Solution 1 - Part 2 (a)

Wolfram Wiesemann

MSc Business Analytics – Machine Learning

1 Importing the Data Set

First load the data set via

```
> df = read.csv ("winequality-white.csv", sep = ";")
```

This is a clean data set without any missing values etc. You can explore that data set via:

```
> summary (df)
> dim (df)
> str (df)
> cor (df)
> df[1:4,]
> df[1:4, 1:2]
> summary (df$alcohol)
```

2 Creating a New Binary Column for Good Wines

We aim to classify wines in a binary fashion. The “quality” column is a number between 1-10. We create the new binary column as follows:

```
> isgood <- function(x) { if (x >= 6) { return (1) } else { return (0) } }
> df$good = lapply (df$quality, isgood)
> df$good = sapply (df$good, as.numeric)
```

Now we can remove the old quality column:

```
> df = subset (df, select = -c (quality))
```

We should also separate our features from the label to be predicted:

```
> X=subset (df2, select = -c (good))
> y=subset (df2, select = c (good))
```

3 Data Normalisation

We are going to normalise the data as the Nearest Neighbours classifier is sensitive to scaling. While here the training set, the validation set and the test set are available in advance, we are going to take a general approach: We normalise according to the training data and apply the same transformation to the validation set and the test set in a consistent manner.

We define a "normalize" function and apply it to the data frame:

```
> normalize = function (x) { return ((x - mean (x)) / sd (x)) }
> X_normalized = as.data.frame (lapply (X, normalize))
```

Alternatively we can use the pre-built "scale" function:

```
> X_normalized = as.data.frame (scale (X))
```

4 Splitting the Data Set

We have 4,898 data points. We are going to split the data into a training set, a validation set and a test set. Before we do so, it is always a good idea to shuffle the data to avoid having different distributions in the training/validation/test sets:

```
> df2 <- df[sample (nrow (df)),]
```

Now we can split the data set manually. We are going to use 2,000 data points for the training set, 1,500 data points for the validation set and the remaining 1,398 data points for the test set:

```
> X_train = X_normalized[1:2000,]
> X_val = X_normalized[2001:3500,]
> X_test = X_normalized[3501:4898,]
> y_train = factor (y[1:2000,])
> y_val = factor (y[2001:3500,])
> y_test = factor (y[3501:4898,])
```

5 Loading and Training the Classifier

We first need to import (and, if necessary, install) the "classification" package:

```
> install.packages (class)
> library (class)
```

We can now try a Nearest Neighbours classifier with $k = 3$:

```
> pred_val = knn (train = X_train, test = X_val, cl = y_train, 3)
> perf = sum ((pred_val == y_val)) / length (y_val)
> perf
```

We obtain a performance of 0.749 on the validation set. To evaluate the performance of the classifier on the training set, we type:

```
> pred_train = knn (train = X_train , test = X_train , cl = y_train , 3)
> perf = sum ((pred_train == y_train)) / length (y_train)
> perf
```

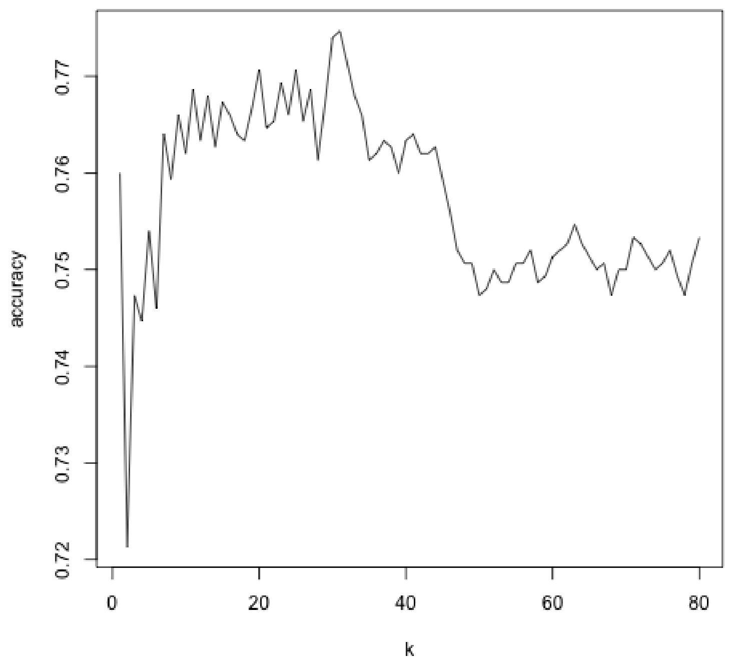
We obtain a performance of 0.8645. Note that you may obtain different numbers due to the random shuffling.

6 Selecting the Best k

We are going to use the performance on the validation set to choose the value of k that has the most promising performance on future data:

```
> out = matrix (NA, nrow = 80, ncol = 2)
> for (k in 1:80) {
+  pred_val = knn (train = X_train , test = X_val , cl = y_train , k)
+  perf = sum ((pred_val == y_val)) / length (y_val)
+  out[k,1] = k
+  out[k,2] = perf
+  }
> plot (out[, 1], out[, 2], type = "l", xlab = "k", ylab = "accuracy")
```

Our plot indicates the $k = 25$ is a reasonable number to pick:



7 Evaluating the Selected Classifier on the Test Set

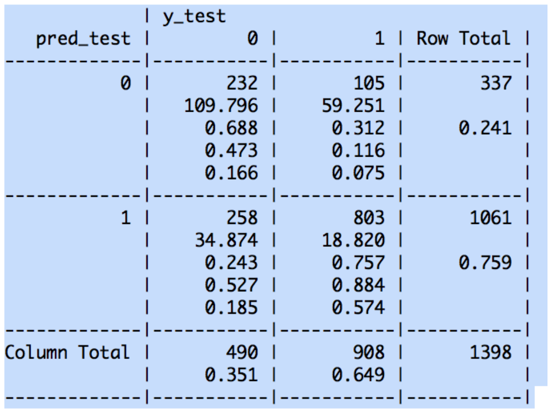
We can now evaluate the classifier with $k = 25$ on the test set:

```
> pred_test = knn (train = X_train , test = X_test , cl = y_train , 25)
> sum ((pred_test == y_test)) / length (y_test)
```

The performance on the test set is 0.74. To generate a confusion matrix, type:

```
> install.packages ("gmodels")
> library ("gmodels")
> CrossTable (pred_test , y_test)
```

In our test set of 1,398 wines, 908 have a good quality. If we classified *all* wines as good, we would have guessed right 65% of the time. By using the Nearest Neighbours classifier, we have improved to 74%:



```
             | y_test
   pred_test |         0 |         1 | Row Total |
-------------|-----------|-----------|-----------|
           0 |       232 |       105 |       337 |
             |   109.796 |    59.251 |           |
             |     0.688 |     0.312 |     0.241 |
             |     0.473 |     0.116 |           |
             |     0.166 |     0.075 |           |
-------------|-----------|-----------|-----------|
           1 |       258 |       803 |      1061 |
             |    34.874 |    18.820 |           |
             |     0.243 |     0.757 |     0.759 |
             |     0.527 |     0.884 |           |
             |     0.185 |     0.574 |           |
-------------|-----------|-----------|-----------|
Column Total |       490 |       908 |      1398 |
             |     0.351 |     0.649 |           |
-------------|-----------|-----------|-----------|
```