

SY19 - TP3 Rapport

Chenxin LIU

2022/10/17

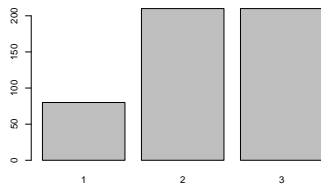
1 Classifieur

1.1 Preparation : Partitioning raw data to train & test

```
set.seed(69)
clas.test.id <- createDataPartition(TPN1_a22_clas_app$y,
                                     p = 1/5,
                                     list = TRUE)

clas.data.test <- TPN1_a22_clas_app[ clas.test.id[[1]],]
clas.data.train <- TPN1_a22_clas_app[-clas.test.id[[1]],]
clas.levels <- levels(TPN1_a22_clas_app$y)
```

1.2 Data exploration



```
## [1] 0.58
```

First we explore the data a little, there is barplot can be seen: Y consists of three classes, the number of class1 is significantly smaller than the number of class2, 3. So if we do not do machine learning and choose the class with the largest proportion each time, our error rate will be 0.58, which will be the highest error rate we can accept

1.3 Nonparametric method kNN

The first method we choose is the non-parametric one, the knn method. Firstly we apply KNN with an arbitrary $k = 10$ to have a look at general result.

```
## [1] "Contingency matrix:"
```

```
##      clas.knn.fit
##      1  2  3
##      1  0  5 11
##      2  0 23 19
##      3  2 14 26

## [1] "Error total:"

## [1] 0.51

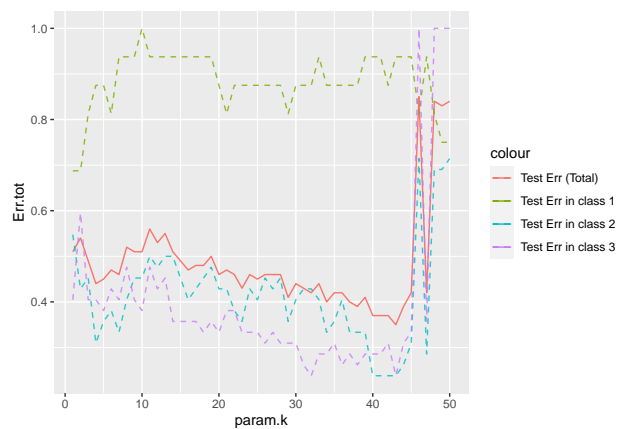
## [1] "Error within each class"

##      1      2      3
## 1.0000000 0.4523810 0.3809524
```

The error rate reaches 0.51

1.3.1 KNN with an arbitrary k

Next we try to iterate over k to see if we can optimize the error rate



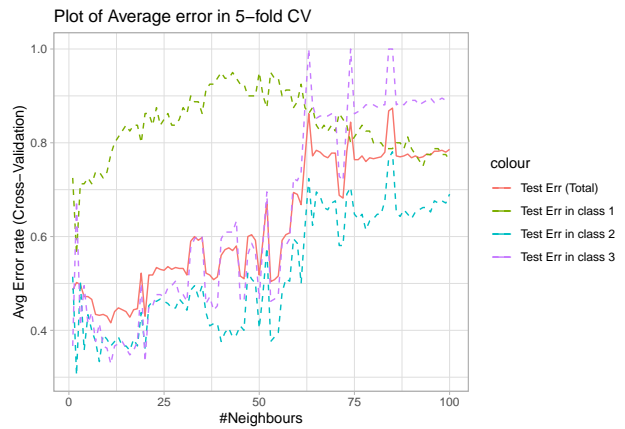
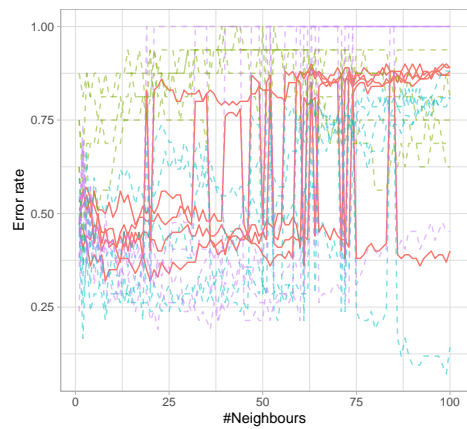
```
## [1] "Error minimal:"
```

	Err.tot	Err.1	Err.2	Err.3	param.k
43	0.35	0.9375	0.2380952	0.2380952	43

Observing the plot, we see that class1 has an error rate of 1 when the value of k exceeds 10, most likely because class1 is a smaller class and is therefore divided into other classes when the value of k increases. But at $k = 43$ we observed a minimum error rate of 0.35, which is unlikely and next we applied cross comparisons to confirm the results. `### k_f-fold validation with k=5` In the k-fold validation, firstly we choose $k = 5$

```
## [1] "result"
## [1] "best parameter k : "
```

param.k	avg.Err.tot
11	0.416

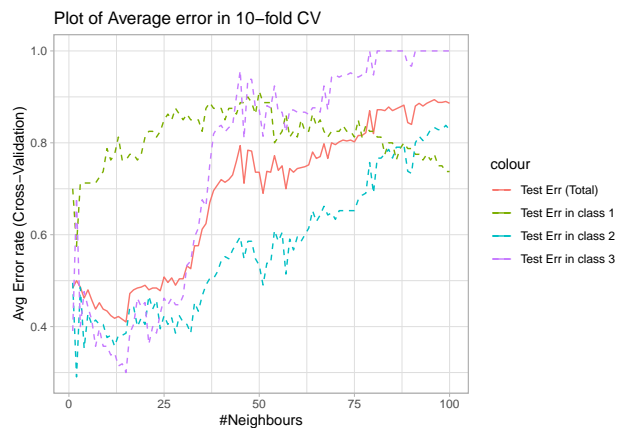
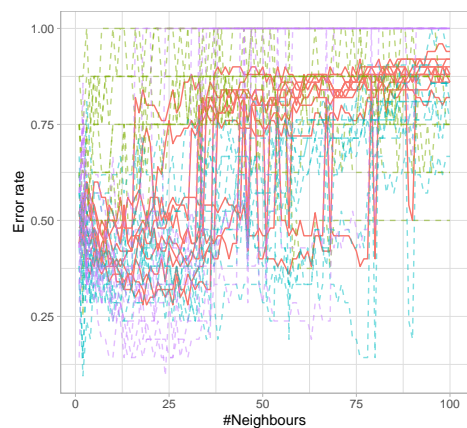


After k-fold =5, we obtained optimal results at KNN k=11 with an error rate of 0.41.

1.3.2 k_f-fold validation with k=10

```
## [1] "result"
## [1] "best parameter k : "
```

param.k	avg.Err.tot
15	0.41



After k-fold =10, we obtained optimal results at KNN k=15 with an error rate of 0.41, The results are similar

1.4 QDA

```
## $contingency.matrix
##      pred_class
## test_class 1 2 3
```

```
##          1  0  5 11
##          2  0 36  6
##          3  0  8 34
##
## $test.error.total
## [1] 0.3
##
## $test.error.within_class
##          1          2          3
## 1.0000000 0.1428571 0.1904762
```

After one QDA, we obtained an error rate of 0.38, which is already better than the optimal KNN result.

1.5 QDA with K-fold validation

1.5.1 k=5, repeats 10

```
##      Err.tot      Err.1      Err.2      Err.3
## Min.   :0.2400 Min.   :0.9375 Min.   :0.09524 Min.   :0.02381
## 1st Qu.:0.3100 1st Qu.:1.0000 1st Qu.:0.21429 1st Qu.:0.11905
## Median :0.3400 Median :1.0000 Median :0.23810 Median :0.16667
## Mean   :0.3294 Mean   :0.9962 Mean   :0.24381 Mean   :0.16095
## 3rd Qu.:0.3500 3rd Qu.:1.0000 3rd Qu.:0.28571 3rd Qu.:0.19048
## Max.   :0.4000 Max.   :1.0000 Max.   :0.38095 Max.   :0.28571
```

1.5.2 k=10, repeats 10

```
##      Err.tot      Err.1      Err.2      Err.3
## Min.   :0.2200 Min.   :0.7500 Min.   :0.09524 Min.   :0.00000
## 1st Qu.:0.2800 1st Qu.:1.0000 1st Qu.:0.17857 1st Qu.:0.09524
## Median :0.3200 Median :1.0000 Median :0.23810 Median :0.14286
## Mean   :0.3234 Mean   :0.9788 Mean   :0.22810 Mean   :0.16905
## 3rd Qu.:0.3600 3rd Qu.:1.0000 3rd Qu.:0.28571 3rd Qu.:0.23810
## Max.   :0.4600 Max.   :1.0000 Max.   :0.42857 Max.   :0.38095
```

1.6 LDA

```
## $contingency.matrix
##      pred_class
## test_class  1  2  3
##          1  7  2  7
##          2  2 23 17
##          3  5 11 26
##
## $test.error.total
## [1] 0.44
##
## $test.error.within_class
##          1          2          3
## 0.5625000 0.4523810 0.3809524
```

1.7 LDA with K-fold validation

1.7.1 k =5, repeats 10

##	Err.tot	Err.1	Err.2	Err.3
##	Min. :0.330	Min. :0.2500	Min. :0.1667	Min. :0.3571
##	1st Qu.:0.380	1st Qu.:0.4375	1st Qu.:0.2679	1st Qu.:0.4524
##	Median :0.420	Median :0.5312	Median :0.3214	Median :0.4881
##	Mean :0.429	Mean :0.5300	Mean :0.3219	Mean :0.4976
##	3rd Qu.:0.470	3rd Qu.:0.6094	3rd Qu.:0.3750	3rd Qu.:0.5476
##	Max. :0.530	Max. :0.8125	Max. :0.5000	Max. :0.7619

1.7.2 k =10, repeats 10

##	Err.tot	Err.1	Err.2	Err.3
##	Min. :0.2200	Min. :0.8750	Min. :0.09524	Min. :0.00000
##	1st Qu.:0.3000	1st Qu.:1.0000	1st Qu.:0.14286	1st Qu.:0.09524
##	Median :0.3200	Median :1.0000	Median :0.23810	Median :0.14286
##	Mean :0.3248	Mean :0.9788	Mean :0.23429	Mean :0.16619
##	3rd Qu.:0.3600	3rd Qu.:1.0000	3rd Qu.:0.28571	3rd Qu.:0.23810
##	Max. :0.4800	Max. :1.0000	Max. :0.52381	Max. :0.42857

1.8 Naive Bayes

```
## $contingency.matrix
##      pred_class
## test_class  1  2  3
##           1 10  1  5
##           2  1 33  8
##           3  4 14 24
##
## $test.error.total
## [1] 0.33
##
## $test.error.within_class
##           1           2           3
## 0.3750000 0.2142857 0.4285714
```

1.9 Naive Bayes with K-fold validation

1.9.1 k =5, repeats 10

##	Err.tot	Err.1	Err.2	Err.3
##	Min. :0.2500	Min. :0.1875	Min. :0.07143	Min. :0.2857
##	1st Qu.:0.3300	1st Qu.:0.3750	1st Qu.:0.16667	1st Qu.:0.4107
##	Median :0.3600	Median :0.4375	Median :0.21429	Median :0.4762
##	Mean :0.3588	Mean :0.4437	Mean :0.21619	Mean :0.4690
##	3rd Qu.:0.3900	3rd Qu.:0.5000	3rd Qu.:0.26190	3rd Qu.:0.5238
##	Max. :0.4400	Max. :0.6875	Max. :0.35714	Max. :0.6190

1.9.2 k=10, repeats 10

##	Err.tot	Err.1	Err.2	Err.3
## Min.	:0.2400	Min. :0.0000	Min. :0.0000	Min. :0.2381
## 1st Qu.	:0.3000	1st Qu.:0.3438	1st Qu.:0.1429	1st Qu.:0.4286
## Median	:0.3400	Median :0.3750	Median :0.1905	Median :0.4762
## Mean	:0.3526	Mean :0.4375	Mean :0.2014	Mean :0.4714
## 3rd Qu.	:0.4000	3rd Qu.:0.6250	3rd Qu.:0.2381	3rd Qu.:0.5714
## Max.	:0.5400	Max. :0.8750	Max. :0.4286	Max. :0.6667

1.10 Multinomial logistic regression

Here, our data have the classes $c > 2$, so we used the “Multinomial logistic regression” method

```
## # weights: 156 (102 variable)
## initial value 439.444915
## iter 10 value 287.826765
## iter 20 value 281.332153
## iter 30 value 272.637048
## iter 40 value 265.495269
## iter 50 value 259.341441
## iter 60 value 253.535272
## iter 70 value 246.931670
## iter 80 value 244.580896
## iter 90 value 242.577689
## iter 100 value 241.311703
## final value 241.311703
## stopped after 100 iterations
```

```
## $contingency.matrix
##      pred_class
## test_class  1  2  3
##           1  6  0 10
##           2  2 22 18
##           3  5  9 28
##
## $test.error.total
## [1] 0.44
##
## $test.error.within_class
##           1           2           3
## 0.6250000 0.4761905 0.3333333
```

1.11 Naive Bayes with K-fold validation

1.11.1 k=5, repeats 10

##	Err.tot	Err.1	Err.2	Err.3
## Min.	:0.3300	Min. :0.1875	Min. :0.1190	Min. :0.3095
## 1st Qu.	:0.4000	1st Qu.:0.4375	1st Qu.:0.2857	1st Qu.:0.4286
## Median	:0.4250	Median :0.5000	Median :0.3333	Median :0.4762
## Mean	:0.4288	Mean :0.5337	Mean :0.3424	Mean :0.4752

##	3rd Qu.:	0.4600	3rd Qu.:	0.6094	3rd Qu.:	0.4048	3rd Qu.:	0.5238
##	Max.	:0.5900	Max.	:0.9375	Max.	:0.5000	Max.	:0.5952

1.11.2 k =10, repeats 10

##	Err.tot	Err.1	Err.2	Err.3
##	Min. :0.2800	Min. :0.1250	Min. :0.04762	Min. :0.2381
##	1st Qu.:0.3800	1st Qu.:0.3750	1st Qu.:0.23810	1st Qu.:0.3810
##	Median :0.4200	Median :0.5000	Median :0.33333	Median :0.4524
##	Mean :0.4172	Mean :0.5112	Mean :0.33524	Mean :0.4633
##	3rd Qu.:0.4600	3rd Qu.:0.6250	3rd Qu.:0.39286	3rd Qu.:0.5238
##	Max. :0.5800	Max. :0.8750	Max. :0.57143	Max. :0.7619

2 Principal component analysis

