Assignment 2

MMS131 - Introduction to Artificial Intelligence
VT-23

Benjamin Elm Jonsson

Mechanical Engineering
Chalmers Institute of Technology
Gothenburg, Sweden

Contents

2

2.1

The following task is to create a decision tree usign the ID3 algorithm. This is done by firstly calculating the information gain for each variable. The variable with the highest information gain is chosen as the first variable in the tree. Then the variables follow in order of information gain.

Information gain is calculated as the following.

$$G(age|T) = H(T) - I(age)$$

Where age acts as an example variable. In order to calculate the information gain the entropy of the training set, $H(T)$, is needed as well as the weighted average of the entropy for the specific variable. In the example above this is shown as $I(age)$. To calculate the entropy the following equation is used.

$$H(a) = -\sum_k P(a_k)log_2(P(a_k)) \tag{1}$$

From this the entropy for the training set can be calculated.

$$H(T) = -\frac{10}{24}log_2\left(\frac{10}{24}\right) - \frac{14}{24}log_2\left(\frac{14}{24}\right) = 0,979868 \tag{2}$$

Then it is needed to calculate the entropy for each specific case for a variable. This means the entropy for *young*, where only the age variable is considered. This is done for each case.

$$H(young) = -\frac{3}{8}log_2\left(\frac{3}{8}\right) - \frac{5}{8}log_2\left(\frac{5}{8}\right) = 0,954434 \tag{3}$$

$$H(medium) = -\frac{3}{8}log_2\left(\frac{3}{8}\right) - \frac{5}{8}log_2\left(\frac{5}{8}\right) = 0,954434 \tag{4}$$

$$H(elderly) = -\frac{4}{8}log_2\left(\frac{4}{8}\right) - \frac{4}{24}log_2\left(\frac{4}{8}\right) = 1 \tag{5}$$

After this the weighted average can be calculated.

$$I(age) = \frac{8}{24}H(young) + \frac{8}{24}H(medium) + \frac{8}{24}H(elderly) = 0,969622 \tag{6}$$

And finally the information gain for the variable age.

$$G(age, T) = H(T) - I(age) = 0,010246 \tag{7}$$

This method is then repeated for each variable. The calculations are shown below. The next variable to be considered is *type*.

$$H(Myopia) = -\frac{8}{12}log_2\left(\frac{8}{12}\right) - \frac{4}{12}log_2\left(\frac{4}{12}\right) = 0,9182958 \tag{8}$$

$$H(Hypermetropia) = -\frac{2}{12}log_2\left(\frac{2}{12}\right) - \frac{10}{12}log_2\left(\frac{10}{12}\right) = 0,65002242 \tag{9}$$

$$\Rightarrow I(type) = \frac{12}{24}H(Myopia) + \frac{12}{24}H(Hypermetropia) = 0,78415911 \tag{10}$$

$$\Rightarrow G(type, T) = H(T) - I(type) = 0,19560889 \tag{11}$$

Next variable is *Astigmatism*.

$$H(Yes) = -\frac{3}{12}log_2\left(\frac{3}{12}\right) - \frac{9}{12}log_2\left(\frac{9}{12}\right) = 0,8112781244 \tag{12}$$

$$H(No) = -\frac{7}{12}log_2\left(\frac{7}{12}\right) - \frac{5}{12}log_2\left(\frac{5}{12}\right) = 0,9798687566 \tag{13}$$

$$\Rightarrow I(Astigmatism) = \frac{1}{2}H(Yes) + \frac{1}{2}H(No) = 0,8955734405 \tag{14}$$

$$\Rightarrow G(Astigmatism, T) = H(T) - I(Astigmatism) = 0,0842945595 \tag{15}$$

Next variable is *Tear production*.

$$H(Reduced) = -\frac{2}{10}log_2\left(\frac{2}{10}\right) - \frac{10}{12}log_2\left(\frac{10}{12}\right) = 0,65002242 \tag{16}$$

$$H(Normal) = -\frac{8}{12}log_2\left(\frac{8}{12}\right) - \frac{4}{12}log_2\left(\frac{4}{12}\right) = 0,91829583 \tag{17}$$

$$\Rightarrow I(Tear\ production) = \frac{1}{2}H(Reduced) + \frac{1}{2}H(Normal) = 0,784159125 \tag{18}$$

$$\Rightarrow G(Tear\ production, T) = H(T) - I(type) = 0,195708875 \tag{19}$$

From these calculations it is clear that *Tear production* is to be selected as the first variable.

Iterate this method using subsets of the data.

$$G(age_{reduced}) = H(T_{reduced}) - I(age) = 0,1097033706 \tag{20}$$

$$G(type_{reduced}) = 0,190875 \tag{21}$$

$$G(Astig_{reduced}) = 0,190875 \tag{22}$$

Choose Astigmatism or Type. I chose Astigmatism.

$$G(Type_{no,reduced}) = 0,459148 \tag{23}$$

$$G(Age_{no,reduced}) = 0,25 \tag{24}$$

Choose Type. The entire left part of the tree can now be constructed. Moving on to the right with the same metholodgy.

$$G(age_{normal}) = H(T_{reduced}) - I(age) = 0,00441105 \tag{25}$$

$$G(type_{normal}) = 0,41829 \tag{26}$$

$$G(Astig_{normal}) = 0,09 \tag{27}$$

Choose Type.

$$G(Astigmatismhyper, normal) = 0,45 \tag{28}$$

$$G(Age_{hyper,normal}) = 0,25 \tag{29}$$

Choose Astigmatism. With this the whole tree can be constructed as can be seen in the figure below.

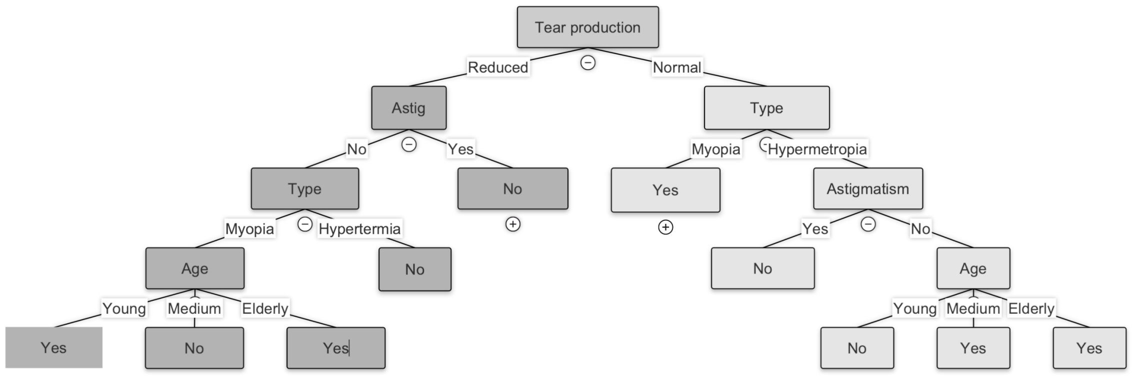


Figure 1: Decision tree, derived using ID3

2.2 a)

The following task consideres the clustering of data. In order to find the final centroid positions and therefore the centers of each cluster Lloyds algorithm is utelized. This algoritm takes starting guesses for k centroids. In this case 3. It then calculates the distance from each data point to these k centroids. Then for each data point the closest centroid is selected and labeled to the centroid. The centroids position is then updated as the median position of all the data points with its label. Iterating this process until the centroids move less than the limit, 0,001, gives the final positions of the centroids. These positions are the following.

$$C_1 = (-3,862; -1,325) \tag{30}$$
$$C_2 = (4,109; -1,678) \tag{31}$$
$$C_3 = (0,186; 7,347) \tag{32}$$

This can be seen in the figure below.

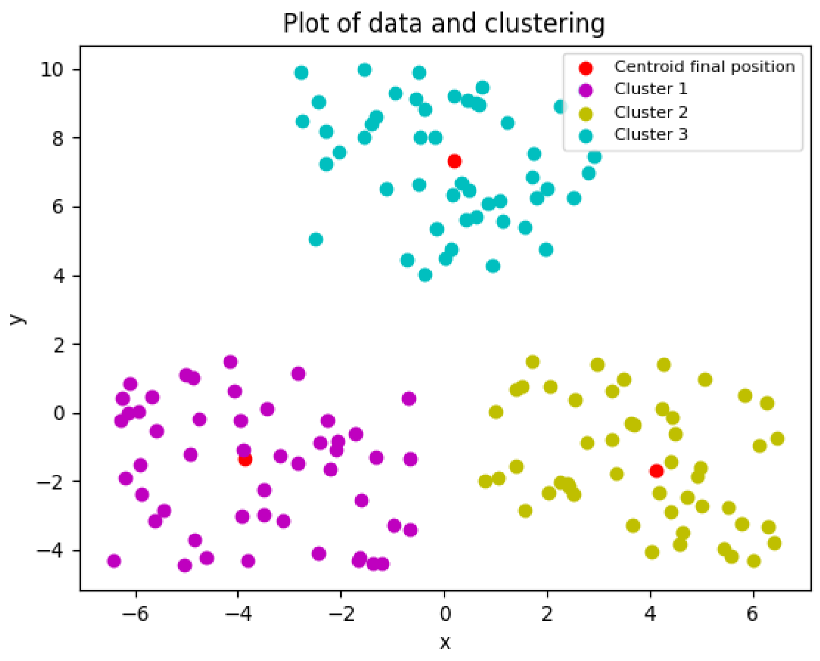


Figure 2: Clustering, centroids final position

In the figure above the red points are the centroids and the others are the given data point. The color is determined by which cluster they belong to.

2.2 b)

In the next task a new point, $(0, 0)$, is introduced. The task is to determine which cluster this point belongs to. This is done by using the kNN algorithm. This algorithm works by calculating the distance from each data point to the new point. Then the k nearest neighbors are examined. This meaning the k nearest data points. The label for these k neighbors are then counted and the majority vote is chosen as the label/class for the new point.

In the code this is done for $k = [3, 7, 11]$ and for the norms $i = [1, 2, 3]$. In this case i determines which distance to use. $i = 1$ means that manhattan distance is to be used according to the following formula.

$$d(P_i, P_j) = \sum_{s=1}^{n} \left(|f_{is} - f_{js}|^i\right)^{(1/i)} \tag{33}$$

To begin with the new point is plotted. This can be seen below.

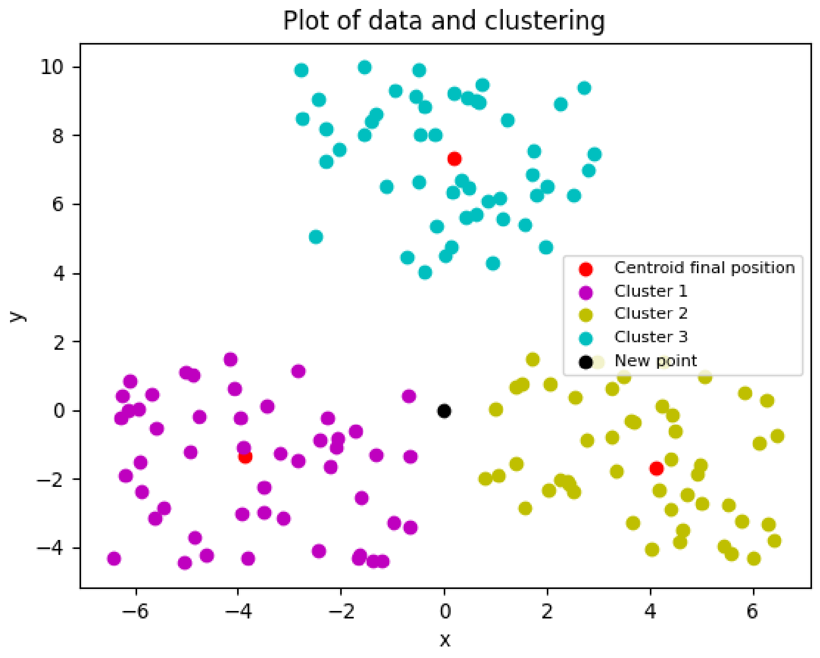


Figure 3: New point introduced

The algorithm above is then utelized for each combination of k and i. It became obvious that i does not impact the classification of the new point. Therefore only the 3 plots with varying k are shown.

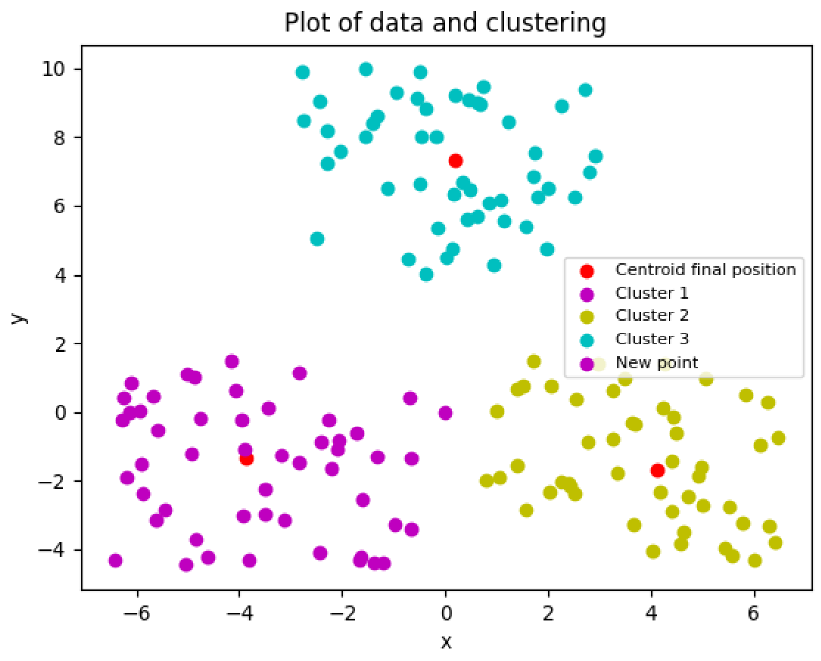


Figure 4: kNN classification, k = 3

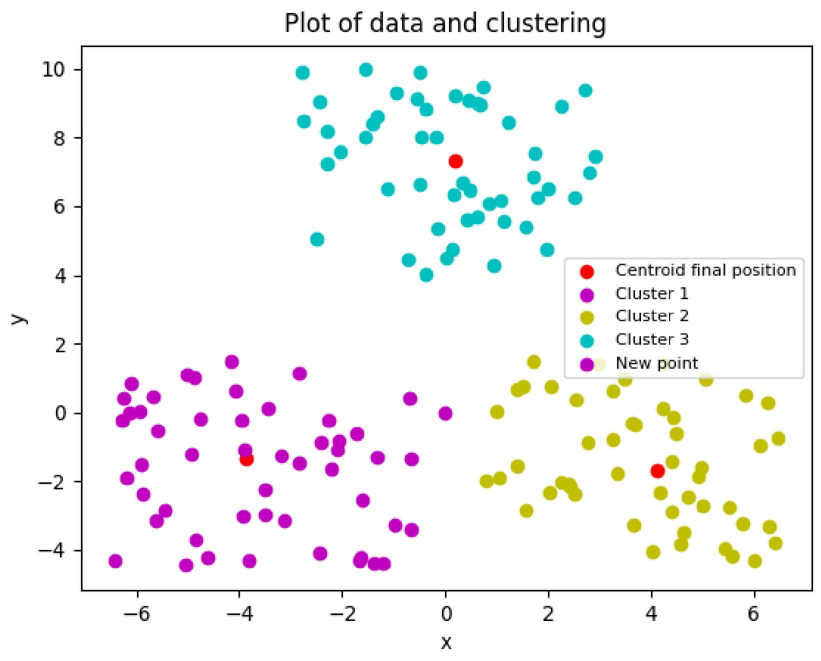


Figure 5: kNN classification, k = 7

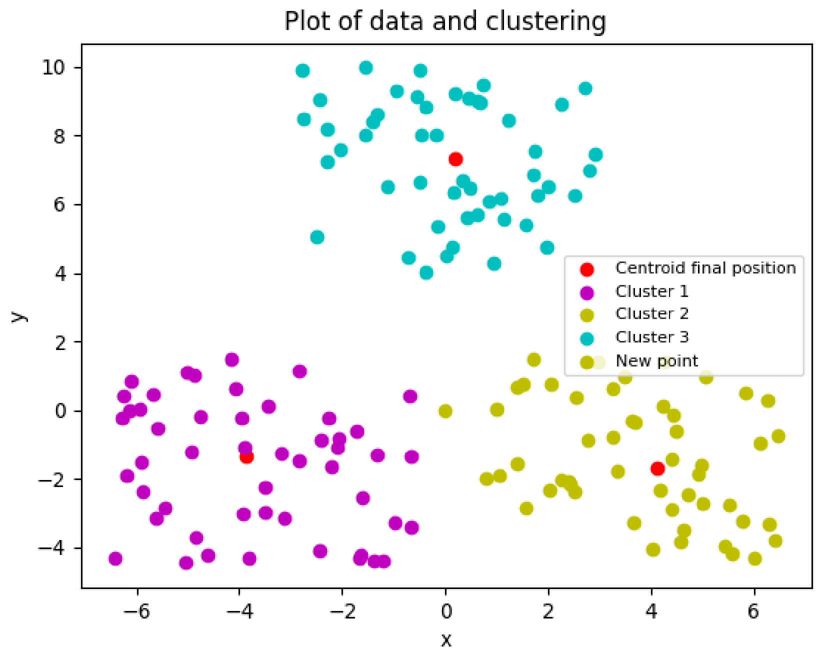


Figure 6: kNN classification, k = 11

2.3

This task consideres the use of a GA, genetic algorithm, in order to approximate a function g as $\hat{g}$. This is done by letting a population of individuals with random parameters, in this case in the range $[-2, 2]$, evolve.

The algorithm starts by initializing the population made up of indiviudals with random parameters as described above. Then each for each individual the approximation $\hat{g}$ is calculated and subsequently the error, ε, to g and finally the fitness of the individual according to the following.

$$\hat{g} = \frac{1 + a_1 x + a_2 x^2 + a_3 x^3}{1 + b_1 y + b_2 y^2 + b_3 y^3} \tag{34}$$

$$\varepsilon = \sqrt{\sum_{k=1}^{K} (\hat{g}(x_k, y_k)_k - g(x_k, y_k))^2} \tag{35}$$

$$\Rightarrow f = e^{-\varepsilon} \tag{36}$$

Then with the fitness for each individual the algorithm chooses two random individuals. The individual with higher fitness,f, has the probability $p_{tournament}$ to be chosen. Subsequently the individual with lower fitness has the probability $1 - p_{tournament}$. The probability of the individual with high fitness being chosen is generally high, which is fundemental to the population evolving generation by generation. The individual chosen goes through to the next generation. The selected individuals are still allowed to be selected after being selected once.

After this two individuals are selected as described earlier and they, with the probability p_{cross}, are crossed at a random crossing point. This means that all parameters up to the crossing point of individual 1 is transferred to individual 2 and vice versa.

Finally, each indivicual has a probability to mutate, $p_{mutation}$. If mutation is to occur, the individual has a probability p_{creep} of the mutation being a creep mutation. The creep mutation is defined as the following.

$$g := g + Unif(-C_r/2, C_r/2) \tag{37}$$

Where g denotes the gene and C_r the allowed range for the creep. Unif denotes a random value in the range within the parentheses.

After this process is done a new population is formed. By then iterating the method described each population will perform better and better generation by generation. The result of the implementation can be seen in the figure below.

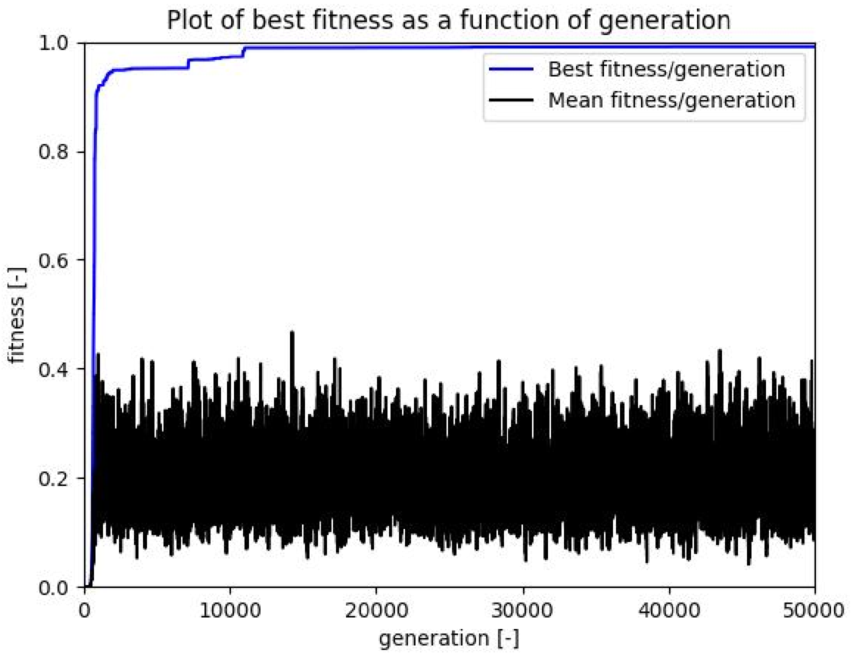


Figure 7: Best and mean fitness per generation

The maximum fitness reached over all generations was $f = 0,993$. This varies run to run due to the random elements of the algorithm. The best parameters found were the following.

$$a_1 = -1,99394473 \approx -2 \tag{38}$$
$$a_2 = 0,00572337 \approx 0 \tag{39}$$
$$a_3 = 0,99044607 \approx 1 \tag{40}$$
$$b_1 = 0,00586999 \approx 0 \tag{41}$$
$$b_2 = 0,49456163 \approx 0,5 \tag{42}$$
$$b_3 = -0,49969582 \approx -0,5 \tag{43}$$
$$\Rightarrow \hat{g}(x,y) = \frac{1 - 2x + x^3}{1 + 0,5y^2 - 0,5y^3} \tag{44}$$

By running the code that evaluates an individual with these rounded parameters the fitness reaches 1 and the error is estimated to be $\varepsilon \approx 4,44 \cdot 10^{-17}$.

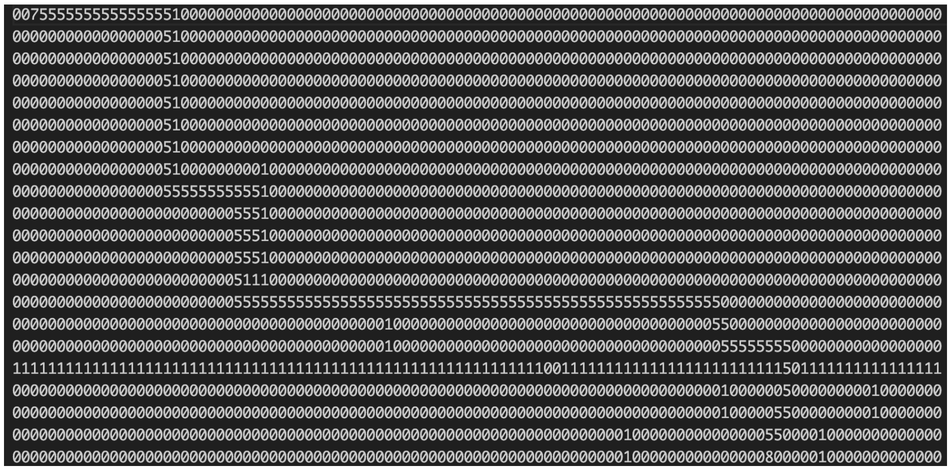
It was then asked that the parameters $p_{tournament}, p_{cross}, p_{mutation}, p_{creep}$, *creep rate*, *population size*, and *number of generations* in order to optimize the performance of the algorithm. It is intuative that more individuals and more generations will yield better results. Therefore these are chosen to be 200 individuals and 50000 generations as can be seen in the figure above. However, for the probabilities mentioned there is no intuative right answer, trial and error was therefore the method used. Therefore many versions of these parameters were used and the combination that gave the results above were the following.

$$p_{tournament} = 0,85 \tag{45}$$
$$p_{cross} = 0,7 \tag{46}$$
$$p_{mutation} = 0,11 \tag{47}$$
$$p_{creep} = 0,7 \tag{48}$$
$$creep\ rate = 0,02 \tag{49}$$

2.4

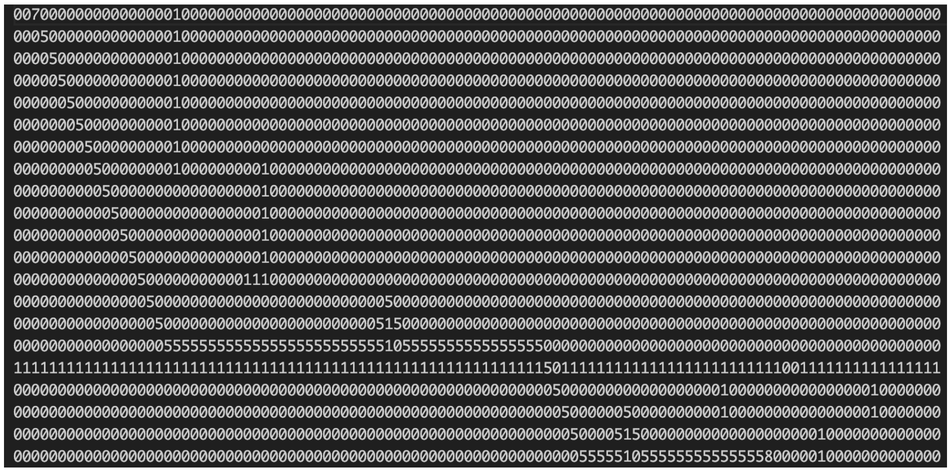
In the last task of this assignment a greedy best first search algorithm is to be implemented. This algorithm works by calculating the euclidian distance from every available neighboring node to the end node. An available neighbor is a neighbor that is within the matrix and has a value of 0 or 3, which is the end node. The node with the lowest distance to the end node is chosen as the new node and the process repeats. The task is to also implement 4- or 8-connectivity. This is done by allowing more neighbors in the case for 8-connectivity.

After completing the task the new altered matrix is saved to a text file. When a node is selected as having been visited 5 is added to its value to show the path taken by simply saving the matrix after the algorithm has run its course. Because of this in the solved matrix the start node has the value 7 and end node 8, symbolising that they have indeed been visited. The results can be seen below.



```
007555555555555555100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000005100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000005100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000005100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000005100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000005100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000005100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000005100000000010000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000005555555555510000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000055510000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000055510000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000055510000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000051110000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000055555555555555555555555555555555555555555555555555555555555555555555555000000000000000000000000
000000000000000000000000000000000000000000100000000000000000000000000000000000000000000000000000550000000000000000000000
000000000000000000000000000000000000000000100000000000000000000000000000000000000000000000000000055555555000000000000000
111111111111111111111111111111111111111111111111111111111111111111111111111001111111111111111111111111150111111111111111
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001000000500000000010000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001000005500000000010000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010000000000000005500001000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010000000000000008000001000000000000
```

Figure 8: 4-connectivity, large maze



```
007000000000000000100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000500000000000000100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000050000000000000100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000005000000000000100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000500000000000100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000050000000000100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000005000000000100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000500000000100000000010000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000050000000000000000010000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000005000000000000000010000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000500000000000000010000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000050000000000000010000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000005000000000001110000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000500000000000000000000000000500000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000050000000000000000000000005150000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000005555555555555555555555555105555555555555555555555555555555000000000000000000000000000000000000000000000
111111111111111111111111111111111111111111111111111111111111111111111111111501111111111111111111111111100111111111111111
000000000000000000000000000000000000000000000000000000000000000000000000000050000000000000000001000000000000000010000000
000000000000000000000000000000000000000000000000000000000000000000000000000005000000500000000001000000000000000010000000
000000000000000000000000000000000000000000000000000000000000000000000000000000500005150000000000000000000010000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000055555105555555555555580000010000000000000
```

Figure 9: 8-connectivity, large maze