TRIED EXO24 Report

Carl Robinson - 12th Nov 2017

K-Nearest-Neighbours and K-means applied to Fisher's Iris dataset

0) Objective, Data and Methods

Objective

To explore and compare the behaviour of supervised and unsupervised classification algorithms by applying the k-nearest-neighbours and k-means algorithms to Fisher's Iris dataset.

Dataset

Fisher's Iris dataset is a very well known multivariate data set introduced in 1936 by the British statistician and biologist Ronald Fisher as an example of linear discriminant analysis. The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimetres.

Methods

The **k-nearest neighbors algorithm** is a non-parametric supervised classification method, where the input consists of the k closest training examples in the feature space, and the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. Either Euclidean distance or Mahalanobis distance can be used to determine which examples are considered to be an object's nearest neighbours.

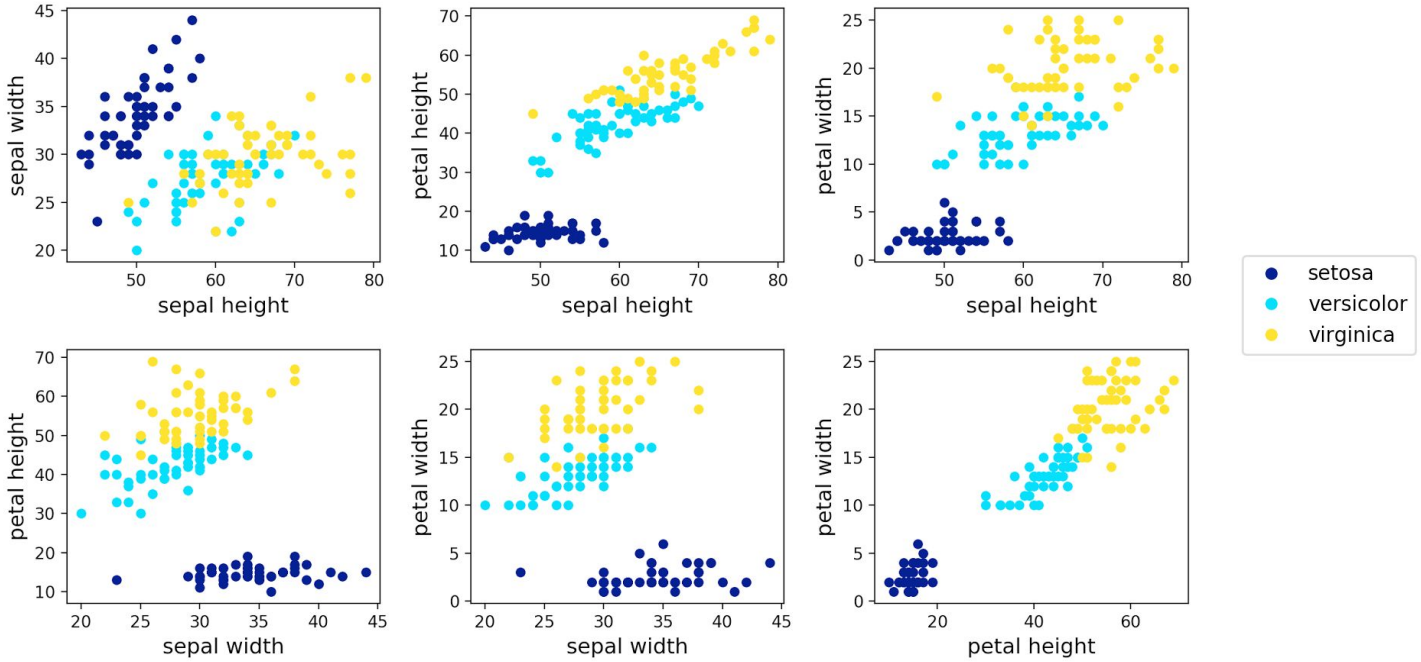
The **k-means algorithm** is an unsupervised clustering algorithm that partitions n observations into k clusters, in which each observation belongs to the cluster with the nearest mean, serving as the centroid of the cluster. This is an NP-hard problem, so heuristic algorithms are often used to converge quickly to a local optimum. K-means is an iterative algorithm that starts with a random initialisation step, then alternates between and assignment step and an update step:

- The initialisation step is where k observations are randomly chosen from the data set and used as the initial means (centroids). The value of k is chosen manually, based on some a priori knowledge of the domain.
- The assignment step is where each observation is assigned to its 'nearest' cluster, whose mean (centroid) has the least squared Euclidean distance from it.
- The update step is where the centroids of the clusters are updated, each taking the mean value of the observations now assigned to it.

The algorithm has converged when the assignments no longer change. Note that as it is a heuristic algorithm, it is not guaranteed that the global optimum will be found, as the result can depend on the initial cluster initialisation.

1) Data representation

Variables plotted against each other 2-by-2



- We can see that setosa have the smallest petals, virginica the largest petals, with versicolor in-between these two. Setosa have short wide sepals, while virginica and versicolor both have generally narrower and taller sepals than setosa.
- There is a clear linear separation between the setosa class and the other two classes for all 6 pairs of variables, whereas virginica and versicolor are never completely linearly separable.
- Versicolor and virginica are much more similar to each other than they are to setosa, as shown by the close proximity of their examples in all 6 plots. This is best seen in the sepal width against sepal height figure, where the versicolor and virginica examples are somewhat mixed. This suggests that classification accuracy for setosa is likely to be higher than the other two classes, as it is linearly separable.
- The examples are all fairly homogenous in all 4 variables, as seen by the fairly tight grouping of each class. Setosa is the most tightly grouped and therefore homogenous of the three classes, whereas virginica is the least homogenous.

2) Splitting the data into train and test sets

The data was split into three configurations of train/test proportions of the total number of examples:

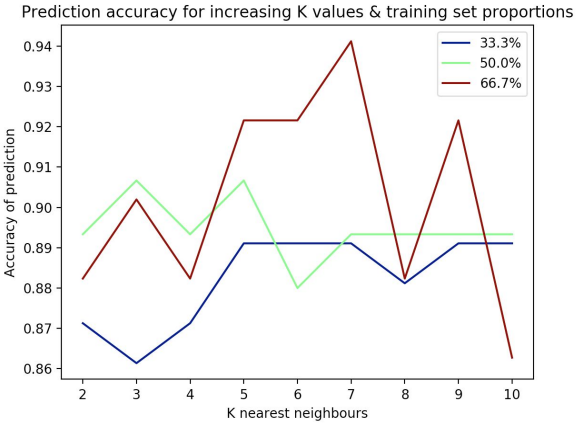
1. 33.3% train, 66.6% test
2. 50.0% train, 50.0% test
3. 66.6% train, 33.3% test

The k-nearest-neighbours algorithm and the k-means algorithms were then trained and tested using each of these train/test proportions, and the results were analysed and compared.

3b) Classification with K-nearest-neighbours algorithm

For each of the three train/test proportions, the k-nearest-neighbours algorithm was used to classify the test set using increasing values of K from 2 to 10. The classification accuracy was then calculated as number of correctly classified test set_examples divided by the total number of examples in the test set.

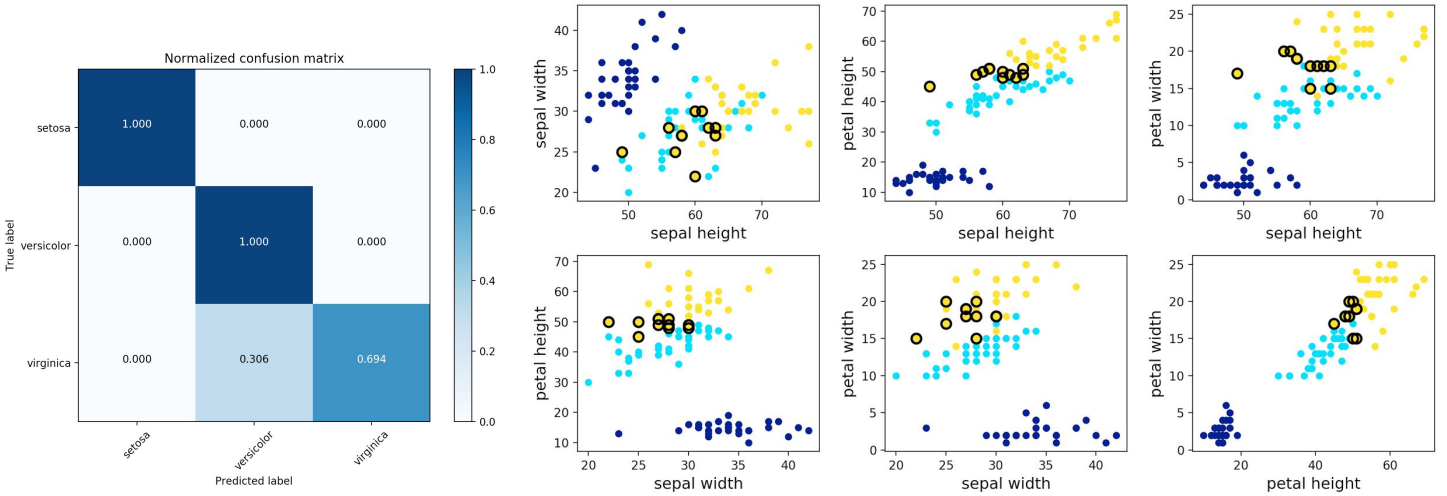
K	Train = 33.3%	Train = 50.0%	Train = 66.6%
2	0.871287128713	0.893333333333	0.882352941176
3	0.861386138614	**0.906666666667**	0.901960784314
4	0.871287128713	0.893333333333	0.882352941176
5	**0.891089108911**	0.906666666667	0.921568627451
6	0.891089108911	0.88	0.921568627451
7	0.891089108911	0.893333333333	**0.941176470588**
10	0.891089108911	0.893333333333	0.862745098039



- The results are tabulated above, with the highest accuracy for each train/test proportion highlighted in bold. The highest accuracy obtained can be seen to increase as the proportion of examples used for training increases. This suggests that a larger test set results in more accurate classification.
- Plotting the accuracies on a figure highlights this trend, with a clear vertical arrangement of plots.

<u>Train/test proportions: 33.3% train, 66.6% test</u>

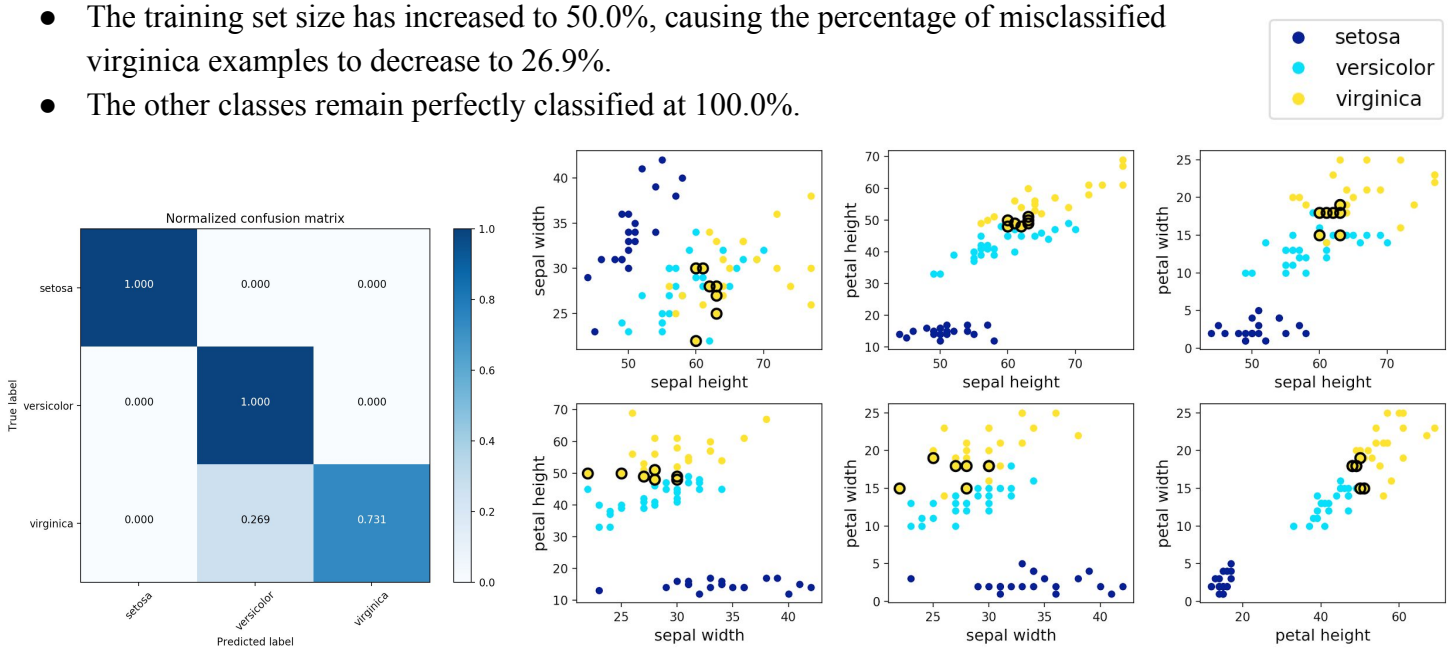
- For this configuration, the highest accuracy was 89.1%, obtained when k=5.
- The confusion matrix clearly highlights the fact that all setosa and versicolor test examples were correctly classified. Only virginica examples were misclassified.
- Specifically, it shows that 30.6% of virginica examples were misclassified, and all of these were misclassified as versicolor.



- The misclassification of virginica as versicolor is due to the fact that examples from these two classes are located very close to (and often overlap) the decision boundaries that separate them across all 6 pairs of variables
- In particular, the sepal height versus sepal width figure shows that many of the misclassified are located on the opposite side of the decision boundary, and a significant distance away from it (i.e. the yellow markers that are in the middle of the cloud of blue markers). For these misclassified points, the k-nearest-neighbours algorithm is likely to have considered many points of the opposite true class when calculating the classification, due to their euclidean proximity.

Train/test proportions: 50.0% train, 50.0% test

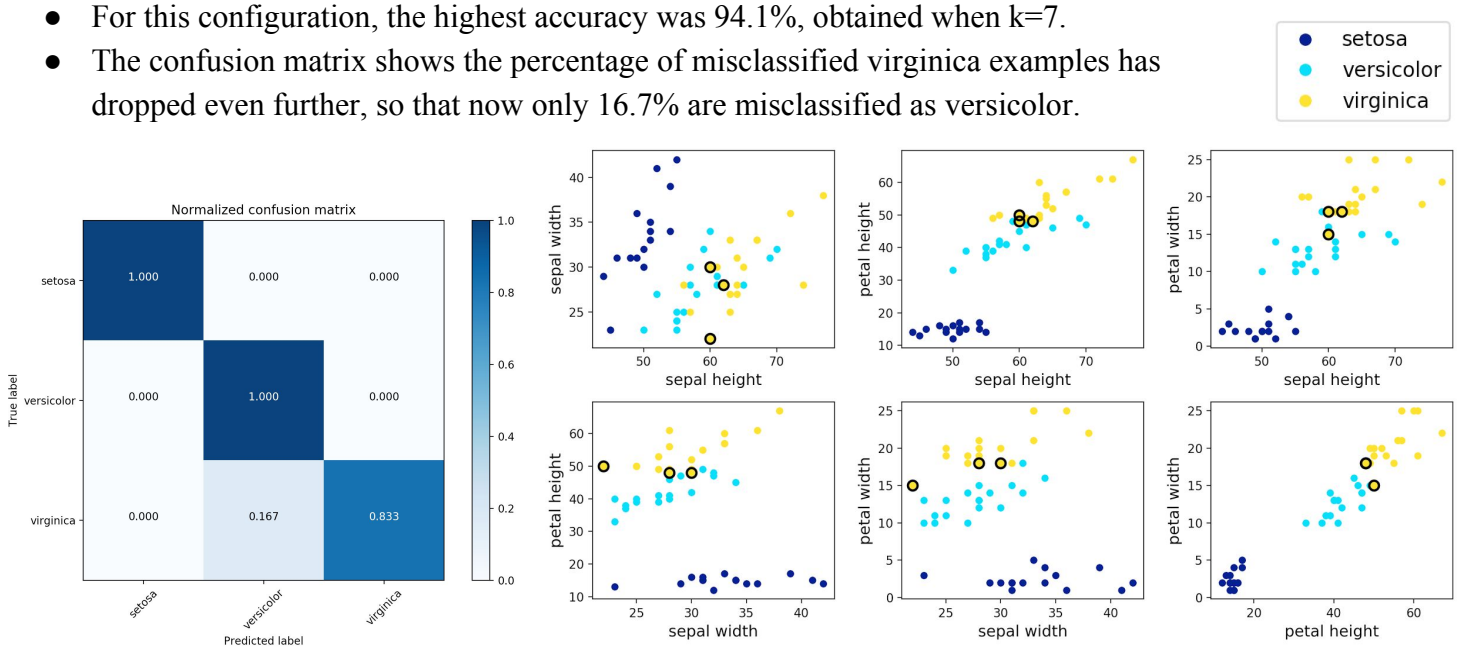
- For this configuration, the highest accuracy was 90.7%, obtained when k=3.
- The training set size has increased to 50.0%, causing the percentage of misclassified virginica examples to decrease to 26.9%.
- The other classes remain perfectly classified at 100.0%.



- There are fewer test examples in this configuration, so consequentially there are fewer points in the plots and fewer misclassified points.
- The improved accuracy obtained by increasing the training set size means that the misclassified examples are now more tightly grouped around the imaginary decision boundary between versicolor and virginica.
- In terms of the k-nearest-neighbours algorithm, these examples are the the most ambiguous to classify, as each one is closely surrounded by relatively equal numbers of versicolor and virginica examples. Therefore the sum of the distances to the training examples of each class is likely to be very similar, and the classification result can easily be affected by natural variation in the variables.

Train/test proportions: 66.6% train, 33.3% test

- For this configuration, the highest accuracy was 94.1%, obtained when k=7.
- The confusion matrix shows the percentage of misclassified virginica examples has dropped even further, so that now only 16.7% are misclassified as versicolor.



- This is the configuration with the smallest test set, leading to the fewest test examples on the plots, and the fewest misclassified examples.
- Also, a very large training set results in better accuracy, so there are fewer misclassified examples.
- The misclassified are all located along the imaginary decision boundary between versicolor and virginica.

3c) Classification with K-means algorithm

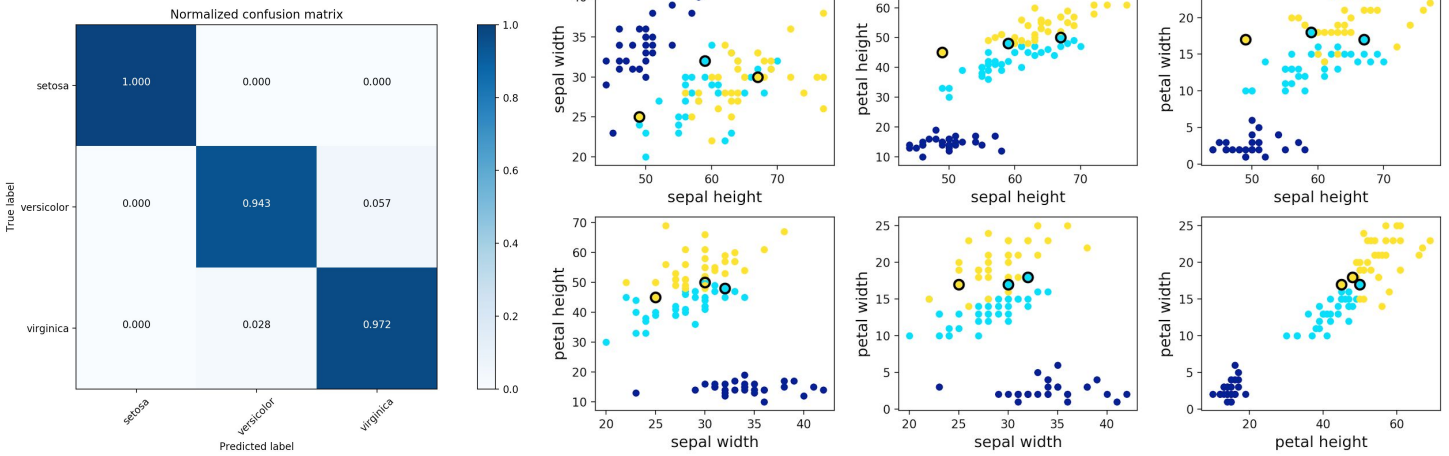
For each of the three train/test proportions, the k-means algorithm was used to classify the test set using increasing values of K from 2 to 10. The classification accuracy was then calculated as number of correctly classified test set_examples divided by the total number of examples in the test set.

K	Train = 33.3%	Train = 50.0%	Train = 66.6%
2	0.613861386139	0.626666666667	0.647058823529
3	0.861386138614	0.853333333333	0.823529411765
4	0.613861386139	0.920000000000	0.862745098039
5	0.861386138614	0.853333333333	0.921568627451
6	0.861386138614	0.920000000000	0.941176470588
7	0.851485148515	0.920000000000	**0.980392156863**
8	**0.970297029703**	0.853333333333	0.980392156863
9	0.920792079208	0.920000000000	0.882352941176
10	0.841584158416	**0.986666666667**	0.980392156863

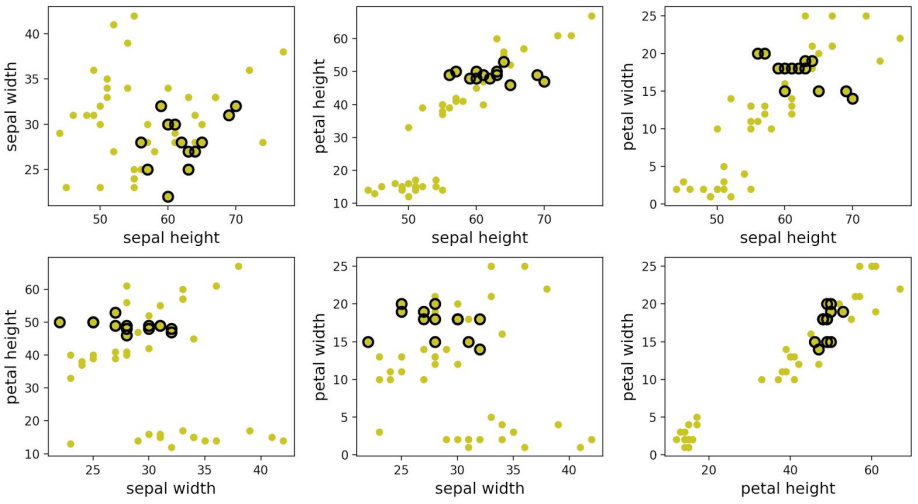
- The results are tabulated above, with the highest accuracy for each train/test proportion highlighted in bold. The accuracies are all significantly higher than those obtained with k-nearest-neighbours algorithm, suggesting that a clustering method is more appropriate for this dataset.
- Surprisingly, the highest accuracies obtained are for values of k >> 3, even though we have a priori knowledge that the actual number of classes is 3.
- Running the k-means training multiple times yields significantly different test set classification accuracies each time. The above table shows just one of these results. As mentioned earlier, this variation is due to the random initialisation of centroids leading to a convergence to local optima as opposed to global optima.
- To solve this issue and find the global optimum for each training set / K value combination, it would be necessary to run the k-means algorithm many times and take the highest accuracy obtained for each combination. Then a fair comparison between K values could be made.

Train/test proportions: 33.3% train, 66.6% test

- For this configuration, the highest accuracy was 97.0%, obtained when k=8.
- The confusion matrix shows that two versicolor examples were misclassified as virginica, and one virginica example was misclassified as versicolor. All setosa examples were correctly classified. This pattern of misclassification is very similar to the k-nearest-neighbour results.
- This is the configuration with the largest test set, leading to the most test examples on the plots, and the most misclassified examples.
- As with the k-nearest-neighbours algorithm, the misclassified examples are all located along the imaginary decision boundary between versicolor and virginica.

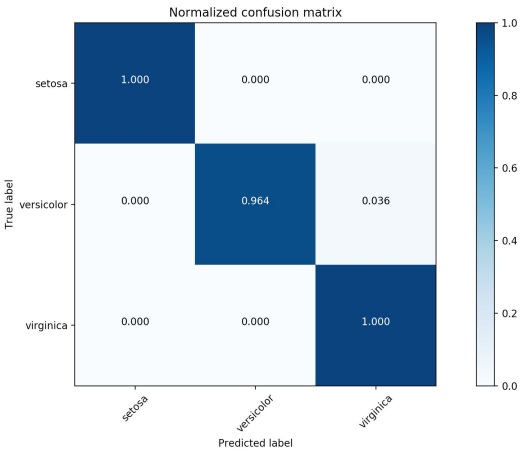


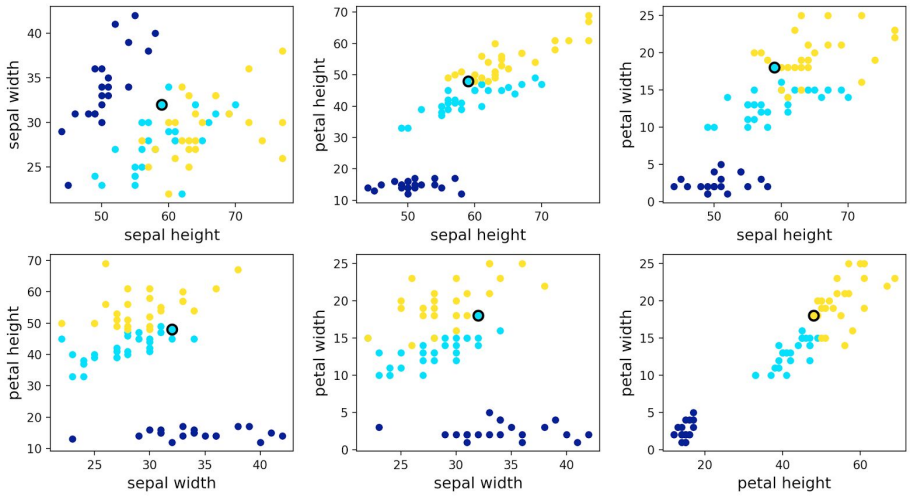
- To eliminate the effect of the random initialisation, the k-means algorithm was run 25 times. The examples which never changed class over all 25 runs can be regarded as ‘strong form’ examples.
- The figures to the right show the examples which changed at least once (i.e. the encircled examples are NOT strong form).
- It can be seen that these elements all lie in the region of space that is occupied by both the virginica and versicolor classes, and in particular they are grouped along the imaginary decision boundary that would separate the classes if it existed.



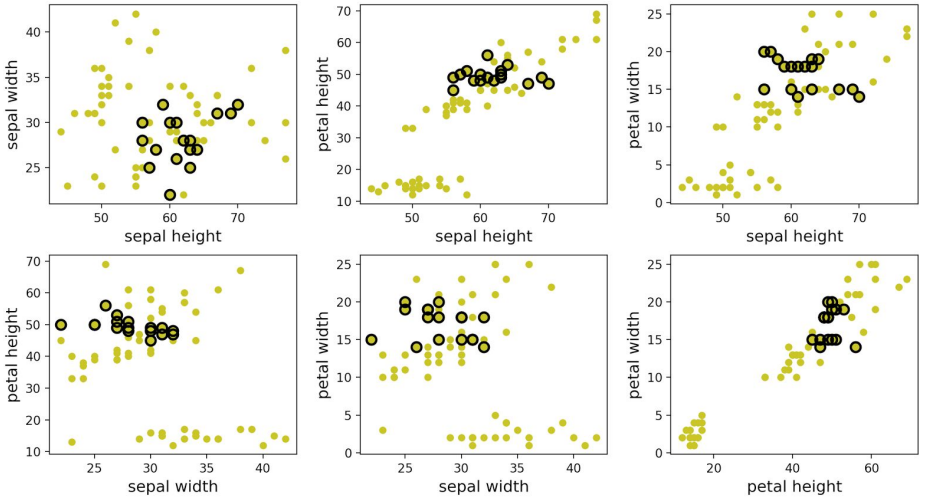
Train/test proportions: 50.0% train, 50.0% test

- For this configuration, the highest accuracy was 98.7%, obtained when k=10.
- The confusion matrix shows that only one versicolor example was misclassified as virginica. All setosa and virginica examples were correctly classified.



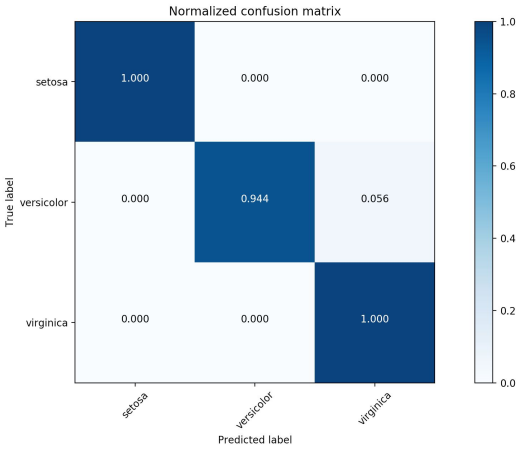


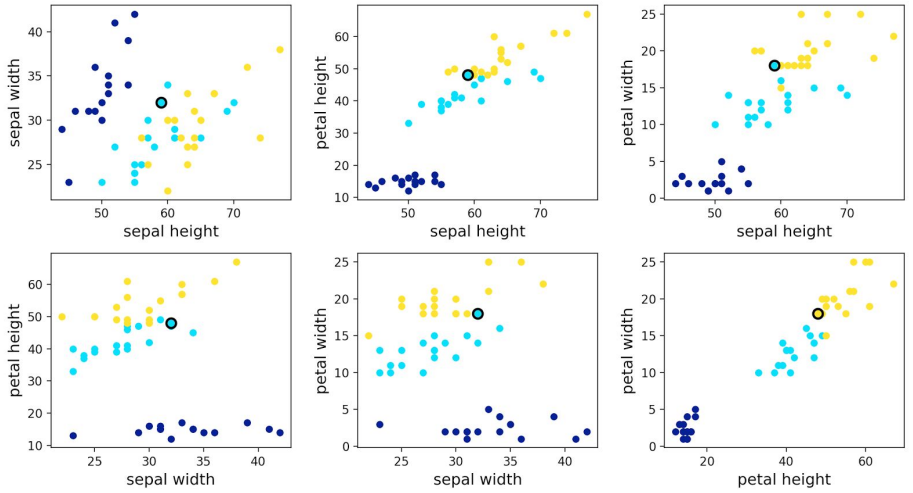
- While there are fewer test examples in the plots below, the encircled examples that remain have not changed status.
- This shows that strong form examples remain strong form regardless of training set size.



Train/test proportions: 66.6% train, 33.3% test

- For this configuration, the highest accuracy was 98.0%, obtained when k=7.
- The confusion matrix shows that, again, only one versicolor example was misclassified as virginica. All setosa and virginica examples were correctly classified.





- There are still fewer test examples, yet the same strong form and non-strong form examples retain their status.