University of Bristol

Symbols, Patterns & Signals

COMS21202

BsC Computer Science

Classifying Wine

Author 1:
Josh Hamwee

Author 2:
Aidan Hood

Email:
jh17602@bristol.ac.uk

Email:
ah16165@bristol.ac.uk

May 1, 2019

Contents

1 Implementation

1.1 Feature Selection

For the feature selection, we used a method that involved us creating 169 scatter graphs of each feature against the other. We then went through each one trying to decide which splits up the data the best. From here, we selected the pairs that clearly separated the data the best and then analysed in more detail the variances of the classes, the apparent correlations, and the degree of separation from other classes for each of the pairs that looked good.

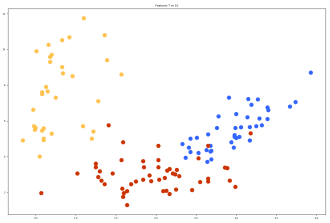


Figure 1: 7 vs 10

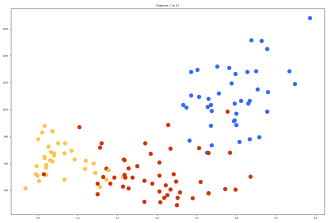


Figure 2: 7 vs 13

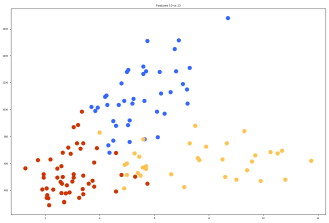


Figure 3: 10 vs 13

All three pairs were very good at splitting up the data into its specific classes. The 7 vs 10 pair of features was in our opinion the best separator of the three classes as it had good overall separation despite having no linear separation. 7 vs 10 also had the lowest average variance of the individual classes, despite 7 vs 13 having a really low variance for the yellow class. It also had the clearest correlations within classes compared to the other two pairs of features. 7 vs 13 was a close second place in our selection, and we believe that it would have worked nearly as well as 7 vs 10 if not the same due to good separation and low class variance. 10 vs 13 looked good by eye initially, but upon closer inspection lacked in separation of classes due to overlapping of all three classes. Therefore, the feature pair 7 and 10 was our final choice.

Even though we managed to condense down the possible pairs of features to 3 by eye, it was still difficult to make a final decision on which looked like it would perform the best. It would require more scrutiny to make a better decision on the features, by running the classifier on multiple feature pairs to determine the best choice, or calculating accurate figures for qualities like the variance of the classes for a certain pair of features so that we had a more concrete set of metrics to judge the pairs on. As the number of features grows, this task becomes almost impossible to be performed manually by a human due to the large quantity of pairs.

1.2 K-Nearest Neighbours

K-Nearest Neighbours was not that much of an issue to implement as the algorithm is rather simple to reproduce. The way in which it works is that it finds the k nearest neighbours to the unknown point, and returns the class that was the most occurring. There are alot of positives to using this classifier, such as the fact that it has no training step, and hence no model is required to be made. There is also a large range of distances that you can use to measure the distances between points.

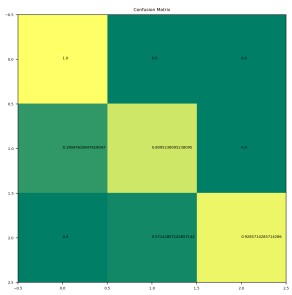


Figure 4: 1-NN

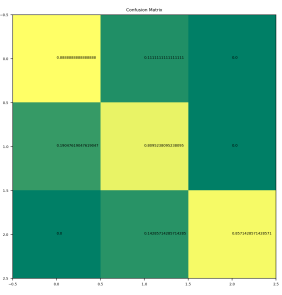


Figure 5: 3-NN

Figure 6: 5-NN

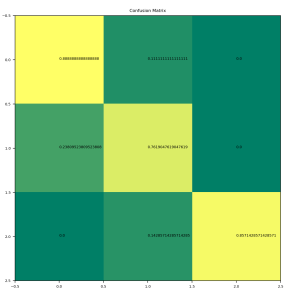


Figure 7: 7NN

We measured up to the value of k = 7. The results however were not quite what we expected as the accuracy decreased with an increase of k. It was a suprising result to us as we thought that the more neighbors the better the result would be, but this is not the case. This is because when the number of neighbors increases, the noise effects the classifier more. We also recognized that the algorithm was particularly sensitive to outliers when ran against the other features. Included are our confusion matrices for this classifier that show that there is a fairly low level of confusion due to the high numbers on the top left to bottom right diagonal. This is to be expected from this classifier as it performed very well. The level of confusion remains fairly constant with an increase in k; this is due to the algorithm scaling fairly well as k increases despite a loss in accuracy due to the added effects from noise.

1.3 3D-Nearest Neighbours

Similar to that of the 2DK-NN above, it was not very hard to implement 3DK-NN as it only required some slight changes to the original 2 feature version. For this classifier we chose to use the same features as 2DK-NN plus feature 13, as we believed it was the third best to classify the data. The choosing of a third feature was challenging because if we wanted to manually selected the best three features that would be too many scatter plots to compare in realistic terms. Thus, we instead compared all the features to our two existing features (7 and 10)and decided that 13 separated the data the best from this. However, we personally believe that a technique such as PCA is better for this, rather than increasing the dimensions.

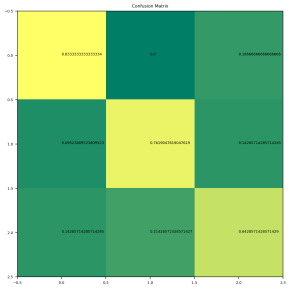


Figure 8: 1-NN

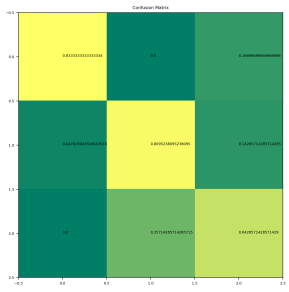


Figure 9: 3-NN

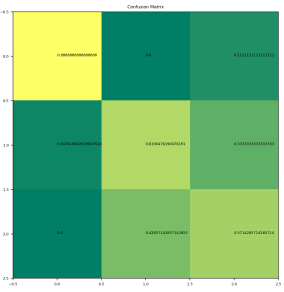


Figure 10: 5-NN

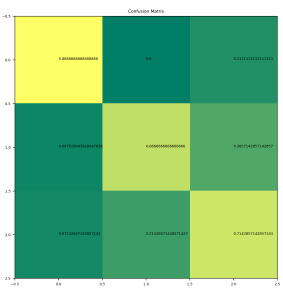


Figure 11: 7NN

What we observed from running this classifier was even more suprising. The results

were worse than that of the original knn. However, when we look deeper, this is actually what we would expect. It comes down to the fact that the knn classifier has the Curse of Dimensionality. Included are our confusion matrices for this classifier that show slightly more confused results than that of the 2D KNN classifier. We think that this is due to the 'vanishing neighbourhood' phenomenon, by which as you increase D the nearest neighbour's become much worse at accurately predicting the class of the subject data point. Interestingly, this effect seemed to be enhanced as you increased k, probably due to an increased effect from this dimensionality issue. Overall however, the confusion matrices showed fairly un-confused results on the whole.

1.4 PCA

Our final implementation of the KNN classifier was with a PCA reduced dataset. This had a major advantage that we did not need to select features for the classifier to run with; instead the PCA algorithm reduced the data sets down from 13 features to 2 via a PCA embedding. This had a number of benefits, but the primary benefit was that this was now really scalable as you increase the number of features the wine is classified under as you no longer need to conduct manual feature selection on a huge amount of feature pairs. It also removes redundant dimensions from the data and reduces over-fitting. With this in mind, the results were worse than that of the 2D-KNN classifier for all k, but this is to be expected as you are not selecting the most separating features, but rather embedding all the features into a 2 dimensional space.

Despite this, we would argue that this is potentially the best KNN classifier, due to the fact that it is highly scalable unlike the other two KNN classifiers and the results are by no means bad. Included is our scatter graph for the PCA reduced data set which shows that when reduced to two features there is significantly more overlap between the classes, which was to be expected. We also included our confusion matrices for this classifier that show a low level of confusion for this classifier, perhaps due to 13 features not being that high comparatively to other machine learning tasks.

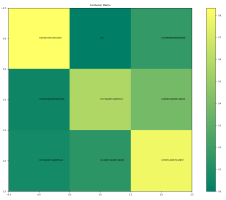


Figure 12: PCA-1

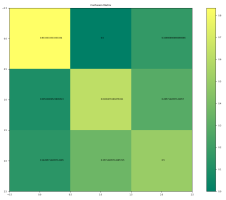


Figure 13: PCA-3

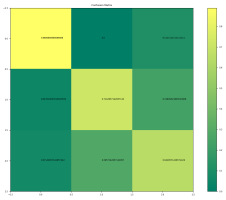


Figure 14: PCA-5

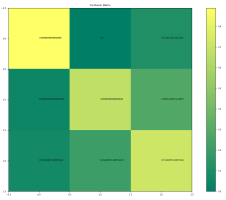
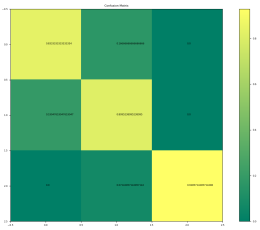


Figure 15: PCA-7

1.5 Naive-Bayes



For our alternate classifier we chose to implement the Naive-Bayes classifier. We thought that this was an elegant classifier, which produced good results with the second highest accuracy of all our implementations. We used a Gaussian distribution as this was the most appropriate approach for the continuous features. It had a high accuracy due to the fact that it is a robust algorithm that works well with our feature selection due to the features being independent from each other. Naive-Bayes also tends not to over fit which is another benefit, and the model size is relatively low in comparison to algorithms like Random Forest.

However, we can see that this classifier would break relatively easily if applied to data where the features are not conditionally independent, or more specifically when the feature dependencies do not cancel out. Included is our confusion matrix for this classifier which shows a very low level of class confusion which is to be expected due to the classifier working on Bayes theorem which would reduce this inherently due to probability.

2 Summary

The 2D KNN classifier had the best overall results, with a highest accuracy at 90.57%, and lower , although still very high, accuracy's for increasing values of K. We believe these are high because KNN operates best with a low number of dimensions and homogeneous features - both fulfilled with our classifier. The confusion of the results was also low, making this a good classifier overall. The only major limitation is that as you increase the number of features the feature selection becomes very hard.

The 3D KNN classifier had the worst results overall, ranging from 69.81% to 79.25%. This is understandable seeing as the KNN classifier scales very badly with an increase in dimensions as discussed earlier. The confusion also shows that as you increase k this problem is amplified, and although he results were not awful, this is the worst classifier we tested.

	K-NN/%	3DK-NN/%	PCA-KNN/%	Naïve-Bayes/%
K=1	90.57	75.47	71.70	84.91
K=2	86.79	69.81	64.15	
K=3	84.91	77.36	66.04	
K=4	86.79	75.47	71.70	
K=5	79.25	69.81	75.47	
K=6	84.91	79.35	81.13	
K=7	83.02	75.47	75.47	

The PCA accuracy was mixed to good, with highs of 81.13%. The confusion matrices showed the results were good, and this classifier is in our opinion the best as it is really scalable and removes the need for feature selection.

The Naive Bayes classifier produced excellent results, with an accuracy of 84.91%. The confusion matrices showed mostly un-confused results, however we are aware that this classifier is limited as soon as you introduce features that are not independent.

Overall we would recommend using the Naive Bayes classifier for data with independent features, or the PCA reduced KNN classifier as these are the most scalable and give the best accuracy.