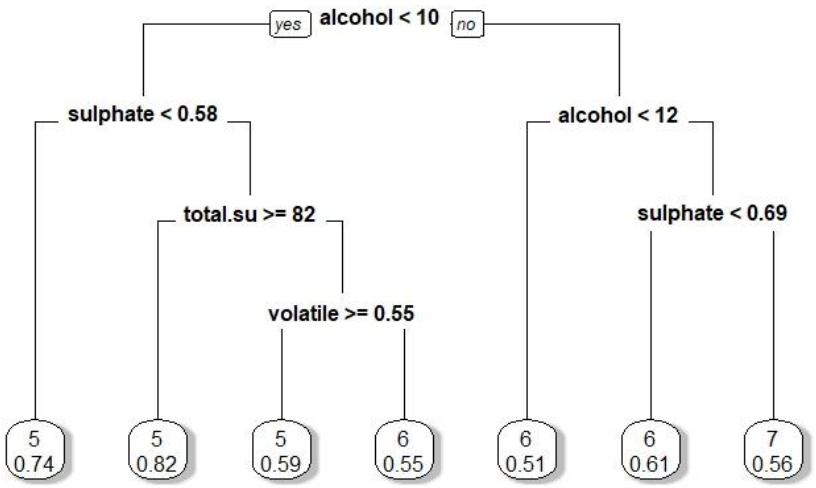
1. Decision Tree- Wine Quality

(1) What’s the decision tree result (what is the number of tree levels and how many rules you obtain)? Any explanation?



上圖為針對這筆資料所做的決策數分析，此樹如果包含最底下的節點的話總共有五個 levels，分類規則(rules)則有六個，最底下節點中的第二個數字代表在此節點的分類當中正確判別出類別的比例。在規則的解釋上，最為重要的決定規則為決策樹最上層的 alcohol 這個變數，除此之外，sulphate、total.su、volatile 都有出現在決策樹的分類規則當中，而其他沒有出現在決策樹上的變數則代表其影響分類結果的能力相較於有出現在樹上的來得小（但不代表沒影響）。

real \ predict	3	4	5	6	7	8
3	0	0	6	4	0	0
4	0	0	27	26	0	0
5	0	0	444	234	3	0
6	0	0	142	460	36	0
7	0	0	7	128	64	0
8	0	0	0	7	11	0

```
> # 計算預測準確率
> confus.matrix
> sum(diag(confus
[1] 0.6053784
```

在準確率方面，此次分類結果可以透過上方左圖的混亂矩陣運算得知準確率大約落在六成左右，由此可知，只要透過決策樹上出現的四個變數就可以在分類酒類的品質達到六成的準確度。

(2) How can you provide your suggestion for wine quality investigation via decision (IF-THEN) rule?

透過此決策樹可以來調整酒生產的參數，藉此影響酒的品質。如果希望生產品質 6 以上的酒，那麼建議一開始就把 alcohol 這個影響因子設在 10 以上（雖然 alcohol 小於 10 也還是有機會生產品質 6 的酒，但還需要其他參數配合），如果對於酒的品質要求更加嚴格，需要生產到品質 7 以上的酒，那麼除了 alcohol 要在 10 以上，sulphate 也建議要在 0.69 以上。反過來說，如果需要生產在品質 5 的酒，則要小心不要讓 alcohol 小於 10 且 sulphate 不小於 0.58 且 total.su 不大於等於 82 且 volatile 不大於等於 0.55，如此一來就會有很大的機率生產品質 5 的酒。

如果可以影響變數的成本的話，那麼就可以做更進一步且有商業價值的決策。舉例來說，如果 alcohol 這項變數為生產酒的最主要的成本（數字越大代表花費成本越高），而且品質數字越大的酒利潤越高，那麼同樣都是生產品質 6 的酒，設定 alcohol<10、sulphate>=0.58、total.su<82、volatile<0.55 也可以同樣產出品質 6 的酒，進而獲得相同利潤，那麼選擇此生產參數組合會比其他的組合還要來的好。

2. NN

(1) 調整參數

```
> ptm <- proc.time()
> bpn <- neuralnet(formula = formula.bpn,
+                  data = trainoftrain,
+                  hidden = c(2,2),
+                  learningrate = 0.01, # learning rate
+                  threshold = 0.5,    # partial derivatives of the error function, a stopping criteria
+                  stepmax = 5e7        # 最大的ieration數
+ )
Warning message:
Algorithm did not converge in 1 of 1 repetition(s) within the stepmax.
> proc.time() - ptm
   user  system elapsed
9617.31   17.44 9892.45
```

在調整參數的過程當中，我有嘗試調整 hidden、learning_rate 以及 threshold 三個參數。但在訓練過程中，時常遇到模型沒有辦法收斂的情況（如上圖），而當將 stepmax 調大時又會遇到跑不出結果的狀況。在反覆嘗試過後，我列出了此次作業有成功訓練出的模型參數，以及其所對應的運行時間(TIME)、誤差(RMSE)，如下表。

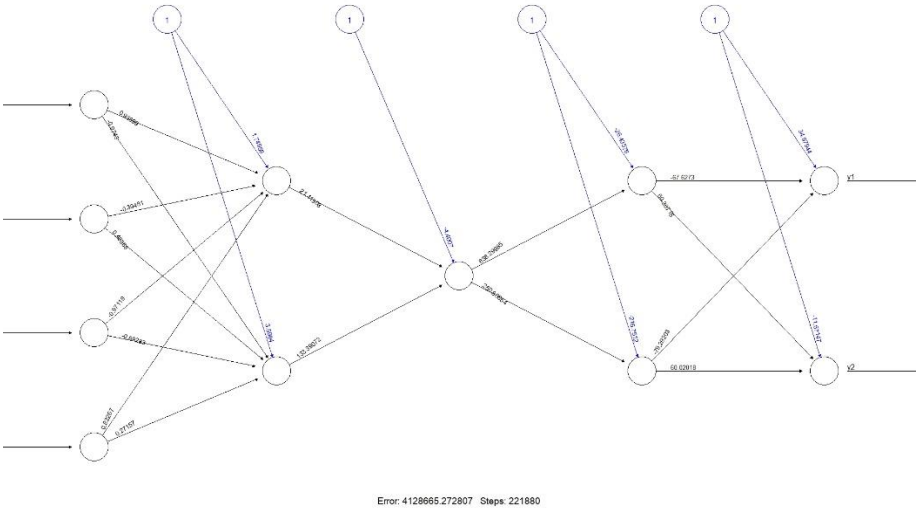
在嘗試的過程當中發現，learning_rate、threshold 這兩個參數基本上要調整至 0.1 以上才有辦法訓練出結果。在成果表現部分，基本上 stepmax 為 5.00E+06 所需的運算時間小於 5.00E+07；誤差值 RMSE 部分大約都落在 140~150 之間。

hidden	learning_rate	threshold	stepmax	TIME	RMSE_Y1	RMSE_Y2
c(2)	0.01	0.1	5.00E+06	330.34	149.5892	151.3489
c(3)	0.1	0.1	5.00E+06	282.13	147.8044	149.4774
c(4)	0.1	0.1	5.00E+06	854.4	140.7885	141.3786
c(4)	0.1	0.5	5.00E+06	40.08	140.4658	140.5019
c(1,2)	0.1	0.1	5.00E+06	516.41	150.2232	149.844
c(2,2)	0.1	0.1	5.00E+07	958.31	144.1039	143.5967
c(2,1,2)	0.5	0.5	5.00E+06	60.23	137.94	138.4739
c(2,1,2)	0.5	0.1	5.00E+06	8.76	142.0261	138.8578

而在我嘗試過的眾多參數組合當中，誤差最小的是當隱藏層有三層，各層分別有 2 個、1 個、2 個節點，learning_rate 與 threshold 皆為 0.5 時，以下為花費時間以及計算誤差的輸出結果截圖。

```
> ptm <- proc.time()
> bpn_3 <- neuralnet(formula = formula.bpn,
+                    data = trainoftrain,
+                    hidden = c(2,1,2),
+                    learningrate = 0.5, # learning rate
+                    threshold = 0.5,    # partial derivatives of the error function, a stopping criteri
a
+                    stepmax = 5e6        # 最大的ieration數
+ )
> proc.time() - ptm
   user  system elapsed
  57.14    0.25   60.23
> #計算RMSE
> sqrt( sum( (predict_value_y1 - actual_value_y1)^2 , na.rm = TRUE ) / length(actual_value_y1) )#Y1
[1] 137.94
> sqrt( sum( (predict_value_y2 - actual_value_y2)^2 , na.rm = TRUE ) / length(actual_value_y2) )#Y2
[1] 138.4739
```

而以下則是 NN 網路的輸出結果截圖。



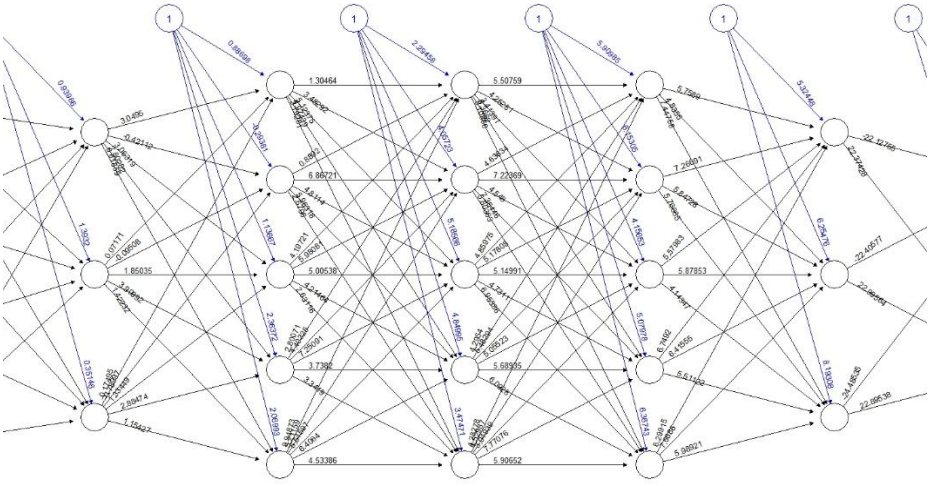
(2) 預測 y1、y2

根據上一小題的參數，我也將訓練出的模型拿來預測作業檔案後部的資料，預測結果如下表。

NO.	X1	X2	X3	X4	Y1	Y2
281	10.8	0.7	0.7	4.8	-111.94	108.6559
282	5.4	12	-1.6	12.6	-111.94	108.6559
283	-5	-12.1	21.4	-9.1	34.97944	-11.6715
284	-10.2	11.5	-4.3	0.8	-111.94	108.6559
285	-0.9	2.1	-4.8	-2.7	-111.94	108.6559
286	-1	-14.1	-19.2	0.5	-111.94	108.6559
287	-10.3	-1.1	-5.2	6.8	-111.94	108.6559
288	2.4	5.7	8.1	-6	-32.6479	48.63568
289	-25.9	-10.5	-1.5	-6.4	-111.94	108.6559
290	-2.9	12.2	11.1	9.6	-111.94	108.6559
291	11.7	13.3	-8.1	9.6	-111.94	108.6559
292	17.2	-1.1	8.6	-0.8	-111.94	108.6559
293	-8.4	-15.3	4.6	-4.2	-111.94	108.6559
294	-12.1	2	11.7	-0.3	-111.94	108.6559
295	1.3	-14.1	-4.3	7.8	-111.94	108.6559
296	-13.3	-1.7	11.7	-5.7	-111.94	108.6559
297	1.2	-7	-16.6	-1.2	-111.94	108.6559
298	-8.1	4.6	7.9	12.9	-111.94	108.6559
299	-7.3	-1	0.8	11.8	-111.94	108.6559
300	10.3	-16	3.2	2.6	-111.94	108.6559

(3) 其他發現

最後，在嘗試過程當中我也發現，當層數多於三層之後，NN 的訓練時間就會突然變得很快，而不論參數為何，輸出模型的預測結果都相同，誤差(RMSE)皆為 142.6999(RMSE_Y1)及 140.4712(RMSE_Y2)。以下以隱藏層為五層，每層節點依序為 3、5、5、5、3，threshold 及 learning_rate 皆為 0.1 時的 NN 網路為例，輸出結果截圖。




```
> bpn <- neuralnet(formula = formula.bpn,
+                  data = trainoftrain,
+                  hidden = c(3,5,5,5,3),
+                  learningrate = 0.1, # learning rate
+                  threshold = 0.1,    # partial derivatives of the error function, a stopping criteria
+                  stepmax = 5e5        # 最大的ieration數
+ )
> proc.time() - ptm
   user  system elapsed
   0.23    0.01    0.27
>
> plot(bpn)
> ############################
> pred<- compute(bpn,testoftrain[,2:5])
> predict_value_y1=pred$net.result[,1:1]
> actual_value_y1=testoftrain[,6:6]
> predict_value_y2=pred$net.result[,2:2]
> actual_value_y2=testoftrain[,7:7]
> #計算RMSE
> sqrt( sum( (predict_value_y1 - actual_value_y1)^2 , na.rm = TRUE ) / length(actual_value_y1) )#Y1
[1] 142.6999
> sqrt( sum( (predict_value_y2 - actual_value_y2)^2 , na.rm = TRUE ) / length(actual_value_y2) )#Y2
[1] 140.4712
```