Symbols, Patterns and Signals: Wine Classifier Report

Nicole Li (nl17247)
Sharlene D'Silva (sd17085)

April 2019

1 Introduction

In this report we discuss the results of our feature selection and classification process on the given wine data set. We used K-Nearest Neighbours Classifier with two and three features, Naive Bayes Classifier and Principal Component Analysis methods.

2 Feature Selection

There are many approaches to automatic and manual feature selection but in this report we used the manual method. We started by plotting all pairwise feature combination and, from the 13 x 13 options, eliminated all those where there is an apparent overlapping of the different classes. This was considered because the overlapping of classes would make finding a decision boundary rather difficult.

The shortlisted combinations include: 13 vs 7, 13 vs 6, 13 vs 10, 7 vs 6, 7 vs 11, 7 vs 13, 13 vs 12, 13 vs 11, 10 vs 13.

Next, by observing the remaining scatter plots, we prioritised those where each class is nicely clustered, then re-checked that the different classes had minimal overlapping. This was done as good clustering would allow to classify test points more accurately.

Since the shortlisted combinations do not vary greatly, we decided on features 7 and 10 as it fit our selection criteria subjectively. Figure 1 shows the scatter plot of our manually selected features.

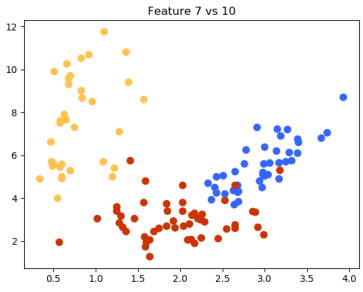

Figure 1: Selected features

Our biggest challenge in selecting features was that it was hard to compare all 13 x 13 plots and keep track of interesting feature combinations as most of the combinations gave similar scatter plots with minute differences.

3 k-Nearest Neighbours Classifier with two features

The KNN method works by classifying data according to the majority class of the test point's k nearest neighbours, which means the class assigned depends on the k closest training data points.

The table below shows the results obtained from running KNN classifier on the selected feaures.

Value of k	Accuracy
$k = 1$	0.90
$k = 2$	0.87
$k = 3$	0.85
$k = 4$	0.87
$k = 5$	0.80
$k = 7$	0.83

When k=1, the test points are classified according to the train point closest to them. According to our result, k = 1 has the highest accuracy recorded. This shows the merit of our feature selection- the train data points are nicely clustered and a clear boundary is somewhat identifiable.

Looking at the accuracy for other k values, there is small drop for when k is 3. This suggests that among the 3 points nearest to our test point, more neighbours belong to the incorrect class, resulting in the decreasing accuracy. This also signifies that noise data has become rather significant.

A solution is to select a larger k value, thus increasing the chances of non-noise data being included. However, there is no guarantee that a larger k value would improve accuracy, which can be seen in our results when k=5. In this case, more noise data was included. If we look at k=7, accuracy increases again, meaning that more of the non-noise data was included making boundaries between classes less distinct.

As we can see, the accuracy of k-Nearest Neighbour depends on how our data is scattered, i.e how clear the decision boundary between data points are because this directly affects how the k values perform with the classifier. The ideal k value is one that is large enough so that the effect of noise data is minimised and small enough so that it is still efficient and does not consider only the main clustered part of a class, making the classifier largely biased.

Another discussion is even k values and their affect on the

accuracy. Although our results for k=2 and k=4 are fairly high among all k values, they could pose a negative effect if other features were to be selected. The k-Nearest Neighbours Classifier classifies according to the majority class of the k train points that are closest to the test point. When k is an even number, there is a possibility for neighbours to be evenly divided into two classes. Here it becomes difficult to choose the 'majority'. A fix could be to choose the class of the point that is closest to the test, but this still does not guarantee a model with higher accuracy and no overfitting.

We can further analyze our results by observing the confusion matrices for different values of k (Figure 2). Confusion matrices help us to take a deeper look into how accuracy is distributed among classes and allows us to find where the errors in classification lie.

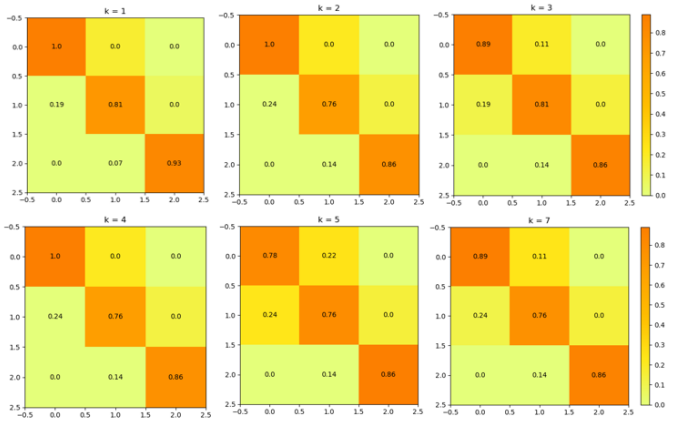

Figure 2: Confusion Matrices for k = 1, 2, 3, 4, 5, 7

In the confusion matrices, majority of the inaccuracies are due to class 1 (blue) being incorrectly classified as class 2 (red). This is due to the overlapping of some classes in Figure 1.

To further discuss decision boundaries: we can see that for the features we choose, the boundary between classes became rather unclear when the value of k increased from 1 to 3. The value remained consistent when k was increased to 7 for class 2 and 3 but not for 1 and 2. The boundary for class 1 and 2 became even more unclear when increased to 5, and slightly improved for k=7.

This is due to results varying depending on how scattered the data is and where the boundary lies. To understand this better, we use Voronoi diagrams to visualise. In the diagrams, the data points seen are the test points while the background regions are deduced using the train set.

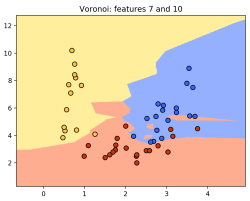

Figure 3: Voronoi diagrams when k=1

As seen in the Figures 3-4, there is a yellow data point that noticeably stays within the red region as k increases. This corresponds to the values in the confusion matrices. As considering neighbours for that test point, there are more red train data points considered than yellow, resulting in a slight expansion of the red region. Referring Figure 1, the small clustering of red data points might be the cause of this.

This could also be a reflection of our feature selection - if the clustering of a class is close to a boundary this would affect results. Although not obvious with our selected features, there has been a slight impact nonetheless.

On the other hand, if train set data are too sparsely scattered, the data could become an outlier and will not be significant enough to classify test points correctly.

Therefore, it is important to select features that show good clustering among classes, good separation and an appropriate k value for an efficient KNN model that does not overfit.

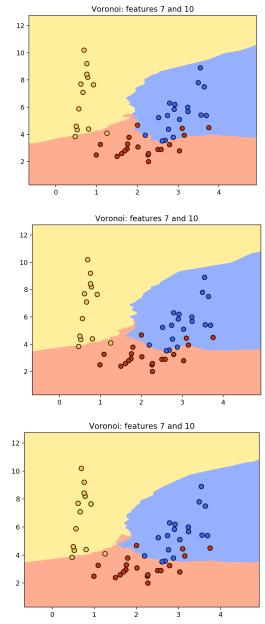

Figure 4: Voronoi diagrams when k=3,5,7

4 Alternative Classifier

For the alternate classifier, we decided to use Naive Bayes Classifier. The algorithm uses Bayes Theorem and makes the assumption that features are independent, i.e. the presence of one feature does not affect the other. It also assumes that the data is normally distributed. According to Bayes Theorem, the posterior values of a class, given the two selected features, are calculated by multiplying likelihood of each feature given the class with the probability of that particular class. The class with maximum posterior value is assigned to the test point.

Applying the Naive Bayes Classifier to the given data set, we obtained an accuracy of **0.85**. This value turns out to be the average of the results from the KNN classifier.

To analyze why this occurs we must first distinguish between the two classifiers. Firstly, KNN is a non-parametric algorithm, i.e. it makes no assumptions about the data. In addition, KNN's decision boundary depends on the classes of the training data and can thus take on any form. To visualise this, we plot the Voronoi diagram Figure 5 and compare it with Figure 3-4.

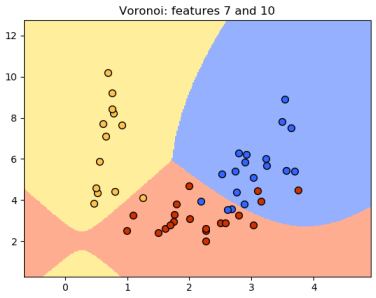

Figure 5: Voronoi Diagram for Naive Bayes Classifier

We observe that the decision boundaries in Voronoi diagram for Naive Bayes are smoother, whereas for KNN the rough boundaries indicate the flexibility of KNN. Let us consider what this means for the test data. In the case of KNN, as the boundary is adapted to the training data, test points of different classes that are close to each other are correctly classified. Whereas for Naive Bayes, the smooth curve incorrectly classifies the nearby and overlapping test points.

This concept has its advantages and disadvantages: depending on the data set and selected features, a rough boundary can easily cause overfitting and in that case Naive Bayes method would be preferred. However, the vice-versa is also possible and for certain data sets KNN will give higher accuracy (this is the case for our data set).

Secondly, as the Naive Bayes classifier is based on probability, it would perform better on larger data. This is reflected in our results as for our considerably small data set, the accuracy of KNN for certain values of k is higher than Bayes classifier.

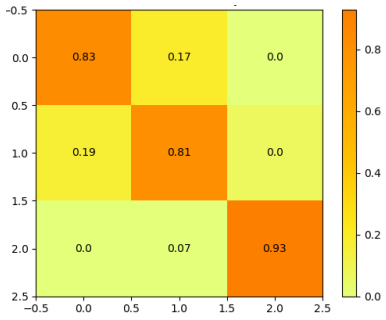

Figure 6: Confusion Matrix for Naive Bayes Classifier

Lastly, we observe the confusion matrix (Figure 6) to substantiate our above arguments. As the diagonal values are less than 1, we analyze that quite a few of the actual values of classes were incorrectly predicted. However, a large part of the error is seen in the incorrect classification of the red and blue class, which demonstrates the effect of the smooth boundary.

This could be avoided if different features were selected. In conclusion, depending on the data set and features chosen the results of both classifiers vary.

5 k-Nearest Neighbours Classifier with three features

To implement the KNN classifier with three features, we changed the distance measure to work with 3-dimensional data instead.

Selecting the third feature proved to be challenging as inspecting the 3-dimensional figures for significant details was hard. However, our aim was to look for good clustering within classes as well as decent separation between the different classes.

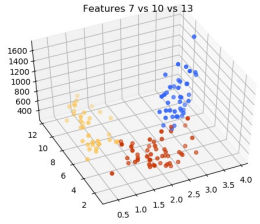

Figure 7: KNN with 3 features

From the 13 options, we selected the last feature, i.e 13, as the third feature. We chose this because, as observed in 7 the three classes seems to be clearly separated and there was minimum overlapping. Data points for each class also appears to be nicely clustered.

Value of k	Accuracy: 2 features	Accuracy:3 features
$k = 1$	0.90	0.75
$k = 2$	0.87	0.70
$k = 3$	0.85	0.77
$k = 4$	0.87	0.75
$k = 5$	0.80	0.70
$k = 7$	0.83	0.75

By applying the KNN classifier to the three selected features, the results were considerably less accurate than KNN with two features (as seen in the table above). This can be reasoned by the fact that the chances of a test point being classified to another class is increased. Unlike in 2-dimensions, where KNN classifies the test point to the majority class of k closest neighbours, 3-dimensions increases the scope of distances considered, and this is further proven with the confusion matrices below.

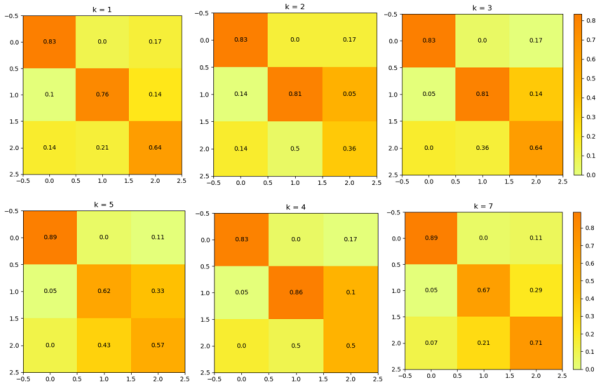

Figure 8: Confusion Matrices for KNN after PCA

Comparing the confusion matrices above with the ones for 2-dimension (Figure 2), we can see that the values for incor-

rect classification is split between two classes instead of one. For example, lets look at k=5 and k=7 for both class 2 in 2-dimensions and 3-dimensions. None of class 2 was incorrectly classified as class 3 in 2-dimensions, unlike in 3-dimensions. Meaning that the distance in 3-dimension is closer than in 2-dimension, supporting our reasoning above. This also applies to k=7, both rows of class 2 and class 3.

This demonstrates the importance of selecting suitable features depending on the data set and type of classifier used.

There appears to be no benefits of using a third feature, but this might be a specific result due to our selected features and data set.

6 Principal Component Analysis and KNN

Principal Component Analysis (PCA) is a technique used to emphasize variation and bring out strong patterns in a data set. We have used PCA to reduce the dimensionality of the data from 13 to 2 features.

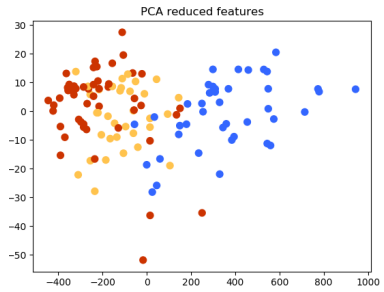

Figure 9: PCA selected features

Figure 9 shows the plot of the features reduced by PCA. In comparison with Figure 1, we observe that there is more overlapping of classes 2 (red) and 3 (yellow). In addition, the clustering of classes is quite low.

Value of k	KNN accuracy	KNN accuracy after PCA
$k = 1$	0.90	0.72
$k = 2$	0.87	0.64
$k = 3$	0.85	0.66
$k = 4$	0.87	0.72
$k = 5$	0.80	0.75
$k = 7$	0.83	0.75

The table above (ref) shows the results of using KNN classifier on the data reduced by PCA as well as the manual KNN results. It is clear that the values of manual KNN give a higher accuracy for all values of k. This is because the training data are too closely overlapping after PCA resulting in no distinct decision boundary between classes. Hence, as we continue to perform KNN considering the k nearest neighbours, choosing the majority class becomes competitive and increasingly inaccurate.

With the help of the confusion matrix, we can further analyze the results.

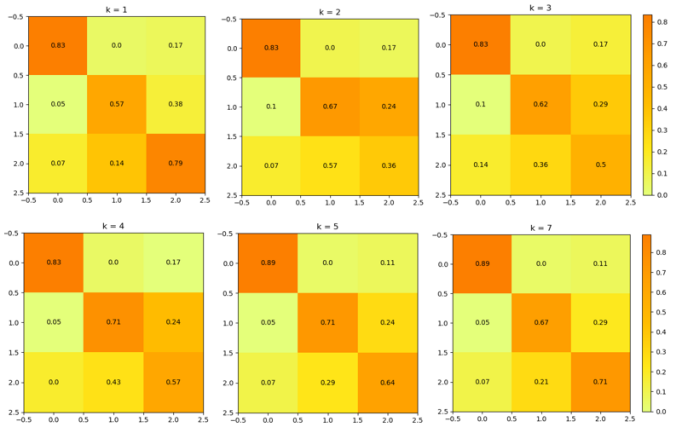

Figure 10: Confusion Matrices for KNN Classifier with k = 1, 2, 3, 4, 5, 7 after PCA

The confusion matrix (Figure 10) shows that the inaccuracies lie in the incorrect classification of classes 2 (red) and 3 (yellow). This reflects the fact that there is little separation between red and yellow data points in Figure 9 and explains why the accuracy with PCA selected features is lower than when we manually picked them. In comparison with Figure 2, where the errors were mainly between classes 1 (blue) and 2 (red), which is seen in Figure 1 where a few points of red overlap the blue cluster.

Overall, most false positives and false negatives have a relatively larger value than our manual KNN, supporting the fact that decision boundary became less prominent.

Also, PCA gives better results when dealing with high dimensionality data. In our case, the data set is quite small and thus, KNN on manually selected features works better.

7 Conclusion

To conclude, feature selection proves to be an important part of each classifier, and should be adjusted depending on how the classifier performs.