plot.

Examples of the GUI display for errors and the required output for each datafile are shown below:

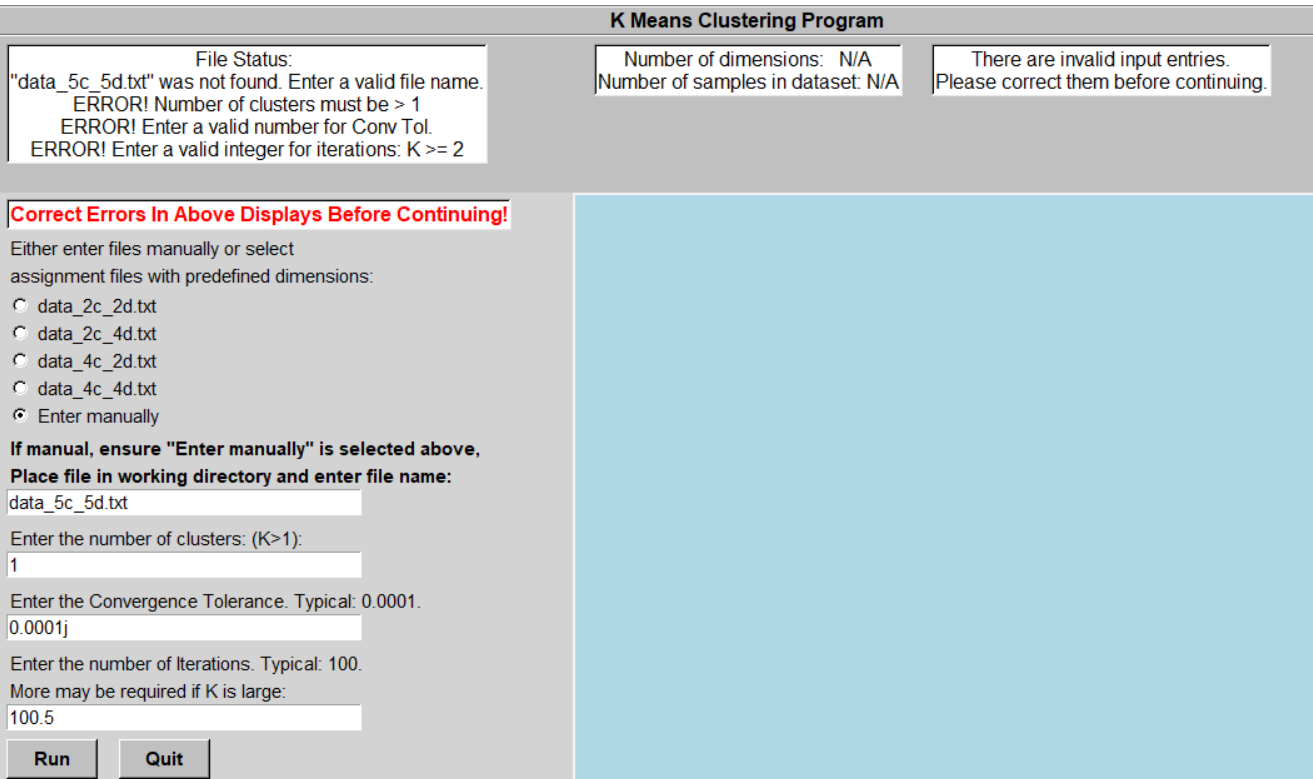


Figure 8. Various incorrect entries have been made in the 4 input entries. All have been handled and the use can reenter the correct input without crashing the program.

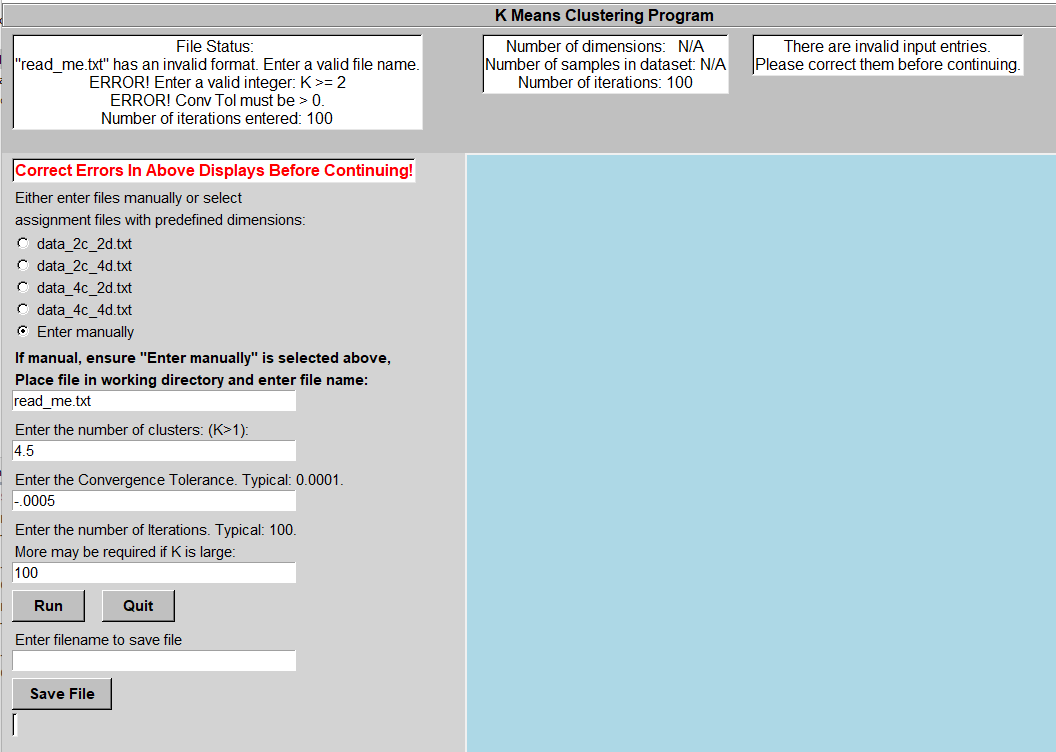


Figure 9: More incorrect entries are handled correctly, without the program crashing.

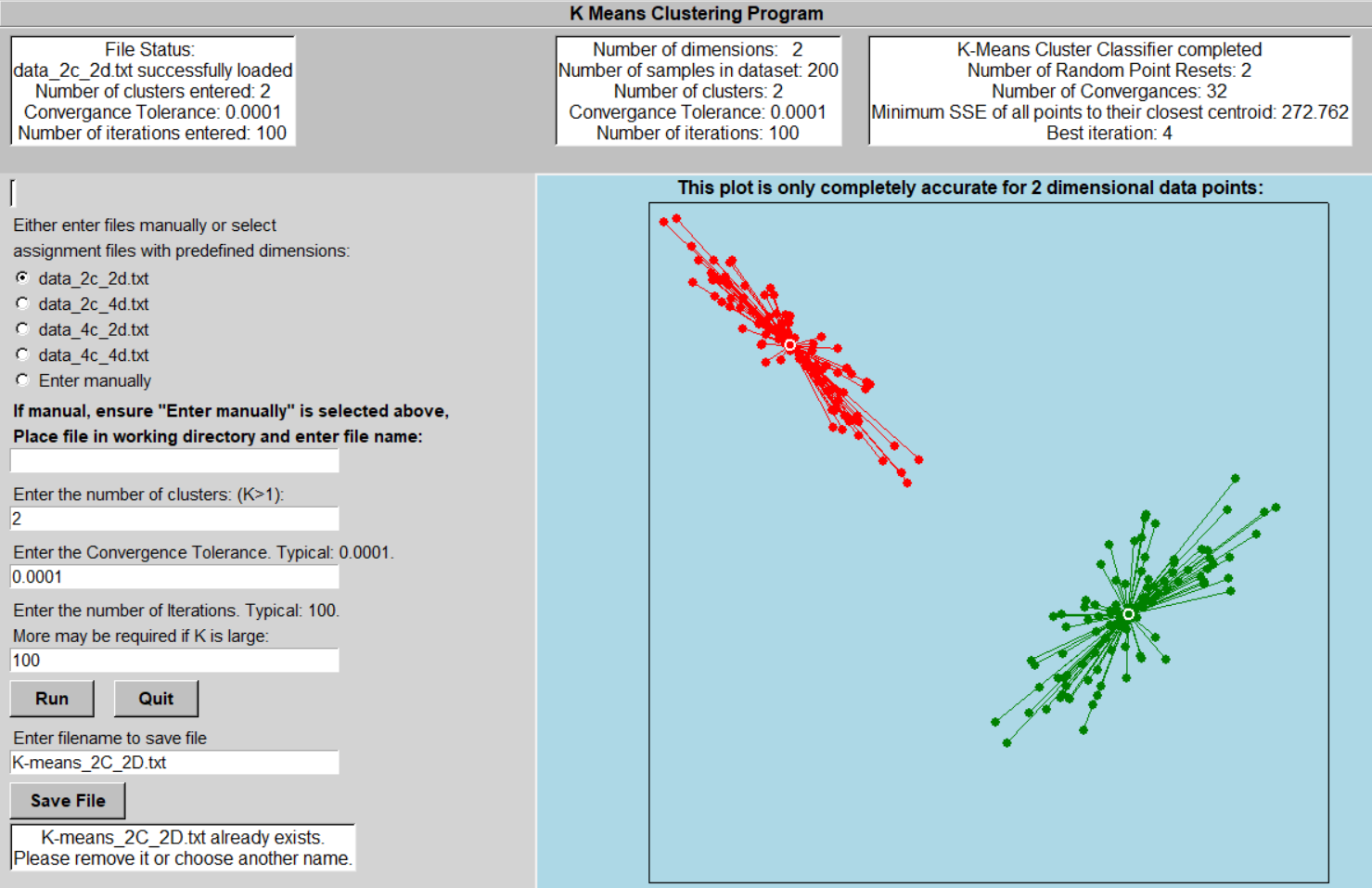


Figure 10. The output of data\_2c\_2d.txt entered from the radio button. Note that if the file already exists it cannot be overwritten.

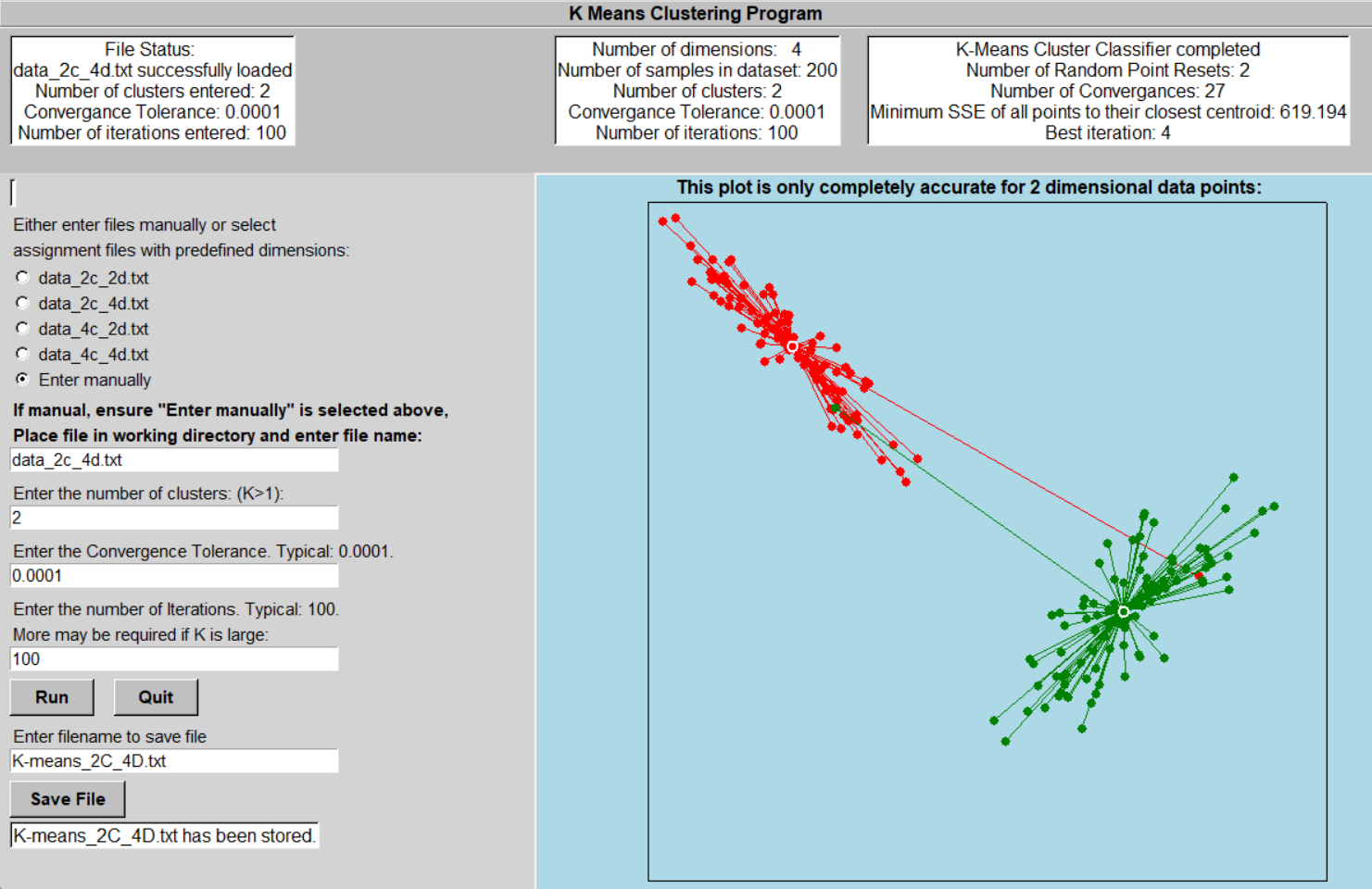


Figure 11. The output of data\_2c\_4d.txt entered manually. As this is 4D data, the plot is not completely accurate, only the first two dimensions of the datapoints were plotted.

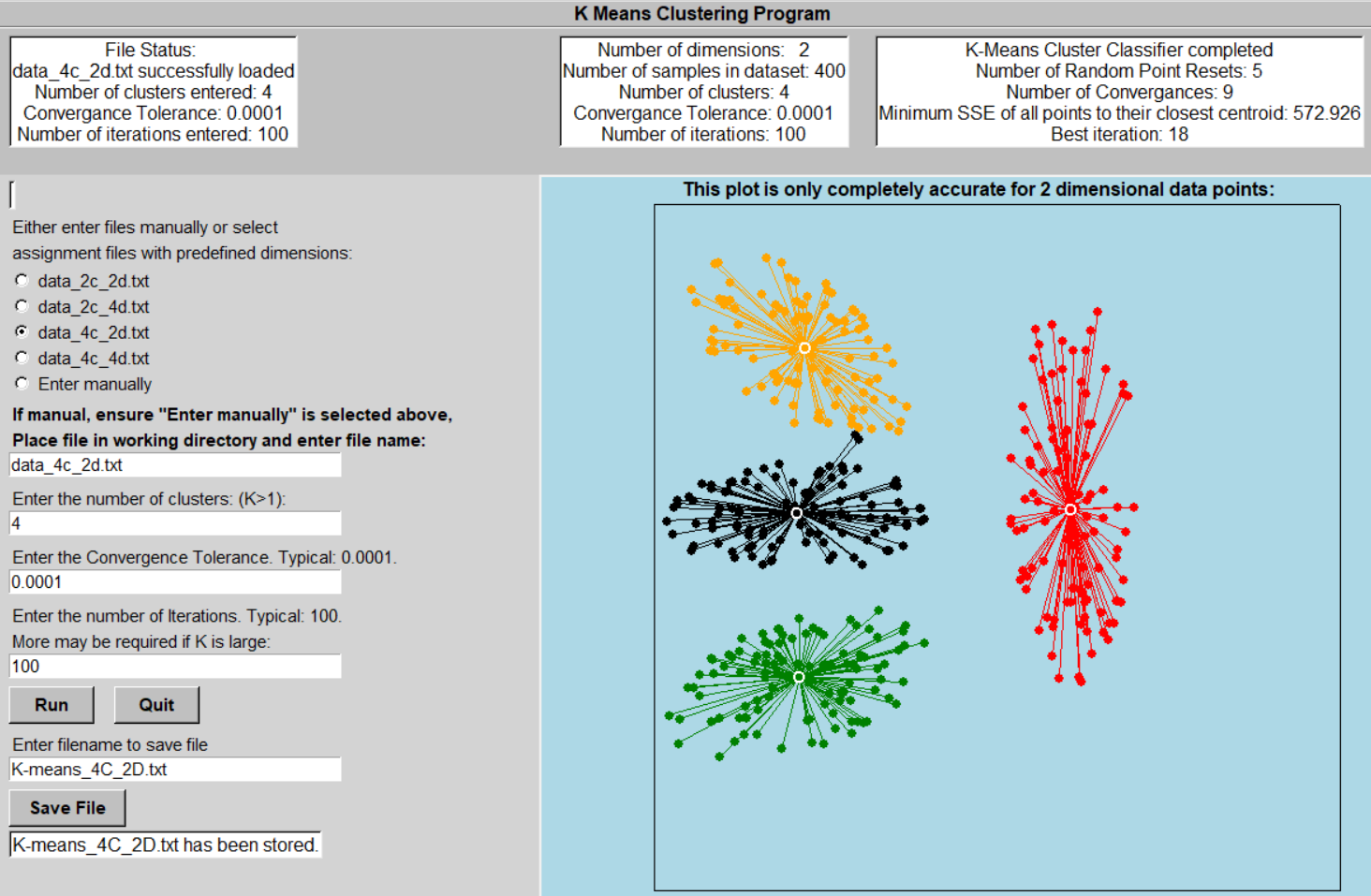


Figure 12. The output of data\_4c\_2d.txt entered manually. Note the minimum SSE is displayed. This will be useful for comparing iterations, to see which was the best. The lowest SSE over all the iterations was selected as the final cluster centroids.

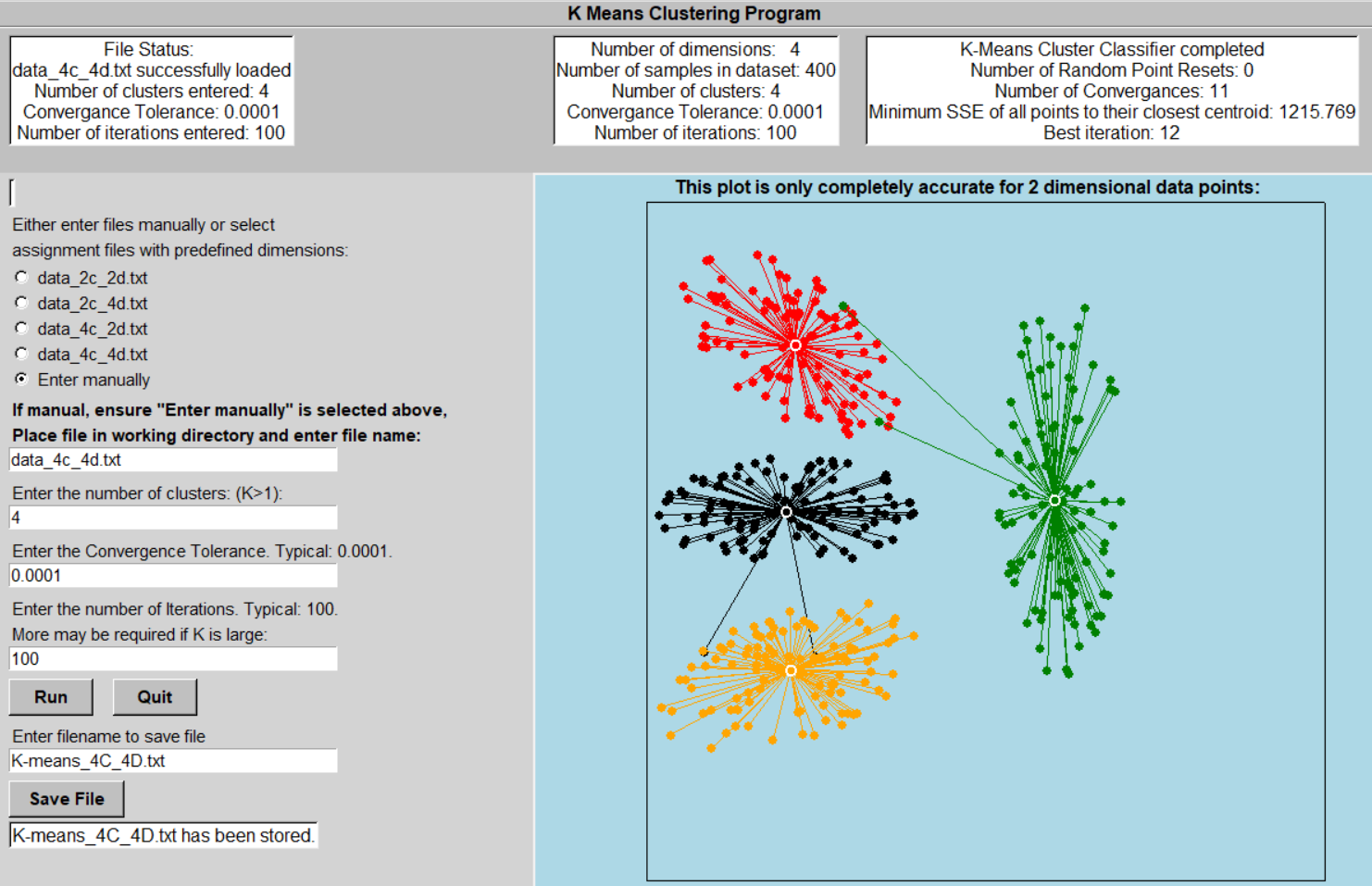


Figure 13. The output of data\_4c\_4d.txt entered manually. Only the first two dimension are plotted.

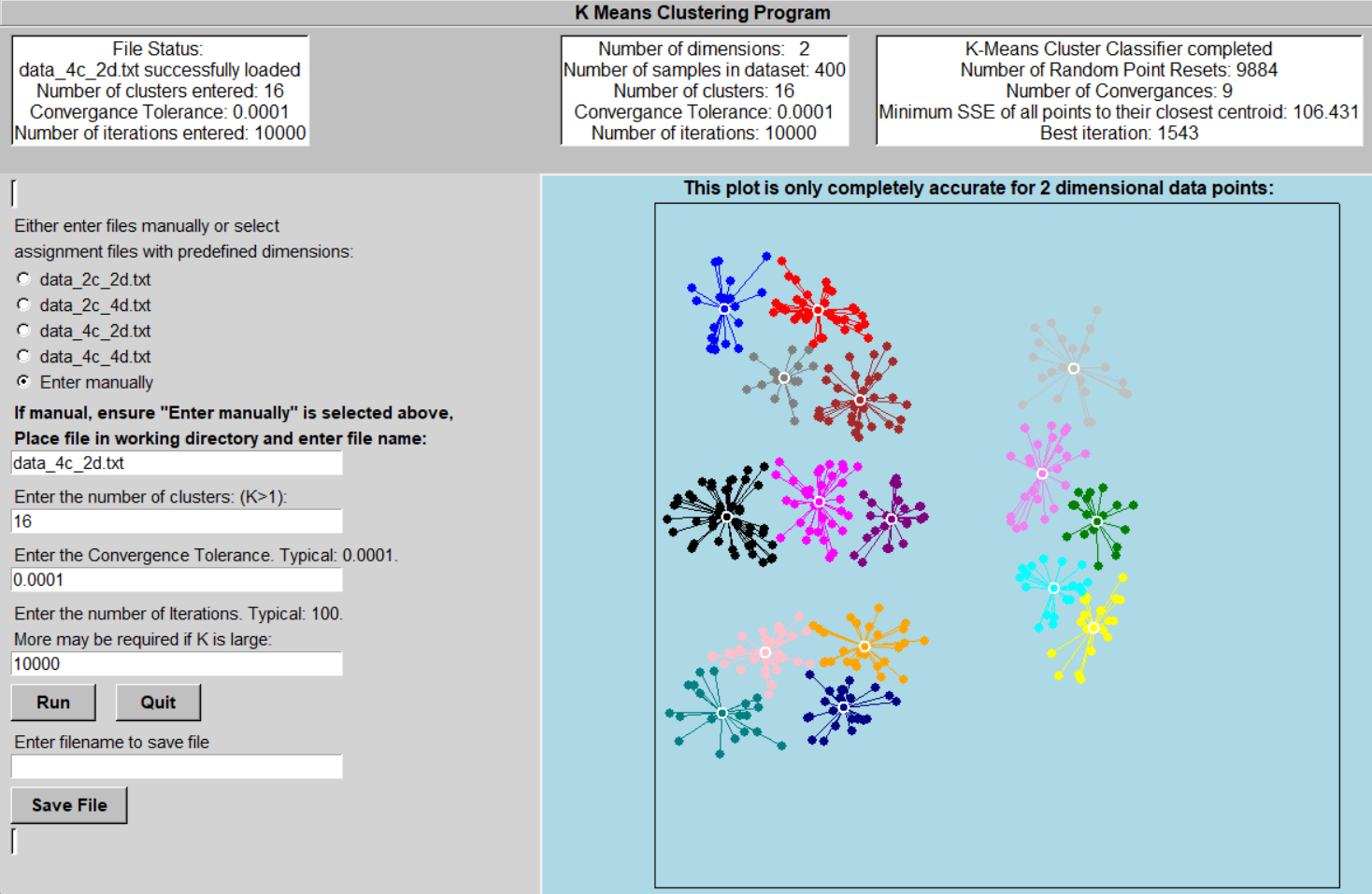


Figure 14: Finally, it was specified that any number of clusters can be run. This is an extreme example of 16 clusters over 10,000 iterations. So many iterations were required for this dataset as the probability of all 16 clusters with random start points having points associated with them is extremely low. Out of the 10,000 iterations there were 9884 Random Point resets and only 9 convergences.

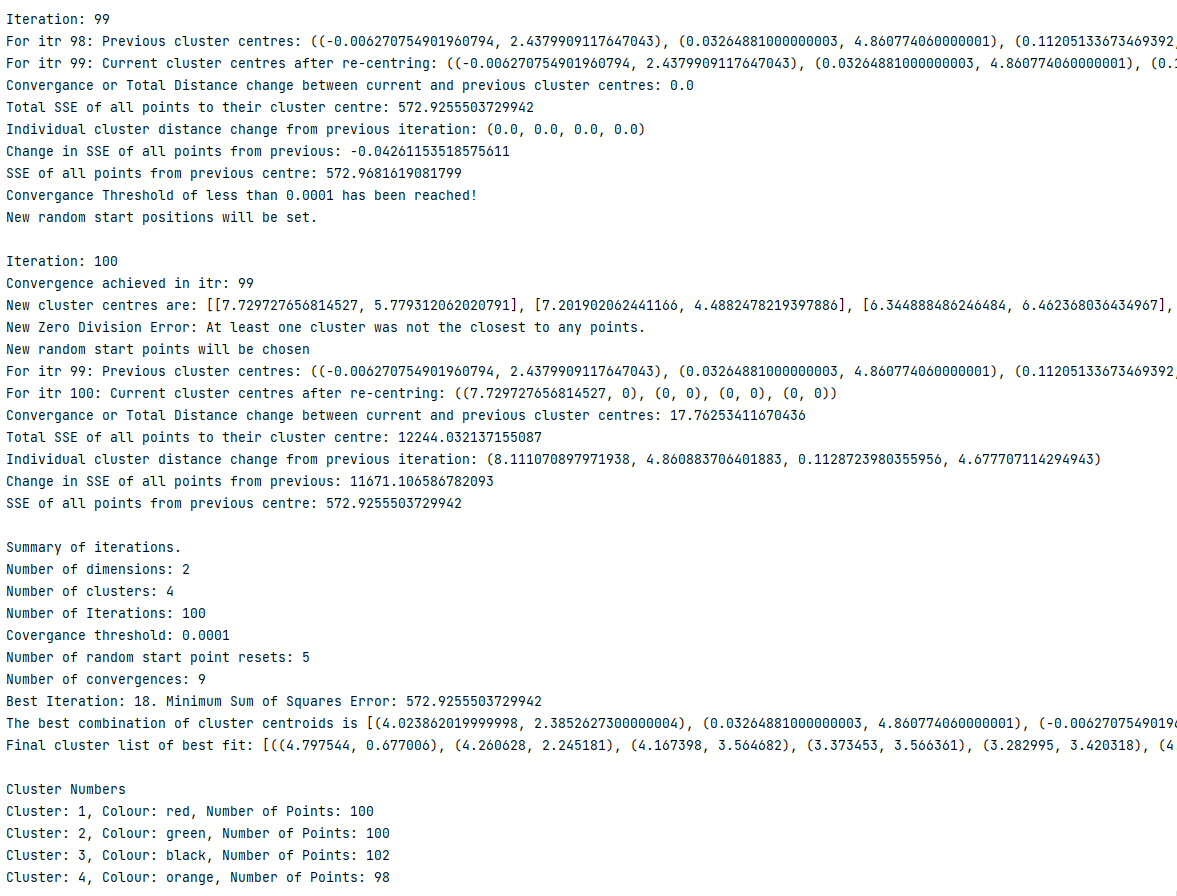


Figure 15: Iteration and summary data is printed out on the CLI. The required text output is also printed out on the CLI but is also attached as the output files for this report. i.e. K-means\_4C\_2D.txt.