Practical Machine Learning

Assignment 1 – k-Nearest Neighbours Report

Michael McAleer (R00143621)

Part 1 – Hyper-Parameters and the *k*-Nearest Neighbour Algorithm

When working with the *k*-nearest neighbour algorithm (*k*-NN) a number of approaches can be taken to improve the accuracy at which the algorithm correctly predicts the value of a test query against a training data set – these is known as hyper-parameter optimisation.

In this test both *k*-NN for classification and *k*-NN for regression are in-use, so a range of parameters are analysed for impact on accuracy across both variants of *k*-NN. In both cases, the output will be as a result of the *k* closes training examples in the feature set so there may be some common patterns between both classification and regression problems. The neighbours are taken from a set of instances for which the class (*k*-NN classification) or value (*k*-NN regression) is known, this is the training set from which the lazy-learning algorithm makes its predictions, so no additional training step is required.

The Value of K

The most obvious parameter for tuning the performance of the *k*-NN algorithm is the *k* value itself. The best value designation for *k* depends on the data, smaller values like *k*=1 can be prone to noise in the data set, known as over-fitting, and larger values can negate this effect of noise on the classification, but selecting a value that is too large can result in under-fitting where the boundaries between classes become less distinct.

There are various methods in use for selecting a *k* value, such as the square root of the number of instances in the training set or n-fold cross validation, but for the purposes of this report the whole range of *k* values available will be under scrutiny, from *k*=1 to max length of training set. The intention of this large scale of values is to see if any discernible patterns arise from using all *k* values available with other hyper-parameters.

Voting Selection

Changing the voting selection method will impact how the algorithm determines the output classification or value of the test instance. Two methods of voting will be investigated during testing of the *k*-NN algorithm; majority voting and inverse distance-weighted voting.

In majority voting all votes are equal. For classification problems, for each class the amount of *k*-neighbours with each class are counted and the class with the most votes is returned. For regression problems, the values of each neighbour are treated as equal, and the average of *k*-neighbours is returned.

In inverse distance-weighted voting, closer neighbours get higher votes in determining the output class or value. When using weighted votes it is also possible to use all neighbours in a training set as those very far from the test query will have very little impact on the prediction. With weighted voting, the weights and values of each neighbour are determined, all votes are summed and the prediction will be the class with the highest vote.

Measuring Distance

As *k*-NN is based on the distance between the test data and each of the instances in the training data, the distance calculation used will impact how the algorithm determines what is regarded as *close* to the test instances. The distances under investigation in this test are ‘Euclidean’, ‘Manhattan’, ‘Minkowski’ and ‘Heterogeneous’.

The '*Euclidean distance*' measures the straight line distance between two points **p** and **q** in Euclidean space (two more dimensions). The distance between them is determined using the Pythagorean formula:

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

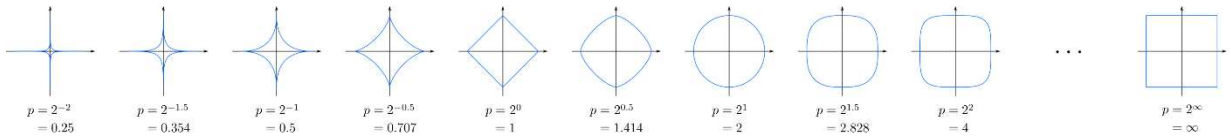
The '*Manhattan distance*' measures distance parallel to each axis, not diagonally (straight line), so to calculate the distance the sum of the absolute values of the differences of the co-ordinates is taken:

$$d_1(\mathbf{p},\mathbf{q}) = \|\mathbf{p} - \mathbf{q}\|_1 = \sum_{i=1}^{n} |p_i - q_i|,$$

The '*Minkowski distance*' measures the distance between two feature vectors **p** and **q**.

$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$$

More specifically the value of p in the calculation can be altered to change the power mean of the differences between **p** and **q** to produce various unit circles:



The Minkowski distance can be viewed as a generalisation of both the Euclidean and Manhattan distance metrics, as *p*=1 results in the Manhattan distance and *p*=2 results in the Euclidean distance.

The '*Heterogeneous distance*' metric is required for this test of k-NN algorithm performance as the classified data is a mixture of both continuous and discrete data. For discrete (categorical) data distance calculations the '*Hamming distance*' metric will be used alongside other continuous distance metrics such as Euclidean or Manhattan. The Hamming distance calculations assigns a value of 1 where features between **p** and **q** are different and 0 if they are the same:

$$\mathrm{d}(p_i, q_i) = \begin{matrix} 0\ if\ qi == pi \\ 1\ if\ qi\ != pi \end{matrix}$$

Normalising the Data

The last area of investigation into the impact of optimising the *k*-NN algorithm is the data itself. This is not an issue for the regression problem as all the instance dimensions are similar, but for the classification data four of the five dimensions are discrete and one is continuous. What this means is that four of the

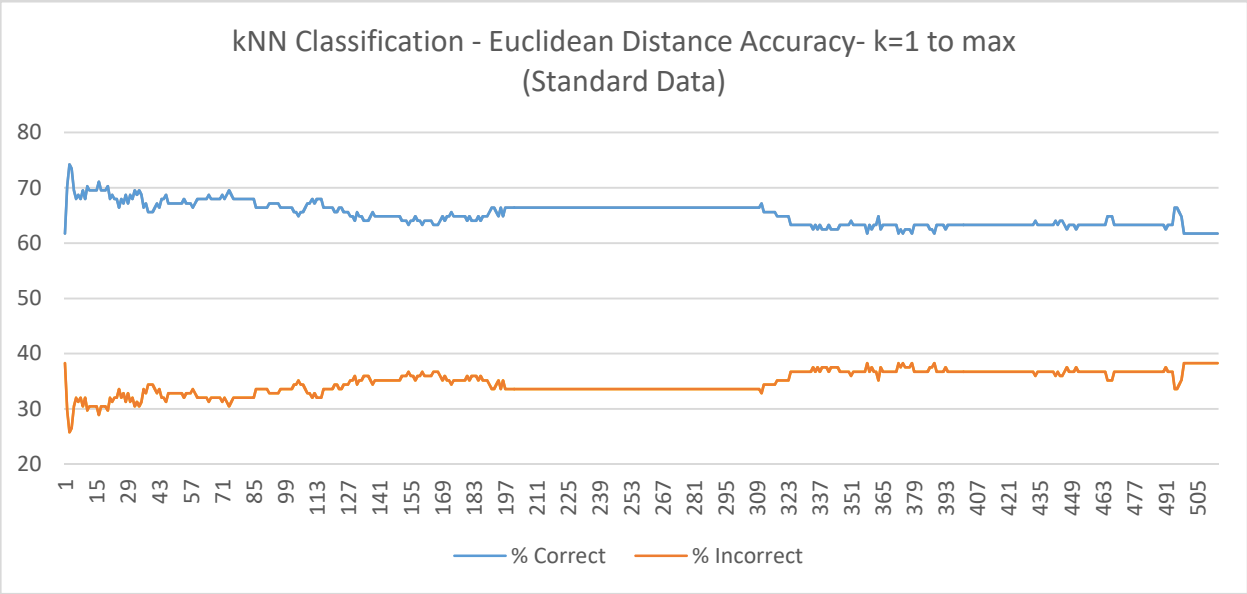
dimensions are in the range of 0-5 whole integer numbers, where the last for age is continuous. This addition of a continuous data dimension in the distance calculation may possibly dominate the end result too much and thus negatively impact the determination of the nearest neighbours. To alleviate this issue the data can be normalised independently by dimension (feature), so that the dimension minimum and maximum values are used to determine a new value for that instance dimension:

```
new_value = (old_value – min_value) / (max_value – min_value)
```

Part 2 – Hyper-Parameter Optimisation with *k*-NN for Classification

(Note: In parts two and three, where zero nearest neighbours is shown in graphs this represents all neighbours in the vote, the range of k values was investigated from zero to the length of the training data set.)

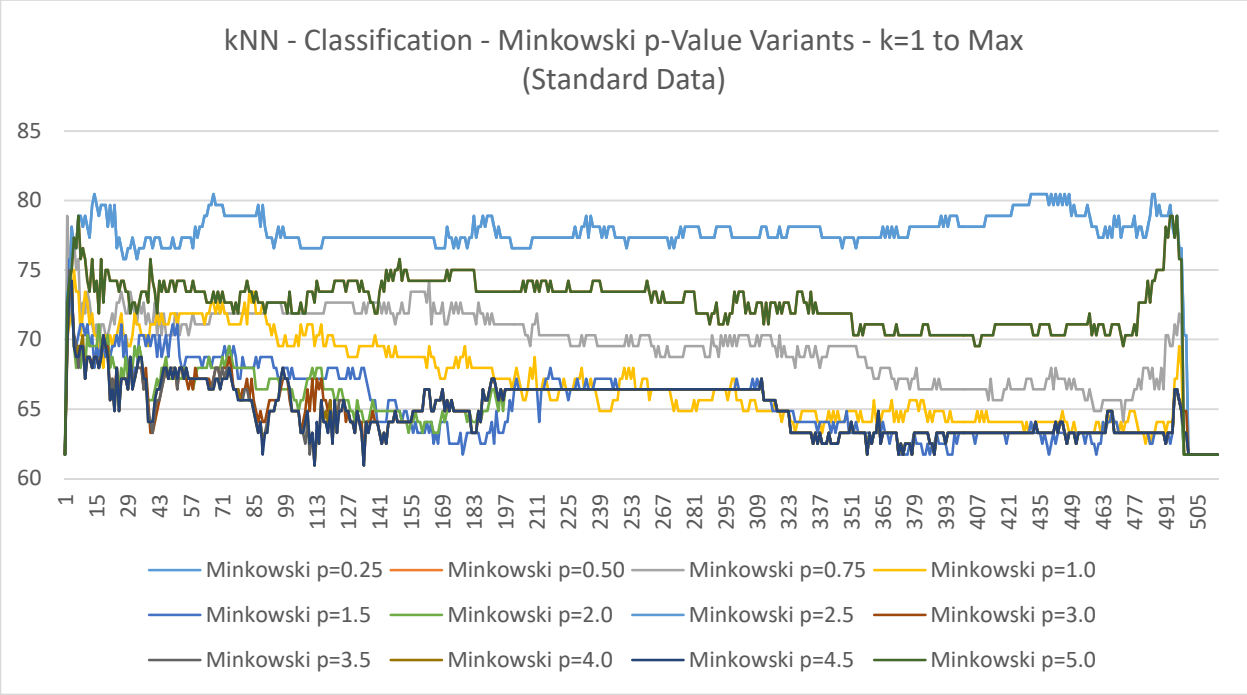
The first area of investigation in the *k*-NN classification was the impact of the *k* value on the overall accuracy of the *k*-NN algorithm. The initial test run used standard (not normalised) data, with majority voting and Euclidean distance, comparisons were then made by changing these constituent hyper-parameters until the most desirable results were found.



The results from the first test were overall good where the *k* value was low with the highest % correct value reached where *k = 2*. The overall % correct value got progressively lower as the *k* value increased, but the value never went below 61.71875% (*k=514 (all)*).

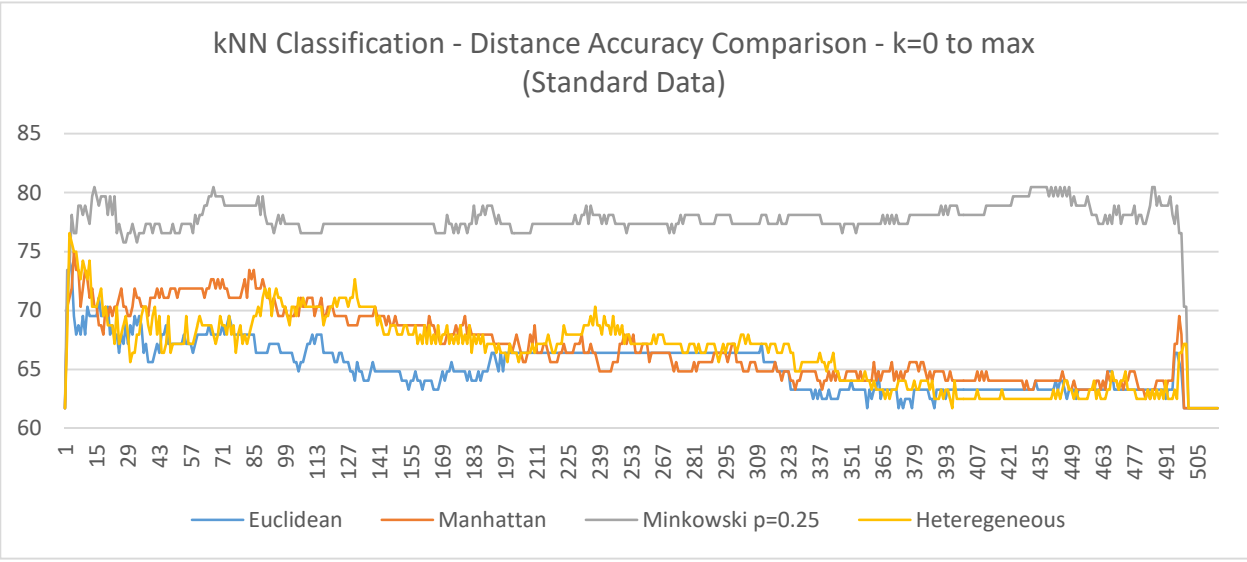
To further examine the impact of the *k* value on the overall accuracy of the algorithm, the same test was run but with all distance calculations. Firstly, the Minkowski distance *p* value variants were run so the most effective of those could be compared with the other types of distance calculation. Only the accuracy of correct predictions is measured as the incorrect prediction is directly proportional to the correct prediction so can be determined without the need of results in the graph. Additionally, a range of *p* values

were used in testing to get a well-rounded insight into the effects of various values on the accuracy of the algorithm. These values for '*p*' are 0.25, 0.5, 0.75, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, and 5.0.



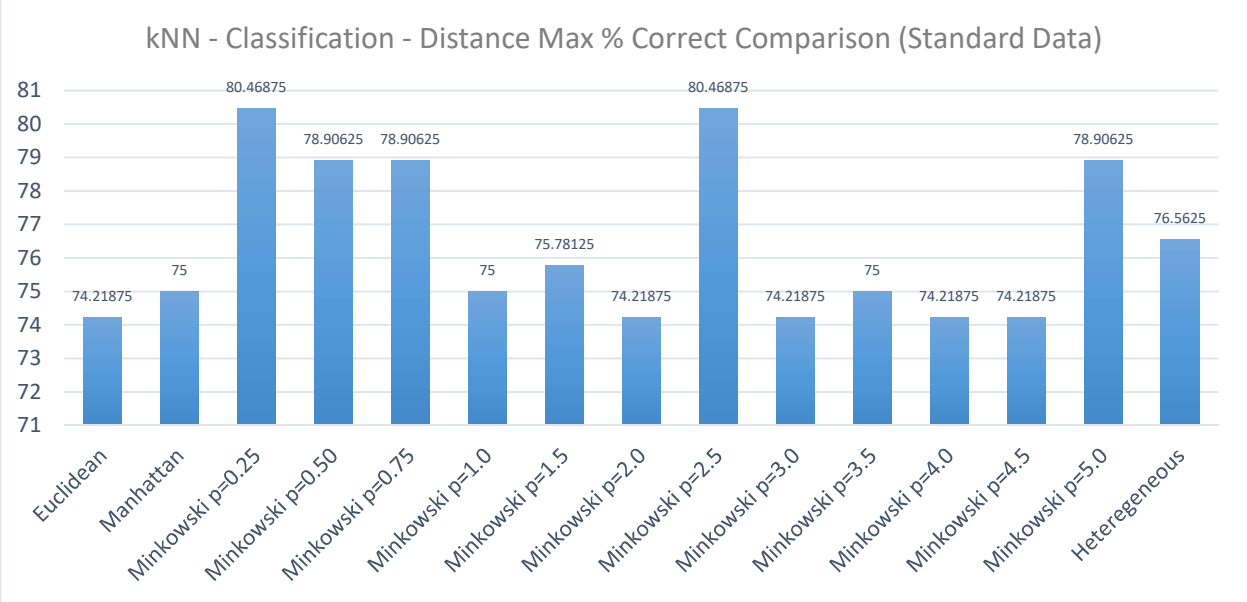
Looking at the Minkowski distance p-value variant results a clear improvement can be seen instantly when looking at the effect of the *k*-value over the various *p*-values. Interestingly, the same results are found for *k* values where *p* is equal to *0.25* and *2.5*. Very impressive results are seen for both these *p*-values, the highest accuracy observed was with *k*=13 and 80.46875%, and an overall average percent of 77.36047%. Similar to the Euclidean distance however, the worst accuracy is seen where all neighbours are used with an accuracy of 61.71875%.

Using the best Minkowski *p*-value variant for comparison the rest of the distance metrics were analysed against each other.



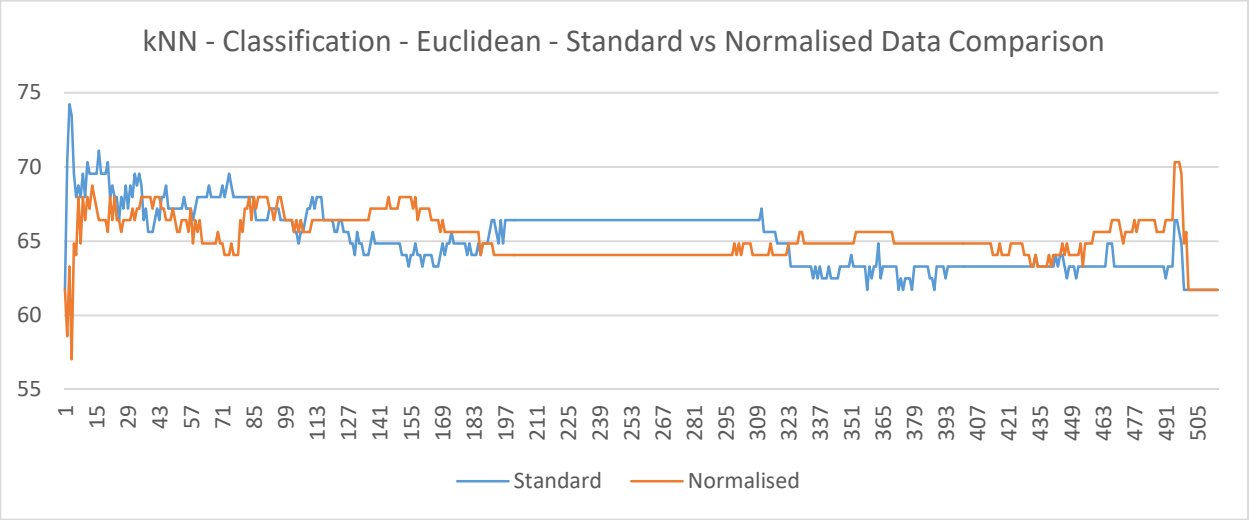
A similar pattern can be seen between the Euclidean, Manhattan and Heterogeneous distances whereby the best results are seen in the lower *k* values and the accuracy keeps decreasing until all neighbours are used for the majority vote. The Minkowski distance is the only exception to this pattern, the accuracy of the algorithm is only impacted by the *k* value when *k* is includes almost all neighbours *k*=max-20. From the various tests it can be said that when working with *k*-value ranges across the distance variations, the best accuracy is seen when *k* is a low value (<10), the exception to this being Minkowski distance where the best *k* value was 13 where *p*=0.25 and the overall accuracy remained high until almost all neighbours were included in the majority vote.

The overall best accuracy seen across all distances was in the region of 74.21875% to 80.46875%, indicating a similar prediction rate when the appropriate *k* value could be determined.



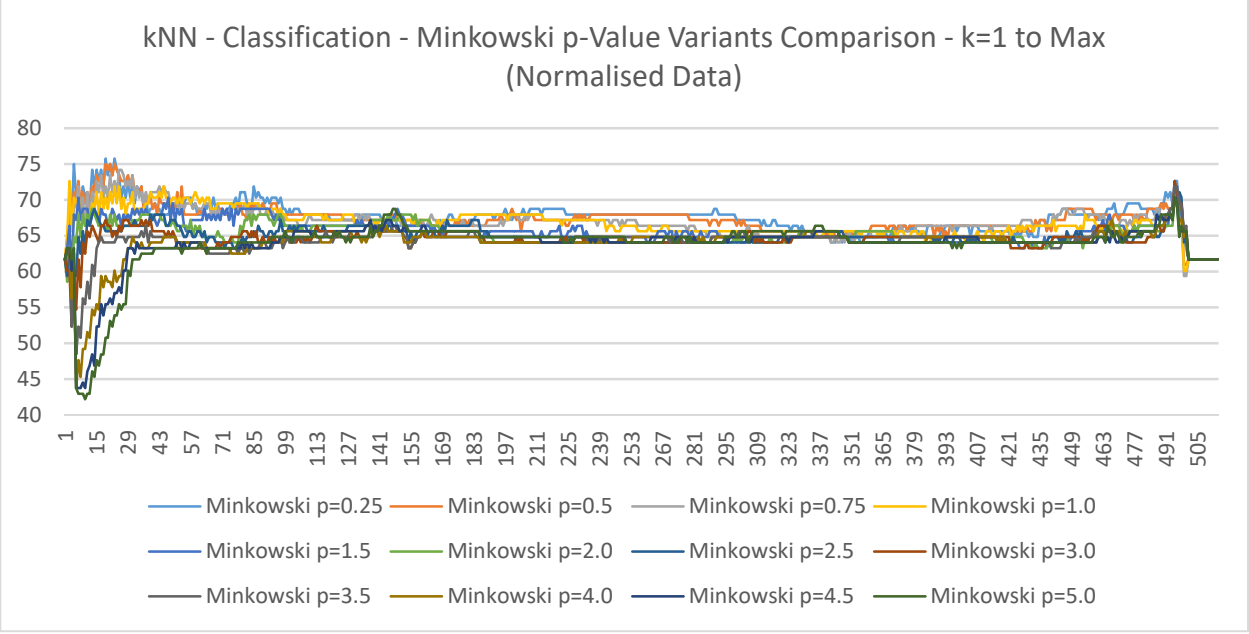
Distance	***k* Value**	**% Accuracy**
Euclidean	2	74.21875
Manhattan	6	75
Heterogeneous	2	76.5625
Minkowski (p=0.25)	13	80.46875
Minkowski (p=0.5)	6	78.90625
Minkowski (p=0.75)	1	78.90625
Minkowski (p=1.0)	4	75
Minkowski (p=1.5)	2	75.78125
Minkowski (p=2.0)	2	74.21875
Minkowski (p=2.5)	13	80.46875
Minkowski (p=3.0)	2	74.21875
Minkowski (p=3.5)	2	75
Minkowski (p=4.0)	2	74.21875
Minkowski (p=4.5)	2	74.21875
Minkowski (p=5.0)	2	78.90625

Using the same voting technique, distance calculations, and *k*-value ranges, the impact of normalising the data was tested. Starting as Euclidean as the base-line, the results of standard vs normalised was analysed.



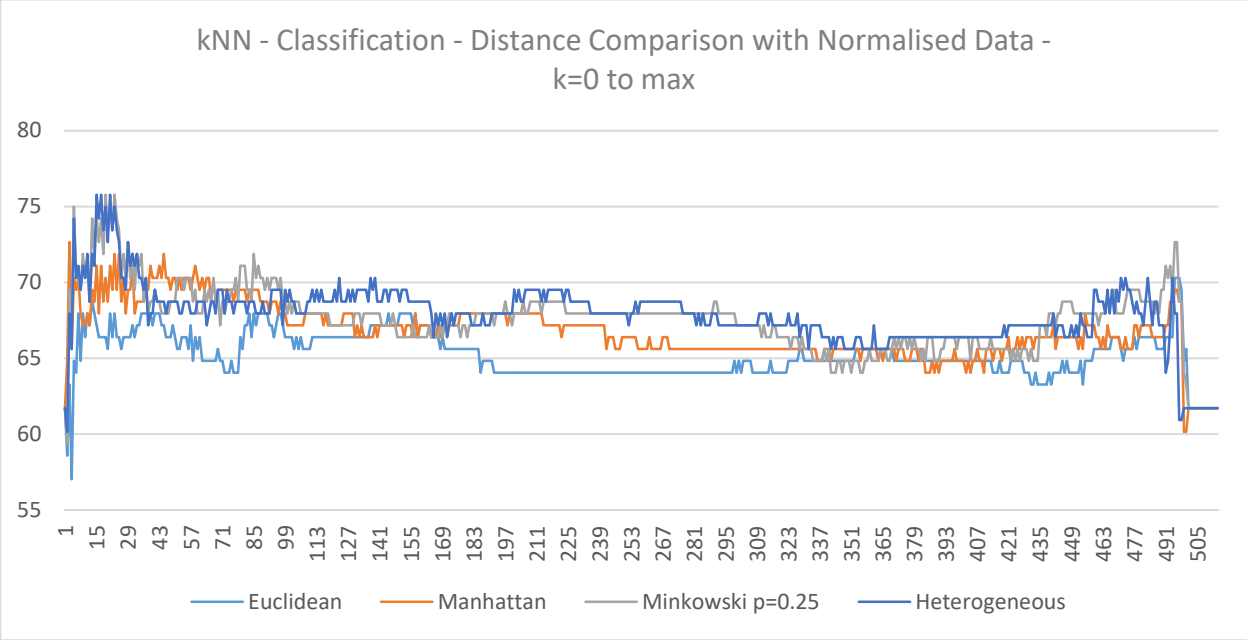
Using normalised data in the *k*-NN algorithm resulted in lower accuracy when the k-value was lower, and the best accuracy when the k-value was almost maximum. Overall the results are not hugely different, standard data had an average accuracy of 65.219% and a max value of 74.21875% and the normalised data had an average accuracy of 65.205% and a max value of 70.3125%. In terms of which is better, the standard data can be seen as the more desirable as it performs best at lower *k*-values where the algorithm is less computationally demanding.

Taking these results and performing the same analysis as with standard data, the effect of normalised data on Minkowski distance variants was investigated. The initial thought was that similar results would be seen, with specific *p*-value variants being the outright best choice for overall accuracy.



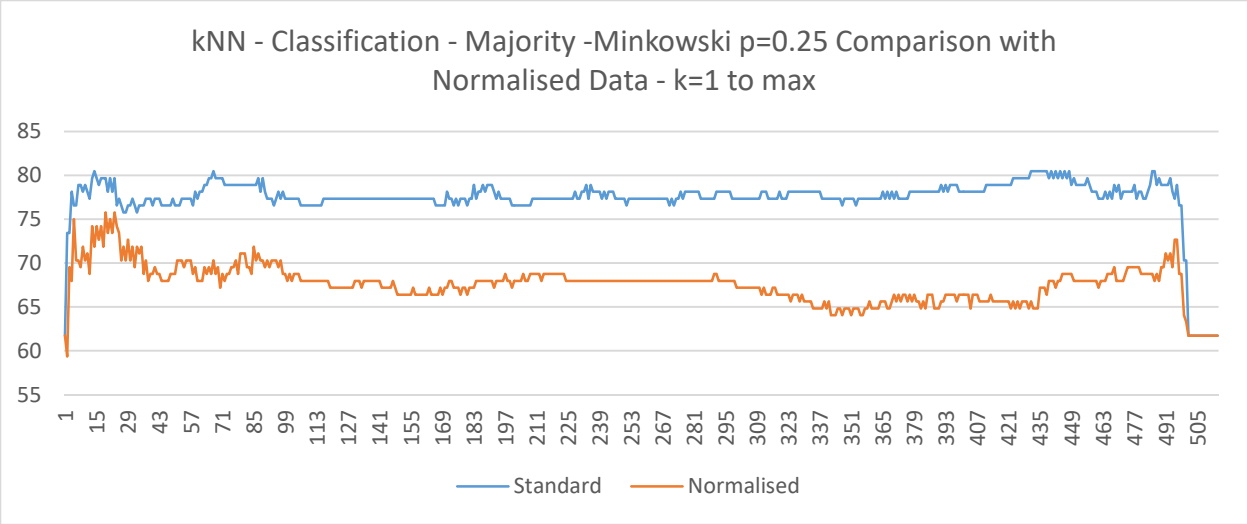
The results from the *p*-value variants using normalised data were not what was expected, a trend was seen where the higher the p-value was the lower the resulting accuracy for low k-values, right down to the lowest accuracy seen across all tests with *p*=4.0/4.5, *k*=5 and an accuracy of 43.75%. The best results were seen for lower p-values and associated k-values, *p*=0.25 and *k*=13 had the best accuracy with 75.78125%. A pattern was observed in the k-values across all p-value variants with the accuracy remaining around the approx. 65% until the k-value almost reached its max value *k*=max-20 and the accuracy increased across all p-values before dropping back down again *k*=max-10.

Taking the best performing Minkowski distance *p*-value of 0.25, normalised data was compared across the remaining distance metrics. Similar patterns were seen across all distance metrics, the best accuracy was seen when the *k*-value was low, and the accuracy continued to decrease uniformly until there was an increase in accuracy nearing the *k* max value. Both Heterogeneous and Minkowski with *p*=0.25 exhibited the best accuracy with 75.78125% where both used low *k*-values.



Thus far the Minkowski calculation metric has been the best when using majority vote and low *k*-values. Comparing both these using standard and normalised data should give a better picture of the effect of data normalisation on well-performing algorithm hyper-parameters.

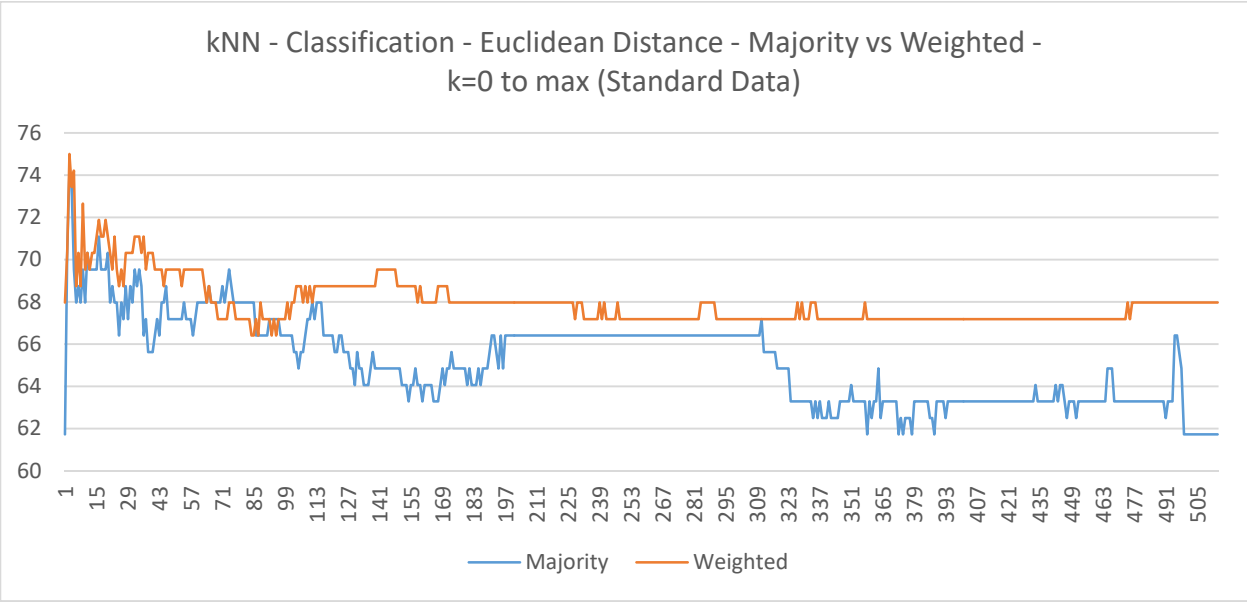
Looking at both results in the table below it can be easily seen that in this instance using the standard data, that is not normalised, provided the best overall accuracy and the best overall average accuracy across all *k*-values. Both standard and normalised variants had the exact same lowest accuracy when *k*=max with 59.375%.



Overall, with majority voting in use in the *k*-NN algorithm, in this case it was found that the standard data set provided the better results than normalised data, and the best accuracy was found using lower *k*-values. The only exception to this finding is using Minkowski difference metric with normalised data and high *p*-values, where the accuracy was at its lowest when the *k*-value was low.

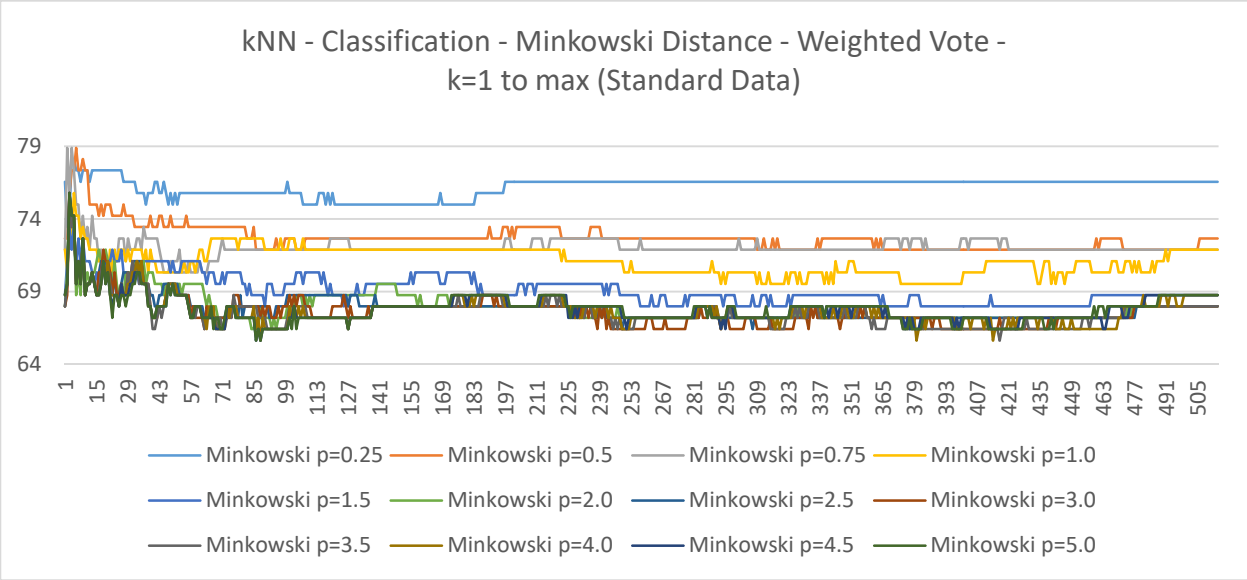
Given more time for analysis and testing, further investigation could be carried out into the effect of changing the distance calculation in the heterogeneous distance for continuous data. In these test Euclidean distance was used for calculating the distance between continuous data dimensions, this could be switched with the best performing distance to determine if the accuracy could be further improved than what was seen individually. So instead of using Euclidean distance with the Hamming distance, the Minkowski distance with *p*=0.25 could be used with the Hamming distance.

With all variations of hyper-parameters tested using majority vote, the next stage in testing the *k*-NN classification algorithm was to use inverse distance-weighted voting to determine the output class. Similar to the majority vote tests, the first test was with standard data, Euclidean distance, and across all *k*-values.

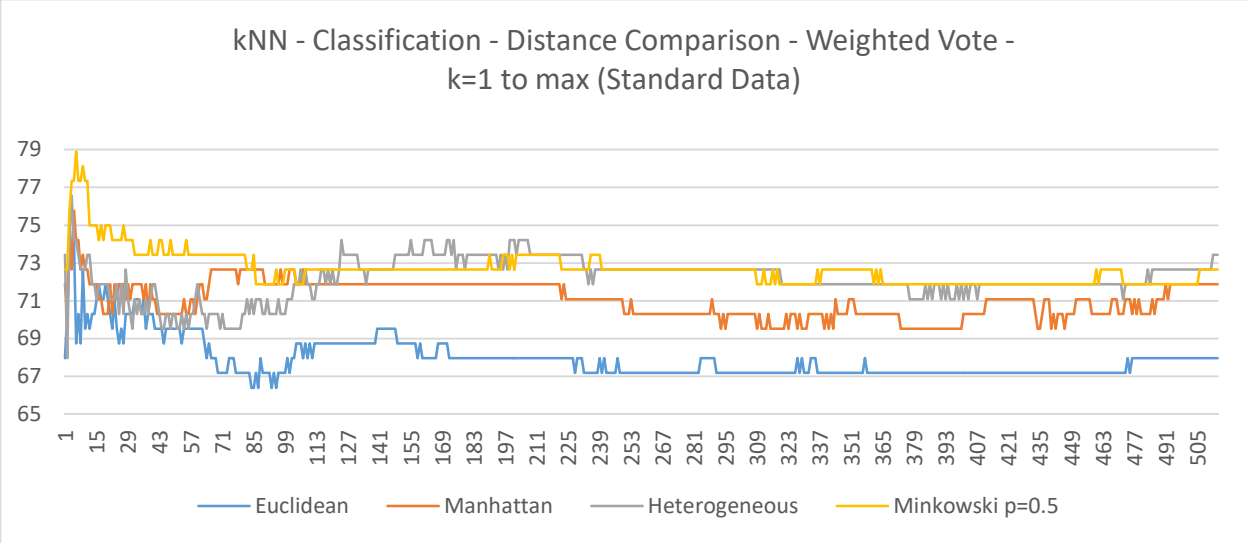


Inverse distance-weighted voting resulted in both higher accuracy in lower *k*-value and more consistent accuracy across all *k*-values, this is likely due to far off neighbours having next to no impact on the final vote. This behaviour can also be attributed to the lack of a drop in accuracy when the *k*-value is set to its max value, taking in to consideration all instances in the training data set.

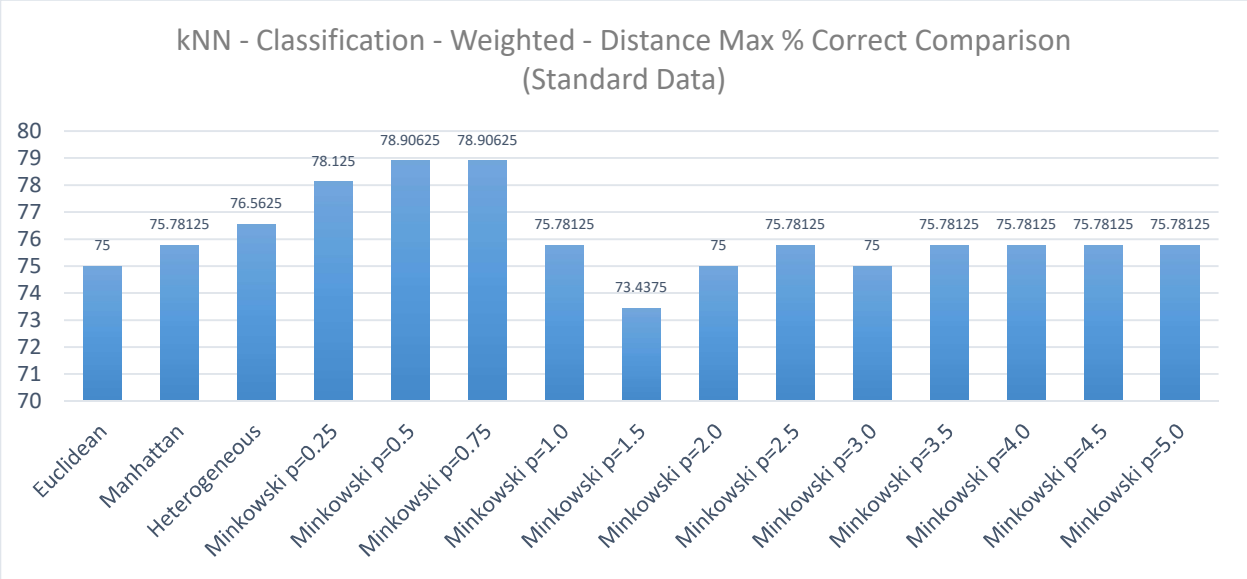
Looking at the impact on the accuracy of the *k*-NN algorithm of the weighted vote, *p*=0.25 does not provide the best accuracy, this is now achieved by *p*=0.5 *k*=5 with an accuracy of 78.90625% and *p*=0.75 *k*=1 with the same accuracy of 78.90625%. The Minkowski p-value of 0.25 does have the most consistently high accuracy which does not fall below 75%, but in this case the highest accuracy is the most important when it is achieved with such low *k*-values.



Using the Minkowski *p*-value of 0.5 as our best solution for weighted voting, the rest of the distance metrics were compared.



The Minkowski distance metric is again the best performing with the highest accuracy achieved, Heterogeneous rate comes in close as the next best distance metric when weighted voting is used.

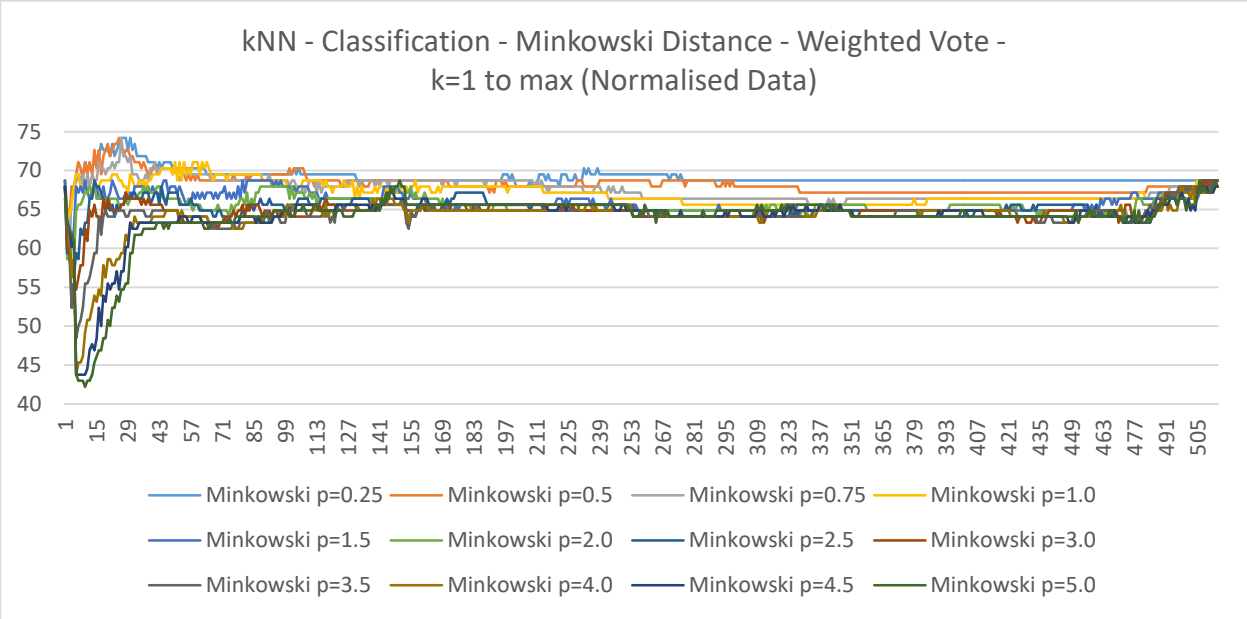


When looking at the associated *k*-values of the best accuracy achieved by each distance metric, consistently the lowest *k*-values provide the best results. Similar to majority voting, the best Minkowski results are achieved with lower p-values.

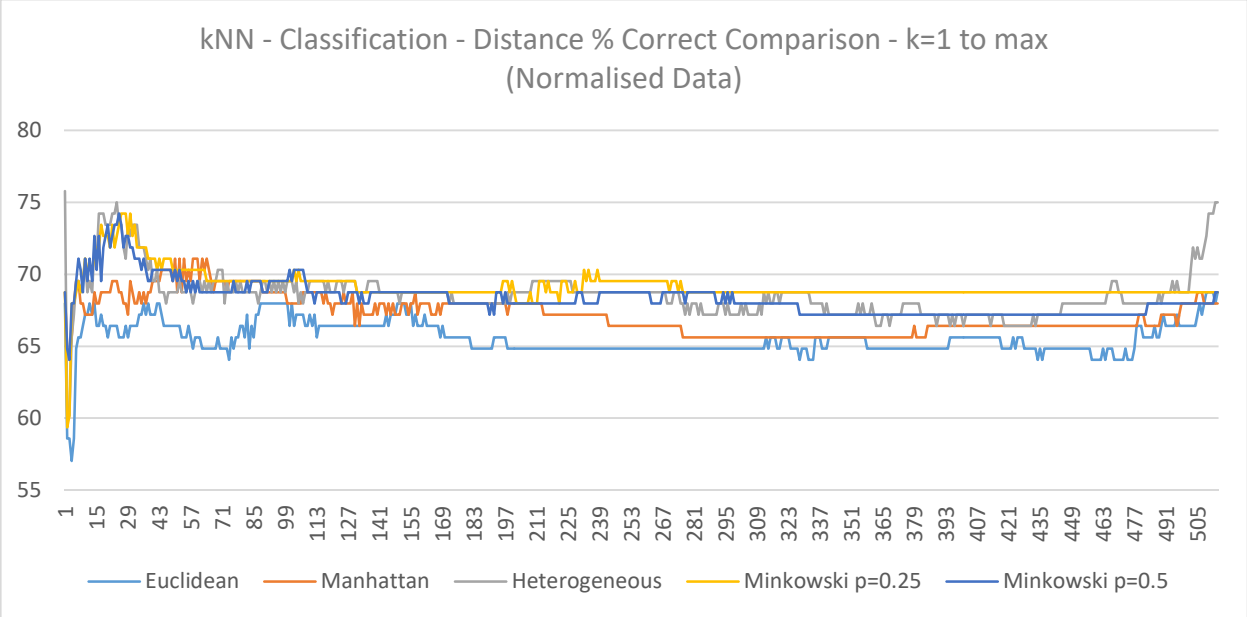
Distance	***k* Value**	**% Accuracy**
Euclidean	2	75
Manhattan	4	75.78125
Heterogeneous	3	76.5625
Minkowski (p=0.25)	3	78.125
Minkowski (p=0.5)	5	78.90625
Minkowski (p=0.75)	1	78.90625
Minkowski (p=1.0)	4	75.78125
Minkowski (p=1.5)	2	73.4375
Minkowski (p=2.0)	2	75
Minkowski (p=2.5)	2	75.78125
Minkowski (p=3.0)	2	75
Minkowski (p=3.5)	2	75.78125
Minkowski (p=4.0)	2	75.78125
Minkowski (p=4.5)	2	75.78125
Minkowski (p=5.0)	2	75.78125

Normalising the data set before running the *k*-NN algorithm had the same impact on the weighted vote as it did on the majority vote, the accuracy fell across all distances and *k*-values, although not by much. Using the Minkowski distance first, the same pattern could be observed whereby the lower *k*-values were prone to lower accuracies, but these improved and stabilised across all *k*-values and *p*-values. The lower

p-values were again the best performing with *p*=0.25, *p*=0.5 and *p*=0.75 providing the best accuracy with 74.21875%. The associated k-values for these were *k*=24 and *k*=25 so the best results were not found with *k*-values as low as were found with majority voting tests using the same distances and *k*-values.



The impact on the remaining distance metrics of normalising the data could be seen to fit the same behaviour pattern, lower *k*-values resulted in the poorest accuracy, with the best accuracy for each distance occurring in the *k*=20 to 30 range. The exception to this was the Heterogeneous distance metric which performed best when the *k*-value approached maximum, the highest accuracy seen for weighted voting using normalised data was 75.78125% when *k*=514 (all).

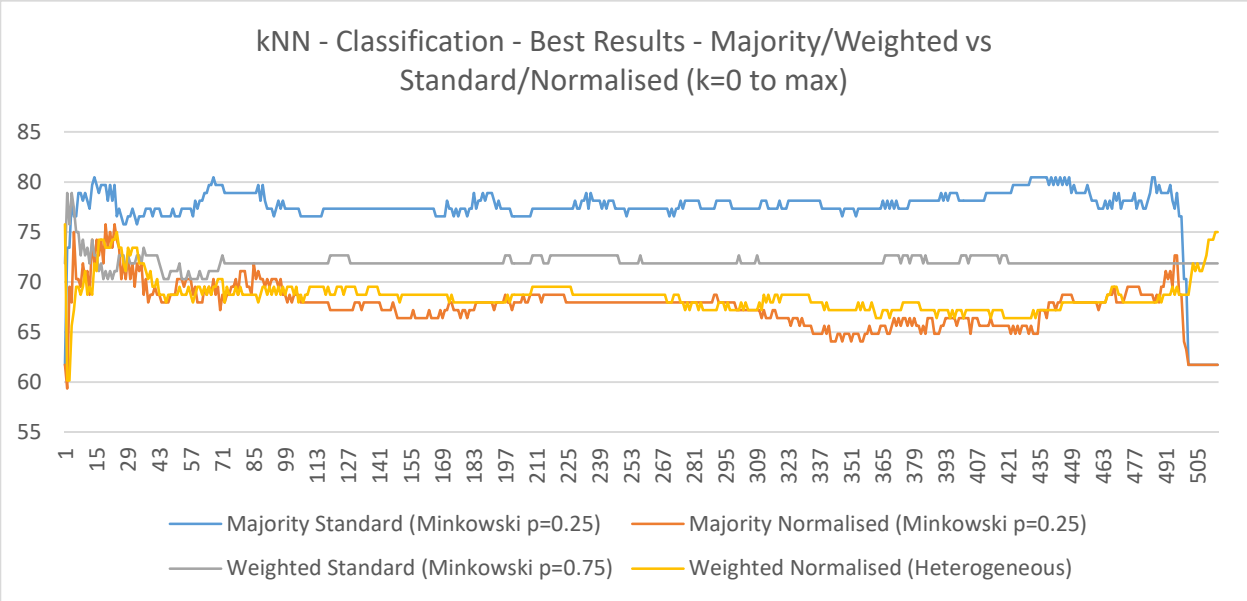


Looking at the highest accuracy achieved across all distance metrics using weighted voting and normalised data, Heterogeneous distance calculation and Minkowski distance calculation with low *p*-values were the

outright best performing. However these results may be misleading as to get the respective highest accuracy for each it was necessary to have substantially higher *k*-values than seen in any of the previous tests. Considering the accuracy values do not beat those of standard data or majority voting, the impact on performance for choosing this combination of hyper-parameters may rule this solution as impractical when running it against larger data sets.

Distance	k Value	% Accuracy
Euclidean	508	68.75
Manhattan	49	71.09375
Heterogeneous	514 (all)	75.78125
Minkowski (p=0.25)	25	74.21875
Minkowski (p=0.5)	24	74.21875
Minkowski (p=0.75)	25	74.21875
Minkowski (p=1.0)	49	71.09375
Minkowski (p=1.5)	9	68.75
Minkowski (p=2.0)	508	68.75
Minkowski (p=2.5)	13	68.75
Minkowski (p=3.0)	506	68.75
Minkowski (p=3.5)	511	68.75
Minkowski (p=4.0)	477	67.96875
Minkowski (p=4.5)	505	67.96875
Minkowski (p=5.0)	149	68.75

Overall, when comparing the results of the majority and weighted tests using both standard and normalised data, the Minkowski distance metric was by far the most successful with usage in three of the four best results. It exhibited the best accuracy for lowest *k*-value with weighted voting and standard data, and the highest accuracy with majority voting and standard data.



It can also be said, with the exception of weighted normalised data, that lower k-values are better for providing the best accuracy. In almost all tests, the best accuracy values were found where $k < 25$ which is approx. 6% of the total amount of instances in the training set. This would fall in line with the method for determining the best k-values as the square root of the total amount of instances in the training data set. The square root of 514 is 22.67 so the result of majority of best accuracies occurring where $k < 25$ seems to fit this assumption almost perfectly.

In the tests run for *k*-NN and classification against this training and test data set it is also apparent that normalised the data does not provide better results than the standard data set. This finding however can only be seen as applicable to this exact test-scenario, given another data set normalisation could and should increase the accuracy where there are vastly different ranges of values being compared.

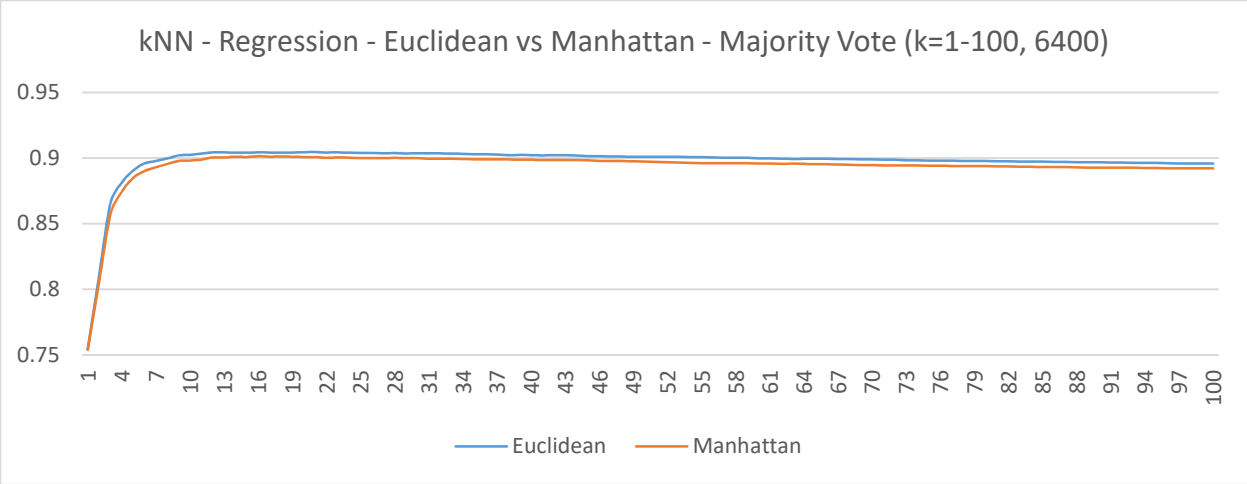
Part 3 – Hyper-Parameter Optimisation with k-NN for Regression

When testing the effect of hyper-parameter optimisation with *k*-NN for regression problems, not all of the same distances were used as with the classification testing. In this series of tests the heterogeneous distance was not implemented as there was no data available about the data set to indicate if it had discrete data as well as continuous data. As such, it was assumed that all data within was continuous data so the main focus on testing using Euclidean, Manhattan and Minkowski distances.

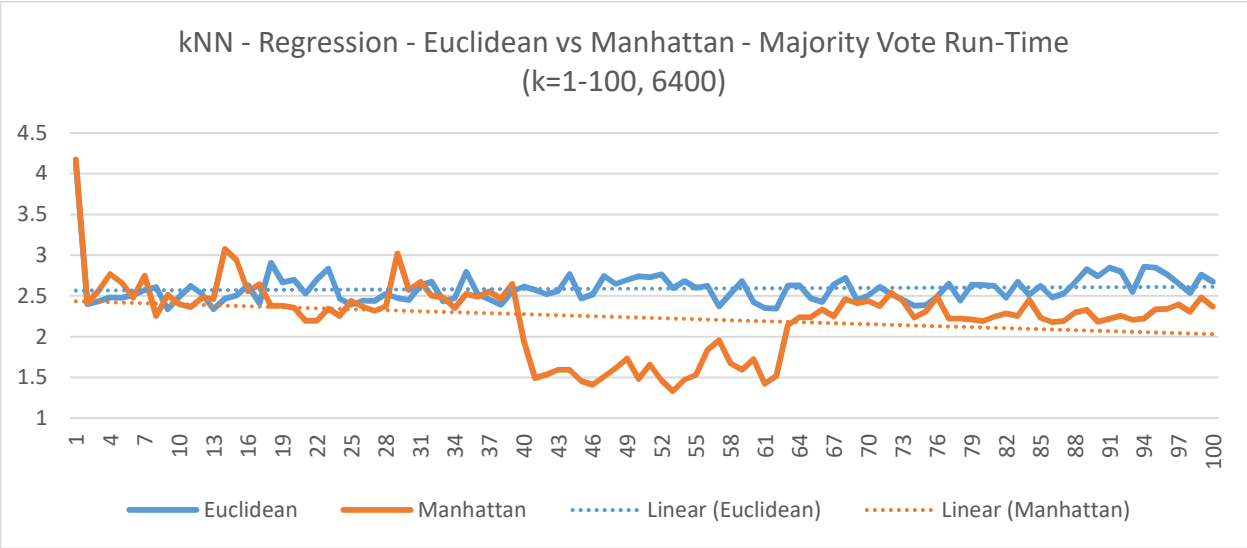
Another difference in testing the regression data is the range of *k*-values used. As there were 6400 instances in the training data set, it was computationally expensive and time-consuming to calculate the r2 value for every *k*-value between 1 and 6400. To lessen the impact of analysing all *k*-values, the range of k-values from 1-100 was analysed and the max value of 6400 to determine if analysing all neighbours had an impact on the final r2 value.

Testing with the Minkowski distance and normalised data was also left out of these tests. Both the training data and test data did not contain any feature values which negatively impacted the final prediction result so it was not deemed necessary, therefore the Minkowski distance metric was only tested with the standard data set. The time taken to run the *k*-NN algorithm against the Minkowski distances for regression predictions was significantly longer than any other tests, so this also had a bearing on the decision – each *k* value was taking 7-9 seconds to run, across all Minkowski *p*-values this resulted in a total run-time of 1.5-2 hours.

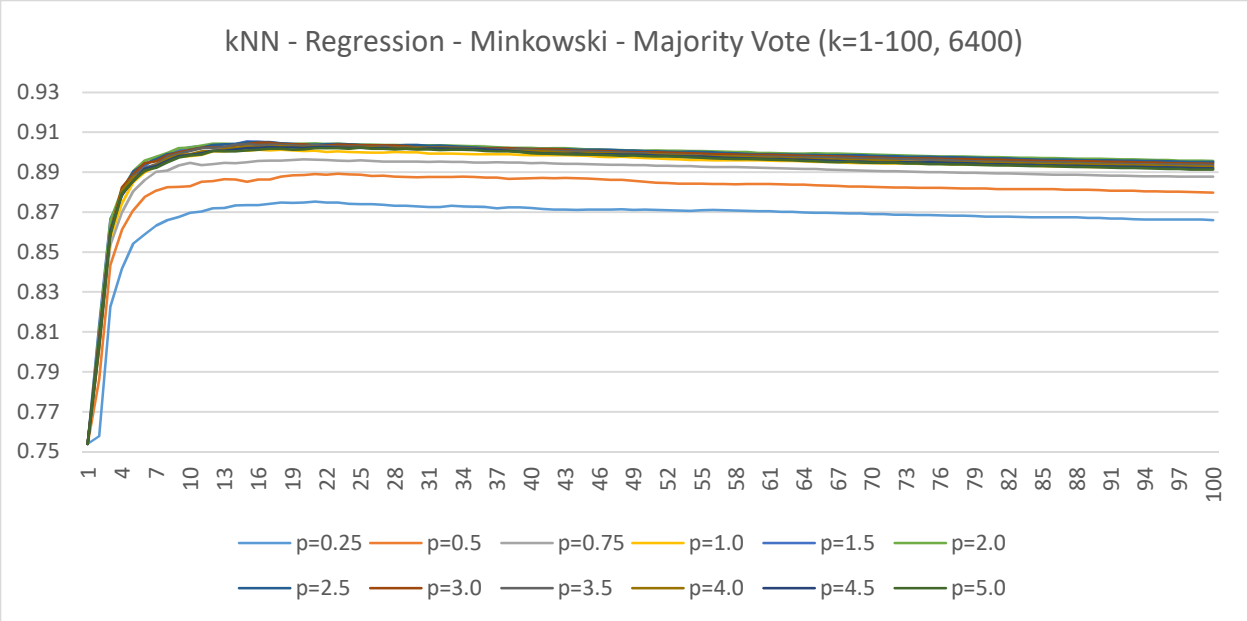
For the regression *k*-NN tests, majority voting with standard data was analysed first, in the tables below the first value is the accuracy for k=6400, the next value is k=1. This pattern is maintained for all tests and charts. For the Euclidean distance, k=20 resulted in the best r2 value of 0.9044882, for Manhattan the r2 result was slightly lower with k=15 resulting in an r2 value of 0.901331. There was a decreasing trend observed after these high points, with the r2 value getting gradually smaller as the k-value increased. The r2 values for k=max were 0.75385573 and 0.753856 respectively. Although these r2 values are not bad, they are still in the top 25% of potential r2 values, they are not as impressive of r2 values in the top 10% of r2 values of 0.9+.



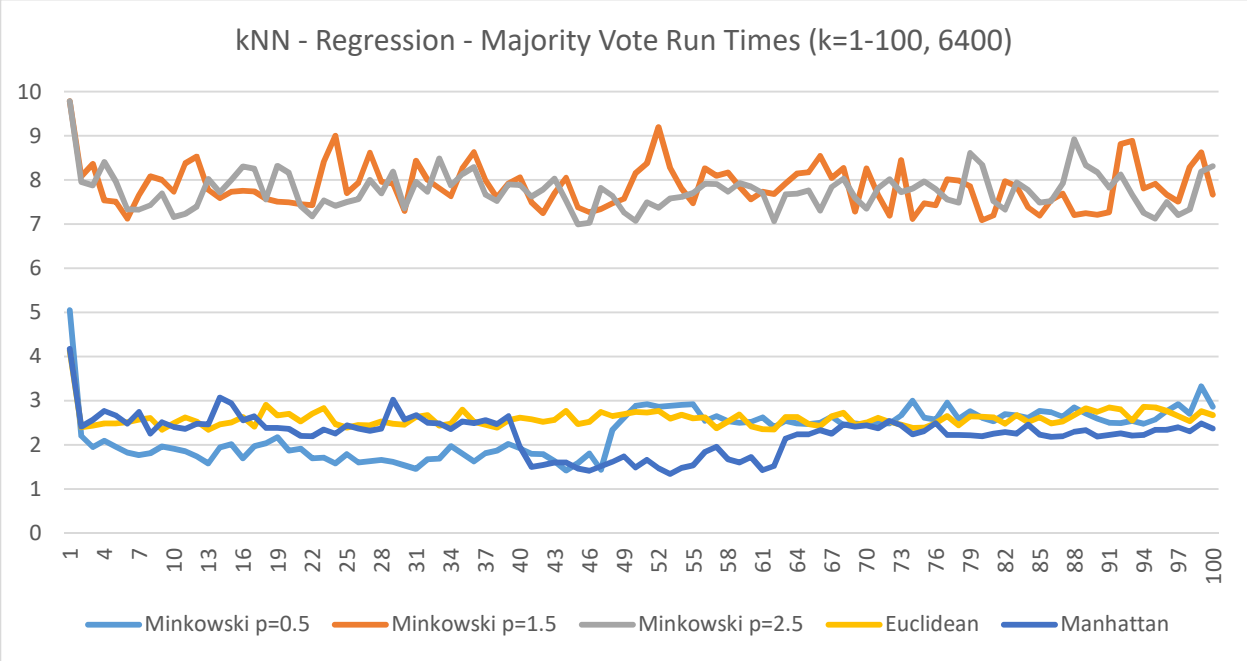
Looking at the run-times involved in determining the best *k*-value for a regression problem, an increasing trend can be seen for the Euclidean distance. The same cannot be said for the Manhattan distance but this is likely due to the abnormal drop in run-times for *k*-values between 40 and 60 which were likely as a result of the environment in which the algorithm was running. It would be expected to see an increase in the run-time value as *k* increased to the point where the *k*=6400 value run-time value of 4.171394s would be reached.



Using the Minkowski distance with the regression data, higher r2 values were possible that those found with Euclidean and Manhattan. A p-value of 1.5 resulted in an r2 value of 0.905362804 when *k*=14. Interestingly, the lowest r2 values were found when the p-value was lower than 1.0, the opposite of the behavior seen when using Minkowski distance for classification problems. The trend of decreasing r2 values after the peak r2 value was also observed for the Minkowski distance.

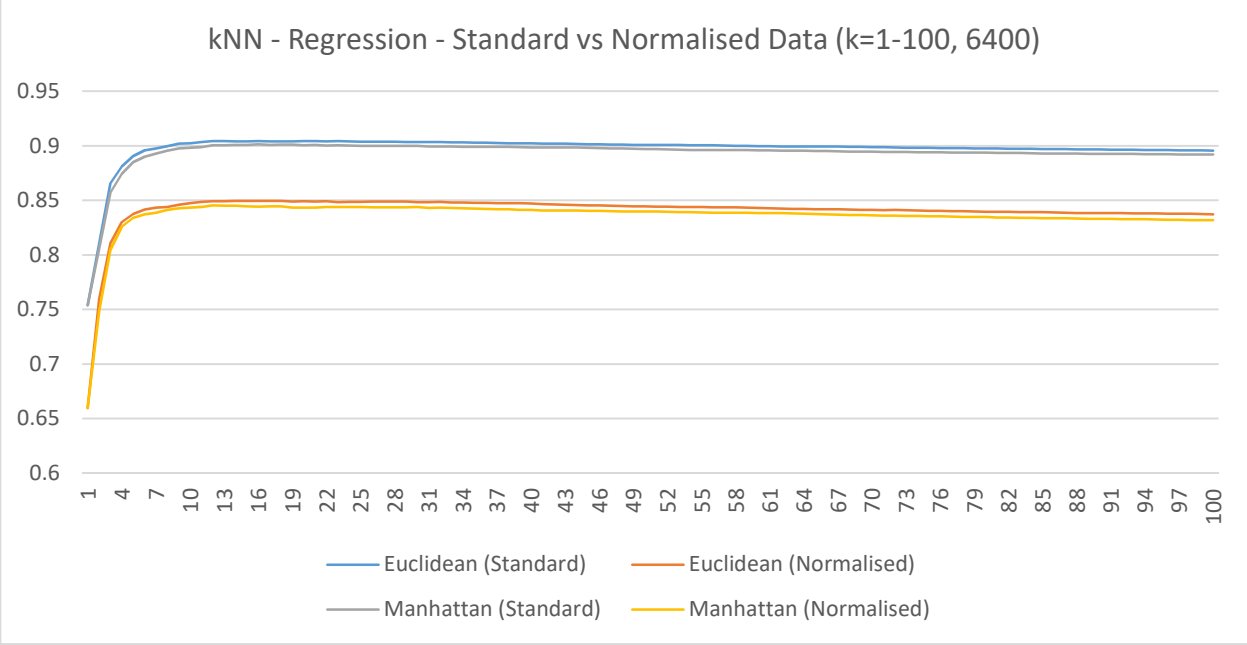


When looking at which algorithm hyper-parameters to use for a given scenario, it is important to take into consideration the run-times involved. Although Minkowski distance calculations have displayed the best r2 results when using majority voting and standard data, the increase in run time is almost treble that seen for the next best r2 value provided by Euclidean distance. When increasing the size of the data set, the decision needs to be made if a negligible increase in the r2 value is worth the three-fold increase in time required to get a similar result.

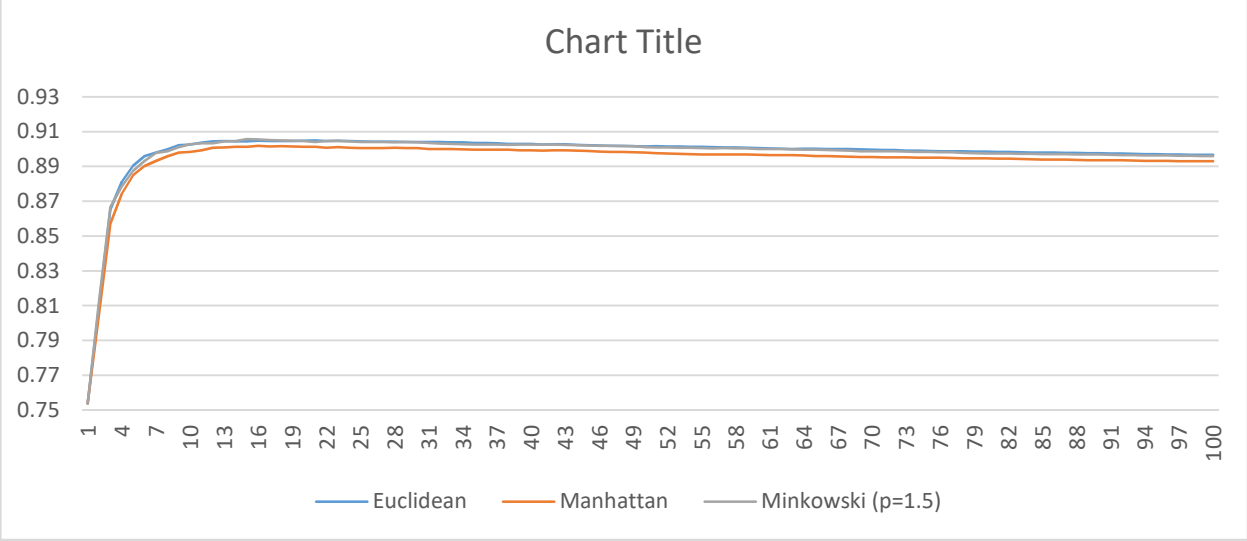


The effect of normalising data when the data did not contain large variations in feature values was also looked at, through the impact of normalising the data for Euclidean and Manhattan distances for majority

votes. In both instances, the normalised data exhibited lower r2 values for each distance metric when compared to the standard data version. This is an expected result, it was deemed that normalisation of the data was irrelevant and would provide no tangible improvement in performance of the regression algorithm.

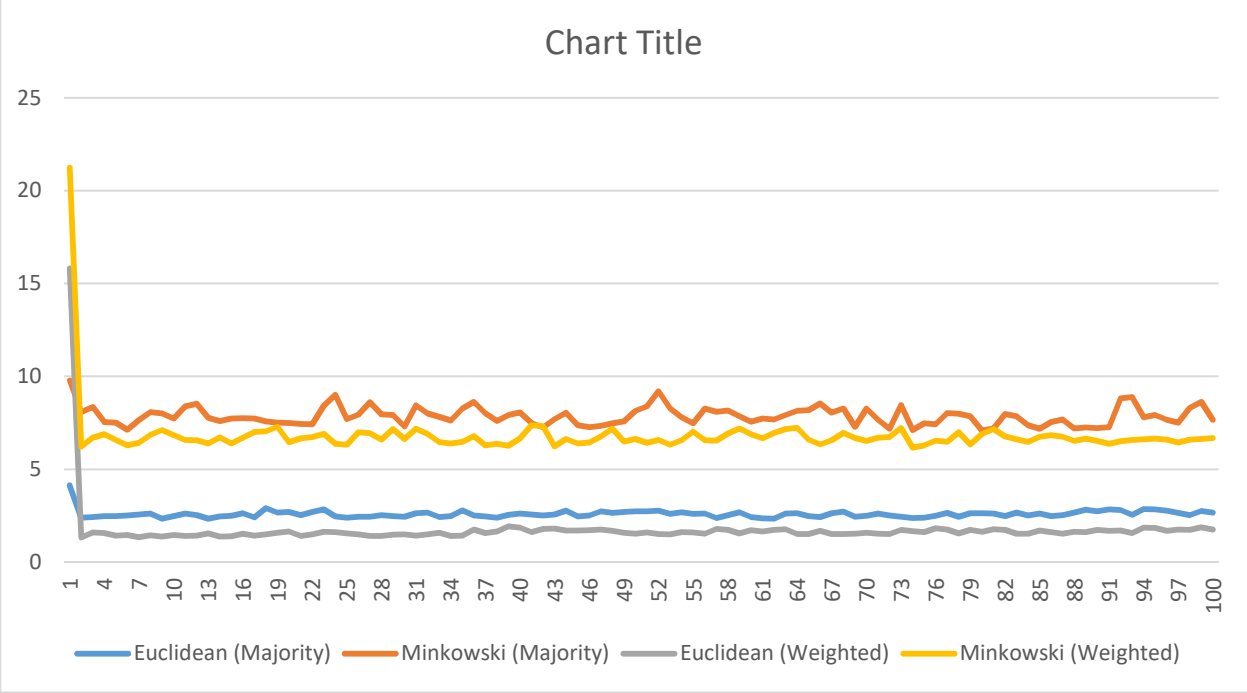


Great results were seen from the majority vote driven regression algorithm tests, with an impressive high from the Minkowski distance calculation again providing the best performance with an r2 value of 0.905362804 albeit with a slower run-time that the Euclidean alternative of 0.9044882. Similar tests were run with distance-weighted voting to gauge any further possible improvement in accuracy and r2 value.



Both Euclidean and Minkowski (p=1.5) distances both provided the best r2 values with 0.904978 and 0.905652502 respectively. Both these values were reached with a similar k-values to their majority voting equivalent (k=20 for Euclidean and k=14 for Minkowski), but there were some surprising results in terms

of run-times for each. For Euclidean distances, the run-time increased when using weighted votes, but the run-time decreased for Minkowski distances using weighted votes. This does not apply to cases where *k*=max value however as there was substantial increases in the run-time for weighted votes where all neighbours were part of the calculation.



Overall, the key piece of knowledge taken from the regression tests for this particular data set and scenario is the impact trying to increase the r2 value further has on the run-time required. The r2 value can be increased further but the impact on the run-time may very likely make the improvement unfeasible due to the amount of time taken to achieve that improvement. For a fitting increase of less than 0.001 is it worth having an algorithm run three times longer, I do not believe so. I think it would be more effective to have a more efficient algorithm to process more training data to improve results than increase the accuracy of a given algorithm for a fixed training data set size.