ECM2423 - Coursework Exercise

Question 1 - 8-puzzle Game Heuristic Search

Question 1.1

In your own words, write a short description of how the 8-puzzle problem can be seen as a search problem.

The aim of the 8-puzzle is to reach a predefined configuration of tiles from another configuration of tiles. The 8-puzzle problem can be seen as a search problem due to its following attributes. First, the puzzle has discrete states, these different states are dependent on the position of each tile in the puzzle. One tile is missing enabling a tile next to it to move into that position. The puzzle has four operators, the missing tile can either move up, down, left, or down. Also, the puzzle has an end goal, this goal can any one of the possible states the puzzle can be in. In this question, the goal state is to have the missing tile in the top right with the rest of time tiles in ascending order. Lastly, a search problem requires a path cost. In this puzzle, the path cost is equal to one each time a tile is moved into the missing tile space.

Question 1.2

Question 1.2.1

In this question, you should first briefly outline the A Algorithm.*

The A* Algorithm is a search technique used to find a predefined node in a graph at the shortest possible cost. It uses weights on each node to decide which node to explore next, the node with the lowest weight is the node to explore next. This weight is usually referred to as ‘f’ and is the sum of two numbers.

The first number ‘g’ is the cost that it has taken (number of nodes traversed) to reach a node from the starting node. This prevents the algorithm going down a rabbit-hole that could potentially lead nowhere ensuring that the higher nodes are all traversed. For example, in the 8-puzzle problem there are two possible moves when starting the puzzle. The ‘g’ number ensures that both moves are explored instead of picking one move and trying to solve the rest of the puzzle from there.

The second number ‘h’ is a heuristic and attempts to guess how close the current node is to the final goal node. There can be many ways to calculate this heuristic for a given problem, a good heuristic can change the performance of an A* search significantly.

Question 1.2.2

Then, you should describe two admissible heuristic functions for the 8-puzzle problem and explain why they are admissible. Make sure you also explain why you chose these two heuristic functions in particular amongst all the possible ones.

The first heuristic function I have picked is the sum of ‘Manhattan’ distances each tile is away from its goal position. So the closer the tiles are to their goal position, the smaller the heuristic value gets. It’s called the ‘Manhattan’ distance because it’s the number of rows plus the number of columns away from the target rather than a direct distance as the crow flies. I chose this type of distance because it is a better representation of how tiles move in the puzzle and therefore should be a better heuristic.

The second heuristic function I have picked is the sum of ‘Euclidean’ distances each tile is away from its goal position. This is similar to the first heuristic but instead the distances are defined as a direct distance as the crow flies. I predict that this heuristic will not perform as well as the first once. I chose this heuristic because I wanted to see if I was correct and if so, how big the solving time is between these two heuristics.

These heuristics are admissible because once they reach their minimum weighting which is 0, the puzzle has been solved. In every puzzle move, a tile can only be moved such that it is closer to its goal by one position. Therefore, they will both always find the shortest path to the solution.

Question 1.2.3

Then, you should implement two versions of the A algorithm in Python to solve the 8-puzzle using the two heuristic functions you have chosen.*

`ECM2423_CA/question_1.2.py` (Start and goal states used from Figure 1)

Question 1.2.4

Briefly discuss and compare the results given by A when using the two different heuristic functions in question 1.2.*

The Manhattan Distance heuristic completed the puzzle in the shortest path on my computer around 7.75x faster than the Euclidean Distance. The Manhattan Distance took 12.09 seconds whereas the Euclidean Distance took 93.75 seconds. While I expected the Manhattan Distance to be faster, I did not think it would be by that much. However, it makes sense that the distance that better represents the limited movement of the tiles is a better heuristic. Tiles cannot move diagonally in one movement like the Euclidean distance would suggest. This demonstrates just how important the heuristic choice is for an A* algorithm in terms of time complexity.

Euclidean Distance

```
[redacted]@Wills-MacBook-Air ECM2423_Coursework % python3 question_1.2.py
Pick a heuristic function:
a. Euclidean Distance
b. Manhattan Distance
a
Initial State:
[
7 2 4
5 0 6
[8 3 1
[
Goal State:
[
0 1 2
[3 4 5
6 7 8

Solving Puzzle...

Puzzle has been solved!

End State:

0 1 2
3 4 5
6 7 8

Time Taken: 93.75 seconds
[redacted]@Wills-MacBook-Air ECM2423_Coursework %
```

Manhattan Distance

```
[redacted]@Wills-MacBook-Air ECM2423_Coursework % python3 question_1.2.py
Pick a heuristic function:
a. Euclidean Distance
b. Manhattan Distance
b
Initial State:

7 2 4
5 0 6
8 3 1

Goal State:

0 1 2
3 4 5
6 7 8

Solving Puzzle...

Puzzle has been solved!

End State:

0 1 2
3 4 5
6 7 8

Time Taken: 12.09 seconds
[redacted]@Wills-MacBook-Air ECM2423_Coursework %
```

Question 1.3

Write a general version of the A algorithm (using either of the two heuristic functions described above) to solve a generic version of the 8-puzzle where the user can input any start and goal state.*

`ECM2423_CA/question_1.3.py`

This code cannot solve any pair of start and goal puzzle configurations, many pairs are simply impossible to solve. With 362880 (9!) possible start configurations, only half of those are solvable for a given goal configuration.

Question 2 - K-Means Clustering

Question 2.1

Briefly outline the main concepts of the k-means clustering algorithm, and then describe how it could be applied to the task of hand-written digit recognition.

The k-means clustering algorithm is an example of an unsupervised machine learning algorithm to group similar data points in a set of data. It identifies k centroids and assigns each data to a centroid cluster while keeping the number of centroids as small as possible. A centroid represents the average or best example of a data point in a cluster. The user determines a target number of centroids before running the algorithm. This can be applied to the task of hand-written digit recognition by first defining 26 centroids using just the simple English alphabet. Each datapoint is then assigned to its closest centroid creating clusters. The mean of each cluster is then found and that becomes the new centroid. This is repeated until the algorithm stops moving the centroids.

Question 2.2

Question 2.2.1

Analyse the data using k-means clustering.

`ECM2423_CA/question_2.py`

Question 2.2.2

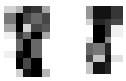
An extra 5 marks will be awarded if you implement your own k-means function.

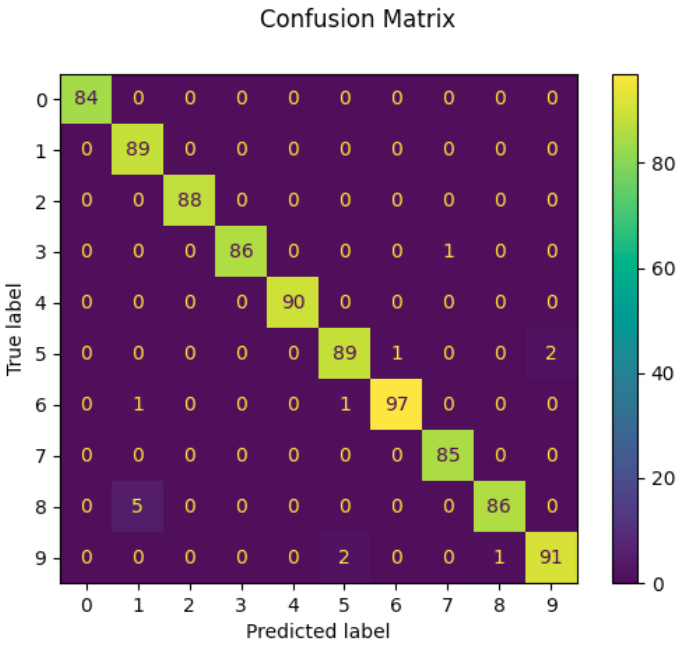
N/A

Question 2.2.3

Present and discuss the results of the k-means clustering on the dataset.

As the dataset also contains the true label for each image, a confusion matrix can be plotted comparing the true label against the predicted label. The matrix below shows ten clear clusters that correctly predict the label of the image. There are still some outliers where the prediction was wrong. The largest of which was the classifier predicting eights as ones, this occurred five times which while is still small, should still be noted. Due to the low resolution of the numbers, eights do seem to sometimes lose one or both of their holes that makes an eight look like an eight. An example of a couple of eights in the dataset are shown below.





Sklearn allows you to create a classification report from the K-means clustering training. The mistake of incorrectly predicted eights as ones is also demonstrated here. The prediction of ones having a precision of 94% compared to the weighted average of all the numbers of 98%. The precision column is the ratio between the number of true positives compared to the total number of both true positives and false positives. For example, no samples were incorrectly classified as a four. Similarly, the recall is the same ratio except the number of false negatives is used instead of false positives. This demonstrates the

K-Means Clustering Classification Report

	precision	recall	f1-score	support
0	1.00	1.00	1.00	84
1	0.94	1.00	0.97	89
2	1.00	1.00	1.00	88
3	1.00	0.99	0.99	87
4	1.00	1.00	1.00	90
5	0.97	0.97	0.97	92
6	0.99	0.98	0.98	99
7	0.99	1.00	0.99	85
8	0.99	0.95	0.97	91
9	0.98	0.97	0.97	94
accuracy			0.98	899
macro avg	0.98	0.98	0.98	899
weighted avg	0.98	0.98	0.98	899

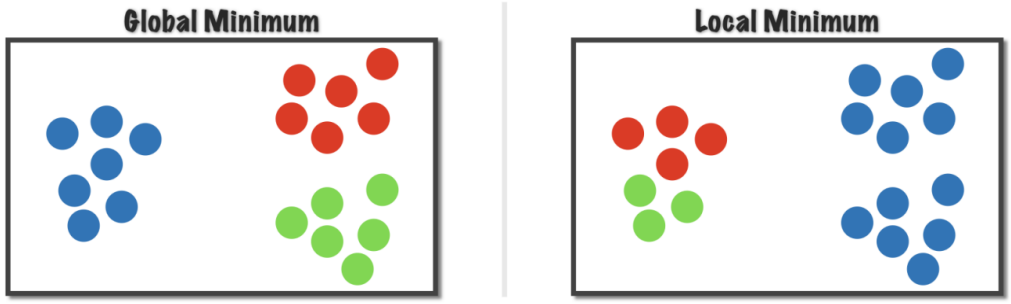
classifiers's ability to not miss any positive samples. For example, all zeros were correctly classified as the recall is 100%.

The f1-score is an average between the previous two columns, precision and recall. This shows that zero, two, and four were the best performing numbers for this classification. Lastly, the support column shows the total sample occurrences for each number which matches the confusion matrix above. There were only 899 samples because half of the data was used for training.

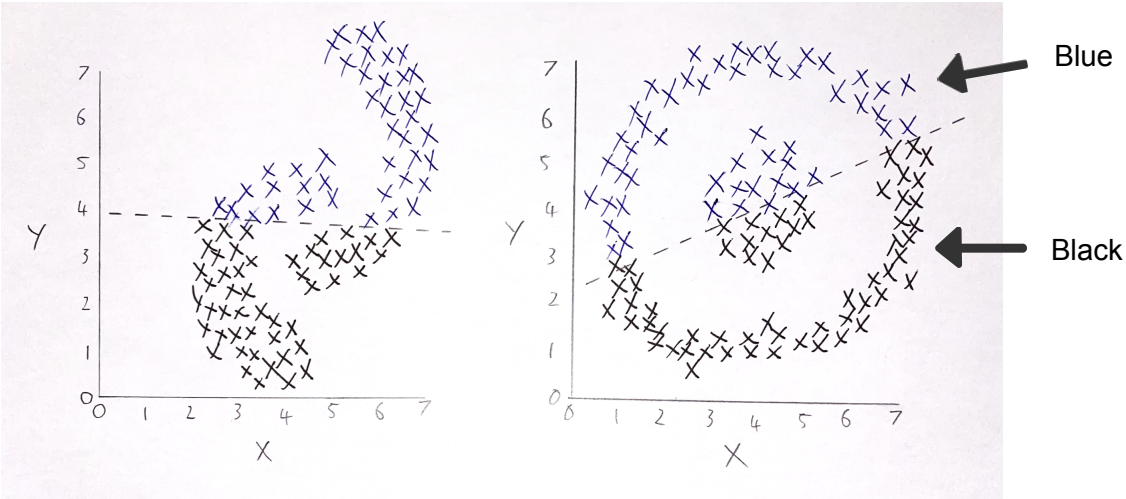
Question 2.3

Briefly outline what the assumptions and limitations of k-means clustering are.

A limitation of k-means clustering is that while it performs well on uniform data, it can struggle classifying data into categories that are of varied volume and density. k-means must be generalised to cluster such varied data. In addition, k-means can be affected by outliers in the data. Either these outliers can incorrectly be put in their own category or centroids of clusters can be swayed by distanced outliers. To reduce this limitation, outliers could be identified and removed. k-means can often get trapped in a local minima as seen below. The global minima which is the target is often hard to get to due to the problem complexity.



An example of where k-mean clustering would not work is where patterns overlap each other or are within each other in a distribution of data while still having a clear pattern from a human's perspective. This is because k-means can only form spherical clusters. An example of this is shown below. Each graph has two categories of data but k-means as shown by the dashed line has not divided them correctly.

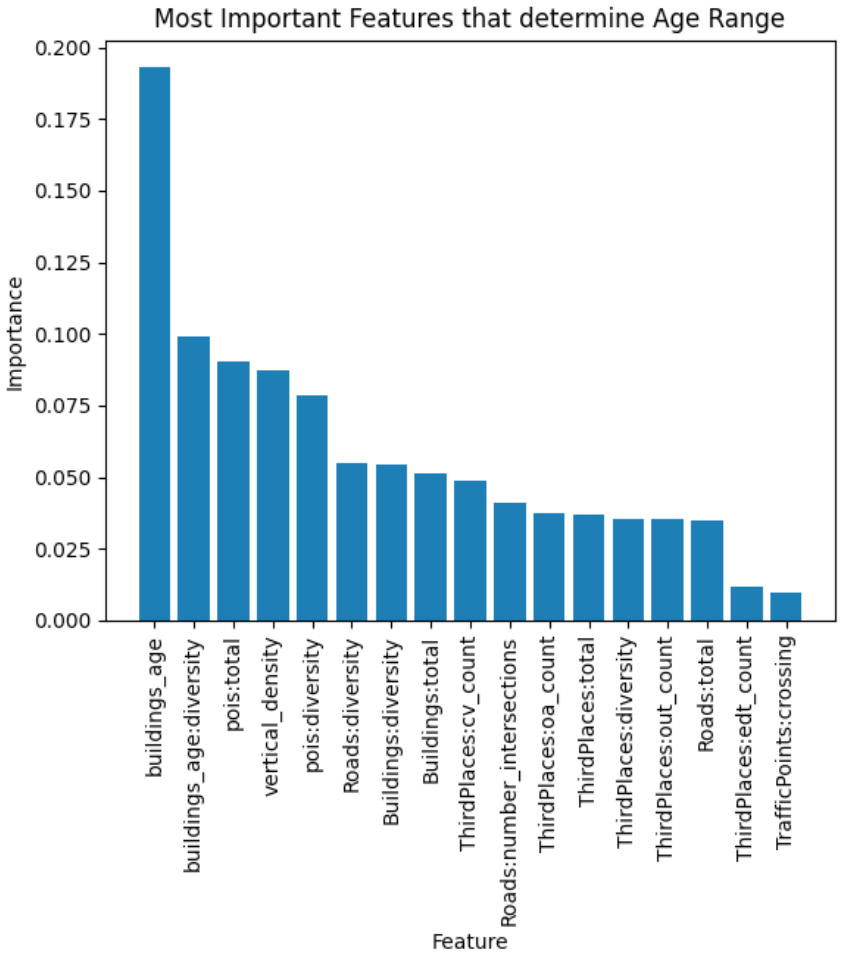


Question 3 - Classification of urban areas using decision trees

Question 3.1

Which features are most important for the analysis? Is there any feature that could be omitted? If so, which ones and why?

`Code:`
`ECM2423_CA/question_3.py`



The bar chart on the right shows that the average age of buildings in the cell is the most important feature for predicting the most present age in a cell. It makes sense that in very general terms, older people tend to live in older buildings. The next most important features which are around half as important are: an index measuring the diversity of types of building; the density of points of interest; the vertical density of buildings; and an index measuring the diversity of points of interest.

The cell ID can be omitted from the decision tree as this is an arbitrary value with each cell having its own unique value, it will have no effect on the most likely age present in a cell. Including this column could cause random patterns to be found and reduce the accuracy of the decision tree.

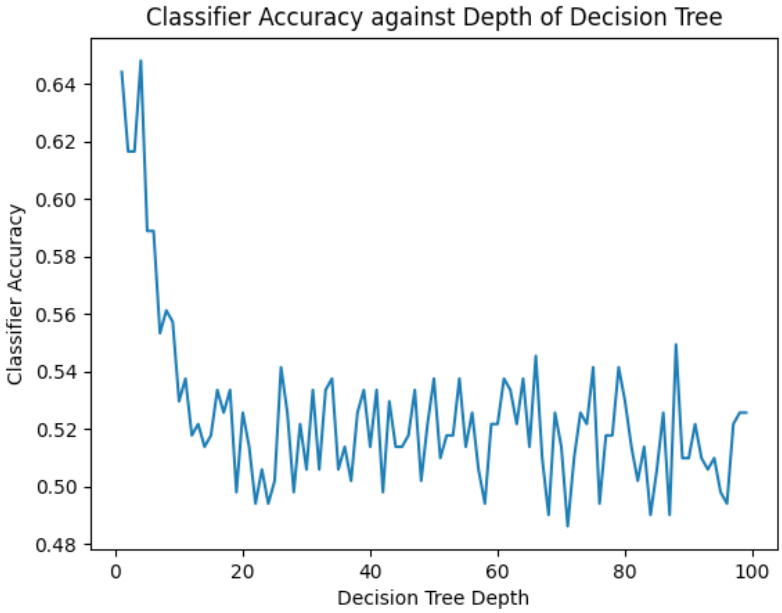
Decision Tree with no max depth defined
Original image can be found at: `ECM2423_CA/Decision Tree.png`

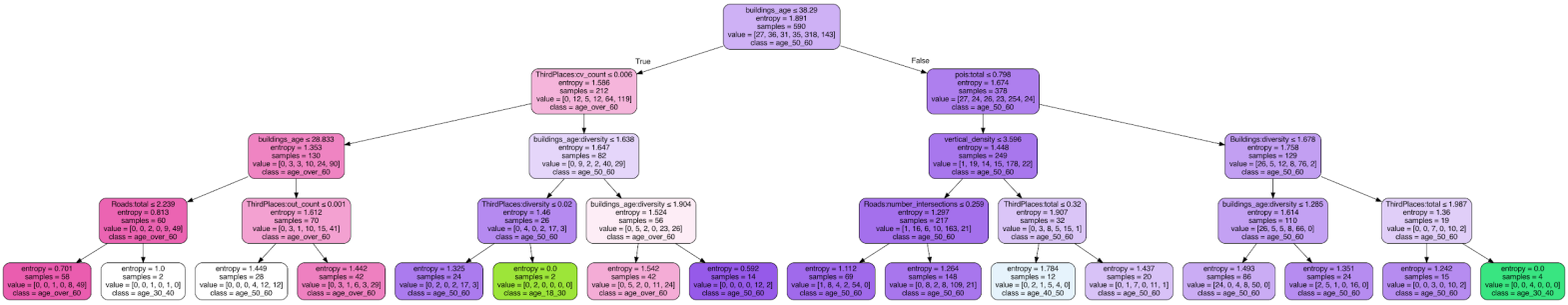
Question 3.2

What is the accuracy of your classifier? What happens to the accuracy if you vary the depth of the decision tree? Briefly discuss your results.

`Code: ECM2423_CA/question_3.py`

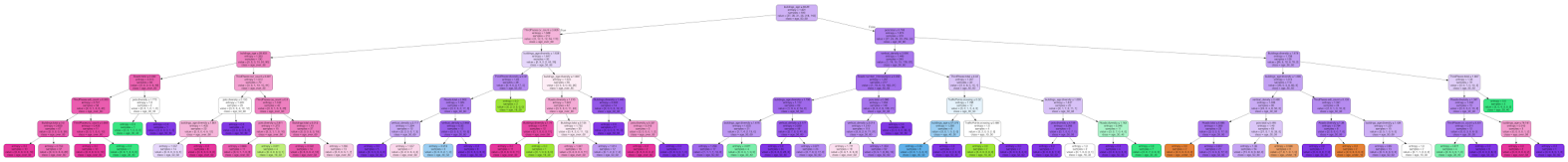
The highest accuracy of the classifier was found to be 65%. The graph to the right shows the classifier accuracy as the depth of the decision tree increases. There's a clear trend that as you increase the depth, the accuracy decreases. This reduction levels off at a depth of 20 to around an accuracy of 52% from there on. The most accurate decision tree depth was four, a decision tree with this depth is shown below. The problem with this particular tree is that it does not contain a path that resolves to an age category of 'age_under_18'. I found the best decision tree to be a depth of 6 as it still has a relatively high accuracy and resolves all possible age categories at the bottom of the tree. This decision tree can also be found below.





Decision Tree with a max depth defined as 4

Original image can be found at: `ECM2423_CA/Decision Tree - Depth 4.png`



Decision Tree with a max depth defined as 6

Original image can be found at: `ECM2423_CA-/Decision Tree - Depth 6.png`