Object Oriented Programming Lab Final Report

Alihan BOZKIR 151220182032

Uml Diagram:

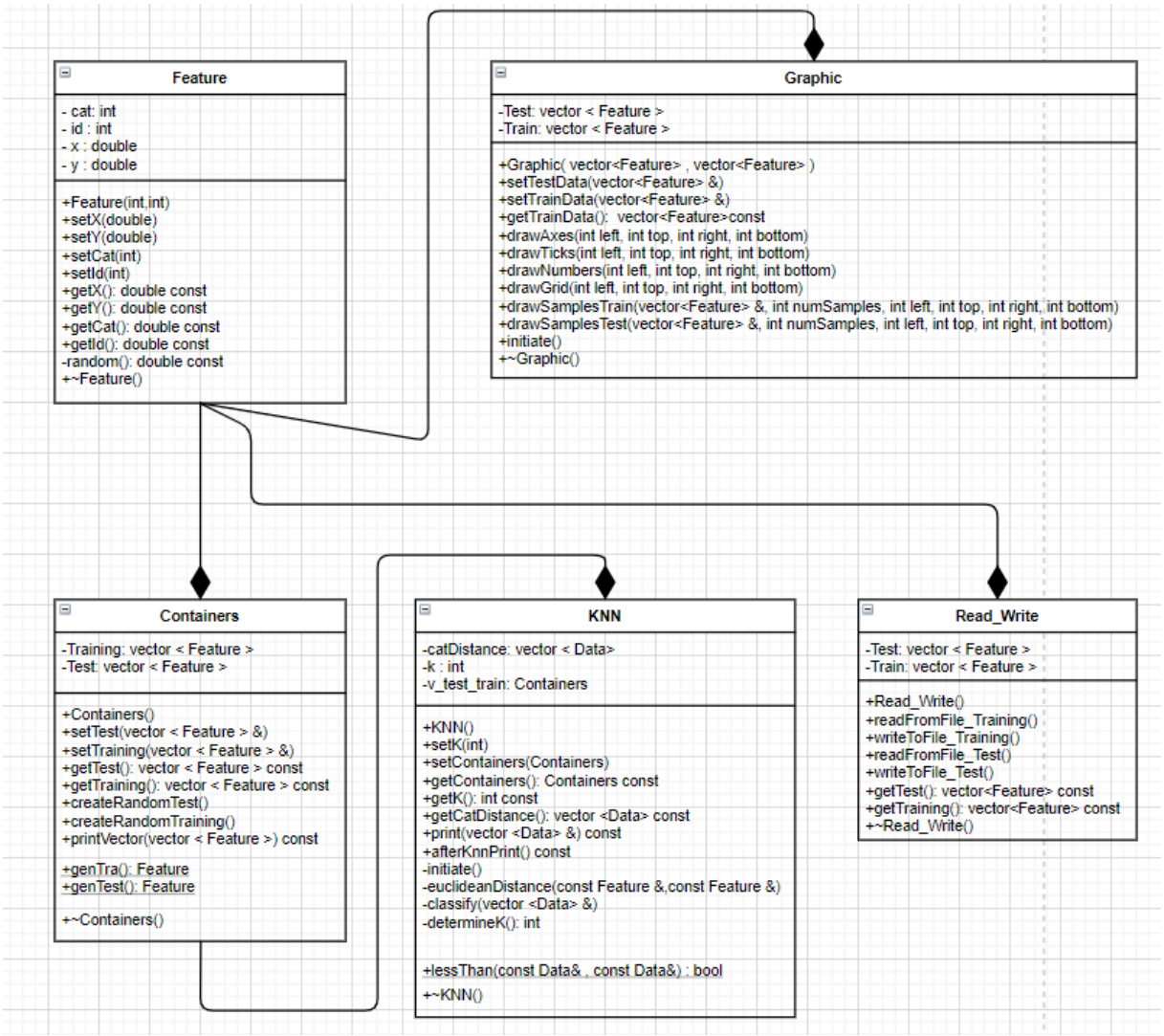

Figure 1

Feature Class:

The constructor takes cat and id as arguments. In this class, in addition to the cat, id, x and y data members that I will create the elements of my data sets, and the set() and get() functions for these data members, there is the random() function for the x and y data members. There is an overloaded output stream function in this class. While the object is being created, x and y values are randomly assigned in its constructor.

Container Class: (has Feature Class)

In the Constructor, it offers the user a choice about the creation of Datasets. In this class, there are 2 vector data members named Training and Test, which contain Feature objects. In addition to the set() and get() functions for these data members in the class, there are 2 functions called createRandomTest() and createRandomTraining() to randomly assign these vectors. These functions use the generate function from the Algorithm library and specifically genTra() for these functions. and genTest() predicate functions are used. And finally, there is the printVector function that prints vectors. Since there are Feature objects in these vectors, the overloaded function in the Feature class is called automatically when these vectors are wanted to be printed.

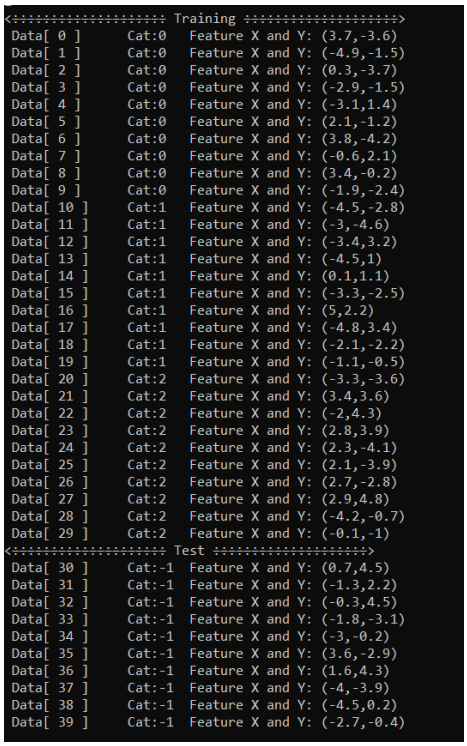

Figure 2

KNN Class: (has Container Class)

In this class, there are data members named Distance, which contains a data structure named Data, with int-valued k and Container object named v_test_train. There are set() and get() functions for these data members in the class. The print(vector &) function uses the overloaded output stream function to print the Data structure. The afterKnnPrint function is used to print data sets to the screen after the KNN algorithm. In summary, this function prints the Test and Training vectors over the Container object through the printVector function in the Container Class.

The initate function is called when the object is created. In this function, firstly the k value is taken by the user. Afterwards, each Test data is sent to the euclideanDistance function with the Training data. The euclideanDistance function calculates the distance between two data and returns this distance. This distance is assigned to the Distance local variable. At the same time, the category of the Training data is assigned to the cat local variable.Data structure is created with distance and cat values (This data structure holds category and distance respectively) and this structure is sent into the tempCatDistance vector with the push_back function. This is done with Test data and 30 Training data. Before moving on to the other test data, the data structures in the tempCatDistance vector with 30 elements are sorted with the help of the predicate lessThan function (This function sorts the vector elements from smallest to largest according to their distances). The sorted tempCatDistance vector is first printed and then sent to the classify function. Within the Classify function, it first counts the categories of the first k values of the tempCatDistance vector according to the k value, increasing the indexes of the 3-element cat[]={0,0,0} array. After this count is done, the maximum category is determined. The category determination algorithm is done in 3 different ways:

1. If the maximum number of categories is a single category, the category of the Test data is assigned to that category.

2.If the maximum number of categories is two different categories, the category of the Test data is randomly selected among these two categories.

3. When the three categories are equal in number, the category of the Test data is randomly selected among these three categories. It returns to the initate function again and this cycle continues until all test data are applied.

```
[0]: (2, 2.18403)    [1]: (2, 2.22036)    [2]: (2, 2.7074)    [3]: (0, 2.72947)    [4]: (2, 2.84605)
[5]: (1, 3.45254)    [6]: (1, 4.30116)    [7]: (1, 4.87647)    [8]: (0, 4.90408)    [9]: (1, 5.31413)
[10]: (0, 5.42033)    [11]: (2, 5.55788)    [12]: (1, 5.60892)    [13]: (0, 5.86941)    [14]: (1, 6.26817)
[15]: (0, 6.99714)    [16]: (2, 7.14493)    [17]: (1, 7.26154)    [18]: (0, 7.3736)    [19]: (2, 7.56902)
[20]: (1, 8.06226)    [21]: (0, 8.20731)    [22]: (0, 8.20975)    [23]: (2, 8.51587)    [24]: (0, 8.63771)
[25]: (2, 8.74757)    [26]: (1, 8.9627)    [27]: (2, 9.03383)    [28]: (0, 9.2358)    [29]: (1, 9.82344)


k=3
Category 0: 0   Category 1: 0   Category 2: 3

 Data[ 30 ]     Cat:2   Feature X and Y: (0.7,4.5)
 Data[ 31 ]     Cat:-1  Feature X and Y: (-1.3,2.2)
 Data[ 32 ]     Cat:-1  Feature X and Y: (-0.3,4.5)
 Data[ 33 ]     Cat:-1  Feature X and Y: (-1.8,-3.1)
 Data[ 34 ]     Cat:-1  Feature X and Y: (-3,-0.2)
 Data[ 35 ]     Cat:-1  Feature X and Y: (3.6,-2.9)
 Data[ 36 ]     Cat:-1  Feature X and Y: (1.6,4.3)
 Data[ 37 ]     Cat:-1  Feature X and Y: (-4,-3.9)
 Data[ 38 ]     Cat:-1  Feature X and Y: (-4.5,0.2)
 Data[ 39 ]     Cat:-1  Feature X and Y: (-2.7,-0.4)

--------------------------------------------------------------------------------------------------------------
```

Figure 3

Read_Write Class: (has Feature Class)

WriteToFile_Training, writeToFile_Test, readFromFile_Test and readFromFile_Training functions are called respectively in the constructor. This class asks the user to enter 30 training data from the keyboard in order (Category, x, y) respectively and saves it to the file with the extension "Training_From_Client.dat". Then, it asks the user to enter 10 test data in order (x , y) from the keyboard, respectively, and saves it to the file with the extension "Test_From_Client.dat". The data read from these files are initialized to the vector Training and vector Test data members in the Read_Write class. This class is written to be called by creating an object in the Container class. This class has only one purpose. When the user wants to enter the values of the Test and Training vectors in the Container, the object is created and the vectors in the Read_Write class are assigned to the vectors in the Container class.

```
For random data set, enter      --->  1
to specify the dataset, enter   --->  2
```

Figure 4

Graphic Class: (has Feature Class)

The graphic.h library has been integrated into the Dev-C++ IDE to be used in this class. The purpose of the class is to print the Training data sets in different colors according to their categories, according to x and y coordinates. It is suppressed as seen in Figure(5). Small circles represent Training datasets, larger circles represent Test datasets.

Figure 5

Changes:

Our Training and Test vectors were sent as arguments to the Constructor of the KNN class, and our vectors were redefined as data members in the Class and their values were assigned using the member initializer list in the constructor. Upon the feedback received, it was understood that this process was unnecessary, and instead of creating the Constructor object in the main, it was created as a data member in the KNN class, and the Test and Training data sets used in the KNN class were accessed via the set and get functions over this object.

As a result, at the end of the program here is the final modified Data Sets as seen in Figure(6).

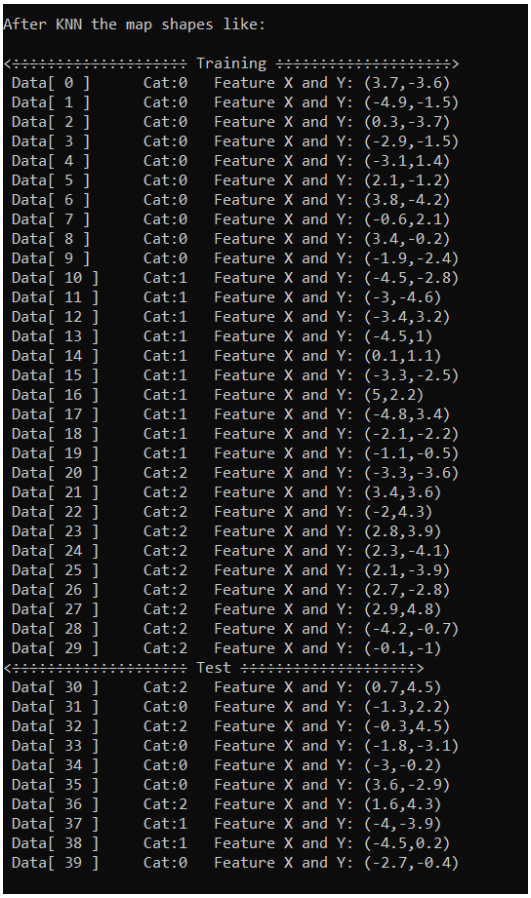

Figure 6