oblig2

anettfre

Autumn 2020

Problem 1

1

```
names = c("TPSA", "SAacc", "H050", "MLOGP", "RDCHI", "GATS1p", "nN", "c040", "LC50")
data = read.csv('qsar_aquatic_toxicity.csv', sep=";", col.names=names)
set.seed(1111)
smp_size = floor(0.67 * nrow(data))
train_ind = sample(seq_len(nrow(data)), replace = FALSE, size = smp_size)
train = data[train_ind, ]
test = data[-train_ind, ]
summary(data)
```

```
##       TPSA            SAacc             H050             MLOGP
##  Min.   :  0.00   Min.   :  0.00   Min.   : 0.0000   Min.   :-6.446
##  1st Qu.: 15.79   1st Qu.: 11.00   1st Qu.: 0.0000   1st Qu.: 1.228
##  Median : 40.46   Median : 42.68   Median : 0.0000   Median : 2.273
##  Mean   : 48.56   Mean   : 58.98   Mean   : 0.9395   Mean   : 2.313
##  3rd Qu.: 70.14   3rd Qu.: 77.73   3rd Qu.: 1.0000   3rd Qu.: 3.395
##  Max.   :347.32   Max.   :571.95   Max.   :18.0000   Max.   : 9.148
##      RDCHI           GATS1p            nN              c040
##  Min.   :1.000   Min.   :0.281   Min.   : 0.000   Min.   : 0.0000
##  1st Qu.:1.975   1st Qu.:0.737   1st Qu.: 0.000   1st Qu.: 0.0000
##  Median :2.344   Median :1.021   Median : 1.000   Median : 0.0000
##  Mean   :2.495   Mean   :1.047   Mean   : 1.006   Mean   : 0.3541
##  3rd Qu.:2.913   3rd Qu.:1.267   3rd Qu.: 2.000   3rd Qu.: 0.0000
##  Max.   :6.439   Max.   :2.500   Max.   :11.000   Max.   :11.0000
##       LC50
##  Min.   : 0.122
##  1st Qu.: 3.601
##  Median : 4.530
##  Mean   : 4.660
##  3rd Qu.: 5.610
##  Max.   :10.047
```

I think to model the count vaariables with a linear effect would give the best result. Looking at the count variables H050, nN and C040: In nN and C040 most samples have 0 number of atoms sincew the median is 0, in nN the median is 1 so there are some more samples with nitogen atoms than with carbon and hydrogen.

```
mod_linear_effect <- lm(LC50 ~ ., data = train)
summary(mod_linear_effect)
```

```
##
## Call:
```

```
## lm(formula = LC50 ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7964 -0.7521 -0.0807  0.6107  5.0678
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.798160   0.294150   9.513  < 2e-16 ***
## TPSA         0.028500   0.003240   8.797  < 2e-16 ***
## SAacc       -0.016799   0.002585  -6.498 2.75e-10 ***
## H050         0.042600   0.072400   0.588 0.556637
## MLOGP        0.432373   0.077150   5.604 4.19e-08 ***
## RDCHI        0.540351   0.165904   3.257 0.001234 **
## GATS1p      -0.692751   0.184495  -3.755 0.000203 ***
## nN          -0.225746   0.057818  -3.904 0.000113 ***
## c040         0.065739   0.091247   0.720 0.471716
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.205 on 356 degrees of freedom
## Multiple R-squared:  0.5165, Adjusted R-squared:  0.5056
## F-statistic: 47.54 on 8 and 356 DF,  p-value: < 2.2e-16
```

```
train_dict = train
train_dict$H050 = ifelse(train_dict$H050==0,0,1)
train_dict$nN = ifelse(train_dict$nN==0,0,1)
train_dict$c040 = ifelse(train_dict$c040==0,0,1)
test_dict = test
test_dict$H050 = ifelse(test_dict$H050==0,0,1)
test_dict$nN = ifelse(test_dict$nN==0,0,1)
test_dict$c040 = ifelse(test_dict$c040==0,0,1)

mod_dichotomize <- lm(LC50 ~ ., data = train_dict)
summary(mod_dichotomize)
```

```
##
## Call:
## lm(formula = LC50 ~ ., data = train_dict)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1577 -0.8078 -0.0958  0.6751  5.1686
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.963239   0.304821   9.721  < 2e-16 ***
## TPSA         0.023321   0.003231   7.219 3.20e-12 ***
## SAacc       -0.013842   0.002230  -6.206 1.51e-09 ***
## H050        -0.224450   0.155803  -1.441 0.150576
## MLOGP        0.422832   0.078108   5.413 1.14e-07 ***
## RDCHI        0.516205   0.168746   3.059 0.002389 **
## GATS1p      -0.715297   0.182644  -3.916 0.000108 ***
## nN          -0.030552   0.149134  -0.205 0.837795
## c040        -0.048004   0.163157  -0.294 0.768759
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.226 on 356 degrees of freedom
## Multiple R-squared:  0.4989, Adjusted R-squared:  0.4876
## F-statistic:  44.3 on 8 and 356 DF,  p-value: < 2.2e-16
```

```
error = NULL
error$train_lin <- mean((train$LC50 - predict(mod_linear_effect, newdata = train))^2)
error$test_lin <- mean((test$LC50 - predict(mod_linear_effect, newdata = test))^2)
error$train_dict <- mean((train_dict$LC50 - predict(mod_dichotomize, newdata = train_dict))^2)
error$test_dict <- mean((test_dict$LC50 - predict(mod_dichotomize, newdata = test_dict))^2)
cbind(error)
```

```
##            error
## train_lin  1.415377
## test_lin   1.462759
## train_dict 1.466875
## test_dict  1.51623
```

We can se from the test error that the methods here produce almost equal results. The metod with dicthomized variables have a little more test error but varies with each run, while the regular linear effect have the smalles train error.

Significance of the regression coefficients for the regular linear effect there is evidence that multiple variables have a correlation to LC50/ Aquatic toxicity. With TPSA, MLOGP and RDCHI having a possivtive correlation, with RDCHI having a higher P-value and therfore less evidence of correlation for RDCHI. SAacc, GATS1p and nN have a negative correlation with Aquatic toxicity.

Significance of the regression coefficients for the dichtomize linear effect there is variables that have a strong correlation to LC50/ Aquatic toxicity than the regual model. With TPSA, MLOGP and RDCHI having a possivtive correlation. RDCHI have a higher P-value and therfore less evidence of correlation for RDCHI. SAacc and GATS1p have a negative correlation with Aquatic toxicity.

Differences in p-vaiues between models: The variable nN have a low value in the linear model but not in the dichtomized model. In the dichtomized model nN have a high p-value. P-value for TPSA is lower in the linear model but still low in the dichtomized model. The p-value for H050 is higher in the linear model.

2

The sample does a random train test split each time. To be sure of different splits i set a new seed for each iteration.

```
error_train_lin = 0
error_test_lin = 0
error_train_dict = 0
error_test_dict = 0

for (i in 1:200){
  set.seed((i))
  train_ind_new = sample(seq_len(nrow(data)), replace = FALSE, size = smp_size)
  train_new = data[train_ind_new, ]
  test_new = data[-train_ind_new, ]
  mod_linear_effect_new <- lm(LC50 ~ ., data = train_new)
  train_dict_loop = train_new
  train_dict_loop$H050 = ifelse(train_dict$H050==0,0,1)
  train_dict_loop$nN = ifelse(train_dict$nN==0,0,1)
  train_dict_loop$c040 = ifelse(train_dict$c040==0,0,1)
```

```
  test_dict_loop = test_new
  test_dict_loop$H050 = ifelse(test_dict$H050==0,0,1)
  test_dict_loop$nN = ifelse(test_dict$nN==0,0,1)
  test_dict_loop$c040 = ifelse(test_dict$c040==0,0,1)

  mod_dichotomize_new = lm(LC50 ~ ., data = train_dict)
  error_train_lin = error_train_lin + mean((train_new$LC50 - predict(mod_linear_effect_new, newdata = t
  error_test_lin = error_test_lin + mean((test_new$LC50 - predict(mod_linear_effect_new, newdata = test
  error_train_dict = error_train_dict + mean((train_dict_loop$LC50 - predict(mod_dichotomize_new, newda
  error_test_dict = error_test_dict + mean((test_dict_loop$LC50 - predict(mod_dichotomize_new, newdata =
}

c((error_train_lin/200), (error_test_lin/200), (error_train_dict/200), (error_test_dict/200))
```

```
## [1] 1.421794 1.477858 1.521795 1.479954
```

The training and test error for the regual model is almost the same, they are slightly lower. The train error and test error for the dichotomized model have both increased, so there is now a grater difference between the two models. The test error is on some runs lower than the train error for the dichotomized model. Since the result of test and train error varies for both models it might sugest that 200 runs is not enough for a stable result.

When we dichotomize variables we loose information and therefore we loose some of the relation to a variable and the response variable. In our case an increase in number of atoms might have a great effect on the Aquatic toxicity, this might explane why example the variable nN, when dichotomized, lost correlation significance. Since the difference between having 1 atom in nN and 11 (that is the max value) might have a lot of inpact on Aquatic toxicity.

3

Using the step function for finding BIC, eventhough labelled AIC in the result of the model the result should be using BIC criterion. Because k is changed from its default value in the step function to log(n).

```
library(MASS)
full.model = lm(LC50 ~ ., data = as.data.frame(train))
null.model = lm(LC50 ~ 1, data = as.data.frame(train))

model.backward.aic = stepAIC(object = full.model, scope = null.model, direction = "backward")
```

```
## Start:  AIC=144.8
## LC50 ~ TPSA + SAacc + H050 + MLOGP + RDCHI + GATS1p + nN + c040
##
##          Df Sum of Sq    RSS    AIC
## - H050    1     0.502 517.12 143.15
## - c040    1     0.753 517.37 143.33
## <none>                516.61 144.80
## - RDCHI   1    15.394 532.01 153.52
## - GATS1p  1    20.460 537.07 156.98
## - nN      1    22.122 538.73 158.10
## - MLOGP   1    45.579 562.19 173.66
## - SAacc   1    61.275 577.89 183.71
## - TPSA    1   112.291 628.90 214.59
##
## Step:  AIC=143.15
## LC50 ~ TPSA + SAacc + MLOGP + RDCHI + GATS1p + nN + c040
##
```

```
##          Df Sum of Sq    RSS    AIC
## - c040    1     0.410 517.53 141.44
## <none>                517.12 143.15
## - RDCHI   1    15.300 532.41 151.80
## - nN      1    21.790 538.90 156.22
## - GATS1p  1    24.055 541.17 157.75
## - MLOGP   1    45.717 562.83 172.08
## - SAacc   1    87.914 605.03 198.46
## - TPSA    1   112.238 629.35 212.85
##
## Step:  AIC=141.44
## LC50 ~ TPSA + SAacc + MLOGP + RDCHI + GATS1p + nN
##
##          Df Sum of Sq    RSS    AIC
## <none>                517.53 141.44
## - RDCHI   1    16.463 533.99 150.87
## - nN      1    21.472 539.00 154.28
## - GATS1p  1    23.735 541.26 155.81
## - MLOGP   1    45.577 563.10 170.25
## - SAacc   1    87.664 605.19 196.56
## - TPSA    1   112.275 629.80 211.11
```

```
summary(model.backward.aic)
```

```
##
## Call:
## lm(formula = LC50 ~ TPSA + SAacc + MLOGP + RDCHI + GATS1p + nN,
##     data = as.data.frame(train))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6799 -0.7608 -0.0870  0.6365  5.0449
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.823294   0.271114  10.414  < 2e-16 ***
## TPSA         0.028257   0.003206   8.813  < 2e-16 ***
## SAacc       -0.015743   0.002022  -7.787 7.46e-14 ***
## MLOGP        0.421637   0.075092   5.615 3.95e-08 ***
## RDCHI        0.552116   0.163606   3.375 0.000820 ***
## GATS1p      -0.715925   0.176682  -4.052 6.23e-05 ***
## nN          -0.215566   0.055933  -3.854 0.000138 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.202 on 358 degrees of freedom
## Multiple R-squared:  0.5156, Adjusted R-squared:  0.5075
## F-statistic: 63.52 on 6 and 358 DF,  p-value: < 2.2e-16
```

```
model.forward.aic = stepAIC(object = null.model, scope=list(lower=null.model, upper=full.model), direct
```

```
## Start:  AIC=394.04
## LC50 ~ 1
##
##          Df Sum of Sq     RSS    AIC
```

```
## + MLOGP   1    319.34  749.12 266.44
## + RDCHI   1    153.46  915.00 339.45
## + GATS1p  1    122.53  945.93 351.58
## + H050    1     51.60 1016.87 377.97
## + SAacc   1     18.43 1050.04 389.69
## + nN      1      9.90 1058.56 392.64
## <none>                1068.46 394.04
## + c040    1      2.31 1066.16 395.25
## + TPSA    1      0.02 1068.44 396.03
##
## Step:  AIC=266.44
## LC50 ~ MLOGP
##
##          Df Sum of Sq    RSS    AIC
## + TPSA    1   112.511 636.61 209.03
## + RDCHI   1    56.769 692.35 239.67
## + GATS1p  1    19.344 729.77 258.89
## + SAacc   1    18.165 730.95 259.48
## + c040    1    12.074 737.05 262.50
## + nN      1     9.462 739.66 263.80
## + H050    1     8.176 740.94 264.43
## <none>                749.12 266.44
##
## Step:  AIC=209.03
## LC50 ~ MLOGP + TPSA
##
##          Df Sum of Sq    RSS    AIC
## + SAacc   1    71.489 565.12 167.56
## + GATS1p  1    18.753 617.86 200.12
## + H050    1    16.878 619.73 201.23
## + nN      1    11.109 625.50 204.61
## <none>                636.61 209.03
## + RDCHI   1     3.385 633.22 209.09
## + c040    1     1.644 634.96 210.09
##
## Step:  AIC=167.56
## LC50 ~ MLOGP + TPSA + SAacc
##
##          Df Sum of Sq    RSS    AIC
## + nN      1   17.8642 547.26 157.83
## + GATS1p  1   12.0585 553.06 161.68
## + RDCHI   1    5.0064 560.11 166.31
## <none>                565.12 167.56
## + c040    1    0.1909 564.93 169.43
## + H050    1    0.1864 564.93 169.44
##
## Step:  AIC=157.83
## LC50 ~ MLOGP + TPSA + SAacc + nN
##
##          Df Sum of Sq    RSS    AIC
## + GATS1p  1   13.2671 533.99 150.87
## + RDCHI   1    5.9948 541.26 155.81
## <none>                547.26 157.83
## + H050    1    1.5711 545.68 158.78
```

```
## + c040    1    0.5781 546.68 159.45
##
## Step:  AIC=150.87
## LC50 ~ MLOGP + TPSA + SAacc + nN + GATS1p
##
##         Df Sum of Sq    RSS    AIC
## + RDCHI  1   16.4631 517.53 141.44
## <none>               533.99 150.87
## + c040   1    1.5736 532.41 151.80
## + H050   1    0.0071 533.98 152.87
##
## Step:  AIC=141.44
## LC50 ~ MLOGP + TPSA + SAacc + nN + GATS1p + RDCHI
##
##        Df Sum of Sq    RSS    AIC
## <none>              517.53 141.44
## + c040  1   0.41029 517.12 143.15
## + H050  1   0.15947 517.37 143.33
```

```
summary(model.forward.aic)
```

```
##
## Call:
## lm(formula = LC50 ~ MLOGP + TPSA + SAacc + nN + GATS1p + RDCHI,
##     data = as.data.frame(train))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6799 -0.7608 -0.0870  0.6365  5.0449
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.823294   0.271114  10.414  < 2e-16 ***
## MLOGP        0.421637   0.075092   5.615 3.95e-08 ***
## TPSA         0.028257   0.003206   8.813  < 2e-16 ***
## SAacc       -0.015743   0.002022  -7.787 7.46e-14 ***
## nN          -0.215566   0.055933  -3.854 0.000138 ***
## GATS1p      -0.715925   0.176682  -4.052 6.23e-05 ***
## RDCHI        0.552116   0.163606   3.375 0.000820 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.202 on 358 degrees of freedom
## Multiple R-squared:  0.5156, Adjusted R-squared:  0.5075
## F-statistic: 63.52 on 6 and 358 DF,  p-value: < 2.2e-16
```

```
mean(model.backward.aic$residuals^2)
```

```
## [1] 1.417878
```

```
mean(model.forward.aic$residuals^2)
```

```
## [1] 1.417878
```

```
#step function with k = log(n)
model.backward.bic = step(object = full.model, scope = null.model, direction="backward", k=log(nrow(tra
```

```
## Start:  AIC=179.9
## LC50 ~ TPSA + SAacc + H050 + MLOGP + RDCHI + GATS1p + nN + c040
##
##          Df Sum of Sq    RSS    AIC
## - H050    1     0.502 517.12 174.35
## - c040    1     0.753 517.37 174.53
## <none>                516.61 179.90
## - RDCHI   1    15.394 532.01 184.72
## - GATS1p  1    20.460 537.07 188.18
## - nN      1    22.122 538.73 189.30
## - MLOGP   1    45.579 562.19 204.86
## - SAacc   1    61.275 577.89 214.91
## - TPSA    1   112.291 628.90 245.79
##
## Step:  AIC=174.35
## LC50 ~ TPSA + SAacc + MLOGP + RDCHI + GATS1p + nN + c040
##
##          Df Sum of Sq    RSS    AIC
## - c040    1     0.410 517.53 168.74
## <none>                517.12 174.35
## - RDCHI   1    15.300 532.41 179.10
## - nN      1    21.790 538.90 183.52
## - GATS1p  1    24.055 541.17 185.05
## - MLOGP   1    45.717 562.83 199.38
## - SAacc   1    87.914 605.03 225.76
## - TPSA    1   112.238 629.35 240.15
##
## Step:  AIC=168.74
## LC50 ~ TPSA + SAacc + MLOGP + RDCHI + GATS1p + nN
##
##          Df Sum of Sq    RSS    AIC
## <none>                517.53 168.74
## - RDCHI   1    16.463 533.99 174.27
## - nN      1    21.472 539.00 177.68
## - GATS1p  1    23.735 541.26 179.21
## - MLOGP   1    45.577 563.10 193.65
## - SAacc   1    87.664 605.19 219.96
## - TPSA    1   112.275 629.80 234.51
```

```
model.forward.bic = step(object = null.model, scope=list(lower=null.model, upper=full.model), direction=
```

```
## Start:  AIC=397.94
## LC50 ~ 1
##
##          Df Sum of Sq     RSS    AIC
## + MLOGP   1    319.34  749.12 274.24
## + RDCHI   1    153.46  915.00 347.24
## + GATS1p  1    122.53  945.93 359.38
## + H050    1     51.60 1016.87 385.77
## + SAacc   1     18.43 1050.04 397.49
## <none>                1068.46 397.94
## + nN      1      9.90 1058.56 400.44
## + c040    1      2.31 1066.16 403.05
## + TPSA    1      0.02 1068.44 403.83
##
```

```
## Step:  AIC=274.24
## LC50 ~ MLOGP
##
##          Df Sum of Sq    RSS    AIC
## + TPSA    1   112.511 636.61 220.73
## + RDCHI   1    56.769 692.35 251.37
## + GATS1p  1    19.344 729.77 270.59
## + SAacc   1    18.165 730.95 271.18
## + c040    1    12.074 737.05 274.20
## <none>                749.12 274.24
## + nN      1     9.462 739.66 275.50
## + H050    1     8.176 740.94 276.13
##
## Step:  AIC=220.73
## LC50 ~ MLOGP + TPSA
##
##          Df Sum of Sq    RSS    AIC
## + SAacc   1    71.489 565.12 183.16
## + GATS1p  1    18.753 617.86 215.72
## + H050    1    16.878 619.73 216.83
## + nN      1    11.109 625.50 220.21
## <none>                636.61 220.73
## + RDCHI   1     3.385 633.22 224.69
## + c040    1     1.644 634.96 225.69
##
## Step:  AIC=183.16
## LC50 ~ MLOGP + TPSA + SAacc
##
##          Df Sum of Sq    RSS    AIC
## + nN      1   17.8642 547.26 177.33
## + GATS1p  1   12.0585 553.06 181.18
## <none>                565.12 183.16
## + RDCHI   1    5.0064 560.11 185.81
## + c040    1    0.1909 564.93 188.93
## + H050    1    0.1864 564.93 188.94
##
## Step:  AIC=177.33
## LC50 ~ MLOGP + TPSA + SAacc + nN
##
##          Df Sum of Sq    RSS    AIC
## + GATS1p  1   13.2671 533.99 174.27
## <none>                547.26 177.33
## + RDCHI   1    5.9948 541.26 179.21
## + H050    1    1.5711 545.68 182.18
## + c040    1    0.5781 546.68 182.84
##
## Step:  AIC=174.27
## LC50 ~ MLOGP + TPSA + SAacc + nN + GATS1p
##
##         Df Sum of Sq    RSS    AIC
## + RDCHI  1   16.4631 517.53 168.74
## <none>               533.99 174.27
## + c040   1    1.5736 532.41 179.10
## + H050   1    0.0071 533.98 180.17
```

```
##
## Step:  AIC=168.74
## LC50 ~ MLOGP + TPSA + SAacc + nN + GATS1p + RDCHI
##
##        Df Sum of Sq    RSS    AIC
## <none>              517.53 168.74
## + c040  1   0.41029 517.12 174.35
## + H050  1   0.15947 517.37 174.53
```

```
summary(model.backward.aic)
```

```
##
## Call:
## lm(formula = LC50 ~ TPSA + SAacc + MLOGP + RDCHI + GATS1p + nN,
##     data = as.data.frame(train))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6799 -0.7608 -0.0870  0.6365  5.0449
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.823294   0.271114  10.414  < 2e-16 ***
## TPSA         0.028257   0.003206   8.813  < 2e-16 ***
## SAacc       -0.015743   0.002022  -7.787 7.46e-14 ***
## MLOGP        0.421637   0.075092   5.615 3.95e-08 ***
## RDCHI        0.552116   0.163606   3.375 0.000820 ***
## GATS1p      -0.715925   0.176682  -4.052 6.23e-05 ***
## nN          -0.215566   0.055933  -3.854 0.000138 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.202 on 358 degrees of freedom
## Multiple R-squared:  0.5156, Adjusted R-squared:  0.5075
## F-statistic: 63.52 on 6 and 358 DF,  p-value: < 2.2e-16
```

```
summary(model.forward.aic)
```

```
##
## Call:
## lm(formula = LC50 ~ MLOGP + TPSA + SAacc + nN + GATS1p + RDCHI,
##     data = as.data.frame(train))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6799 -0.7608 -0.0870  0.6365  5.0449
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.823294   0.271114  10.414  < 2e-16 ***
## MLOGP        0.421637   0.075092   5.615 3.95e-08 ***
## TPSA         0.028257   0.003206   8.813  < 2e-16 ***
## SAacc       -0.015743   0.002022  -7.787 7.46e-14 ***
## nN          -0.215566   0.055933  -3.854 0.000138 ***
## GATS1p      -0.715925   0.176682  -4.052 6.23e-05 ***
```

```
## RDCHI        0.552116   0.163606   3.375 0.000820 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.202 on 358 degrees of freedom
## Multiple R-squared:  0.5156, Adjusted R-squared:  0.5075
## F-statistic: 63.52 on 6 and 358 DF,  p-value: < 2.2e-16
```

```
summary(model.backward.bic)
```

```
##
## Call:
## lm(formula = LC50 ~ TPSA + SAacc + MLOGP + RDCHI + GATS1p + nN,
##     data = as.data.frame(train))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6799 -0.7608 -0.0870  0.6365  5.0449
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.823294   0.271114  10.414  < 2e-16 ***
## TPSA         0.028257   0.003206   8.813  < 2e-16 ***
## SAacc       -0.015743   0.002022  -7.787 7.46e-14 ***
## MLOGP        0.421637   0.075092   5.615 3.95e-08 ***
## RDCHI        0.552116   0.163606   3.375 0.000820 ***
## GATS1p      -0.715925   0.176682  -4.052 6.23e-05 ***
## nN          -0.215566   0.055933  -3.854 0.000138 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.202 on 358 degrees of freedom
## Multiple R-squared:  0.5156, Adjusted R-squared:  0.5075
## F-statistic: 63.52 on 6 and 358 DF,  p-value: < 2.2e-16
```

```
summary(model.forward.bic)
```

```
##
## Call:
## lm(formula = LC50 ~ MLOGP + TPSA + SAacc + nN + GATS1p + RDCHI,
##     data = as.data.frame(train))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6799 -0.7608 -0.0870  0.6365  5.0449
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.823294   0.271114  10.414  < 2e-16 ***
## MLOGP        0.421637   0.075092   5.615 3.95e-08 ***
## TPSA         0.028257   0.003206   8.813  < 2e-16 ***
## SAacc       -0.015743   0.002022  -7.787 7.46e-14 ***
## nN          -0.215566   0.055933  -3.854 0.000138 ***
## GATS1p      -0.715925   0.176682  -4.052 6.23e-05 ***
## RDCHI        0.552116   0.163606   3.375 0.000820 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.202 on 358 degrees of freedom
## Multiple R-squared:  0.5156, Adjusted R-squared:  0.5075
## F-statistic: 63.52 on 6 and 358 DF,  p-value: < 2.2e-16
```

For backward elimination with AIC criterion I get the model lm(LC50 ~ TPSA + SAacc + MLOGP + RDCHI + GATS1p + nN). For forward elimination with AIC criterion I get the model lm(LC50 ~ MLOGP + TPSA + SAacc + nN + GATS1p + RDCHI). For backward elimination with BIC criterion I get the model lm(LC50 ~ TPSA + SAacc + MLOGP + RDCHI + GATS1p + nN). This is the same as with backward elimination with AIC criterion. For forward elimination with BIC criterion I get the model lm(LC50 ~ MLOGP + TPSA + SAacc + nN + GATS1p + RDCHI). This is the same as with forward elimination with AIC criterion. These models are not the same when using forward and backward elimination. This happens here because when we are using different methods for finding the models with forward and backward elimination. In forward elimination we are adding one by one variable that reduces the residual sum of squares until we have reached the stopping criterion. In backward elimination we are removing the variable with the highest p-value, so the variable that have least correlation with the Aquatic toxicity, then using the new model finding the next variable to drop (that now has the highest p-value), until we reach the stopping criterion.

4

Using 10-fold cross-validation. Using MSE as a measure to find the best complexity parameter.

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 3.0-2
```

```
library(leaps)

B = 20
l_grid = 10**seq(-5, -1, length=20)
X = scale(as.matrix(train[, -9]))
y =  scale(as.matrix(train[, 9]))
n = nrow(X)
bootstrap.mse = matrix(NA, nrow = 20, ncol = 1)

for (b in 1:B) {
  mse = 0
  for (j in 1:100) {
    index = sample(1:n, size=n, replace=TRUE)
    mod.ridge = glmnet(X[index, ],y[index,], lambda = l_grid[b], alpha = 0)
    pred = predict(mod.ridge, newx = X[-index,-9])
    mse = mse + mean((pred - y[-index,])^2)
  }
  bootstrap.mse[b] = mse/100
}
#number 17 is lowest
show(l_grid[13])
```

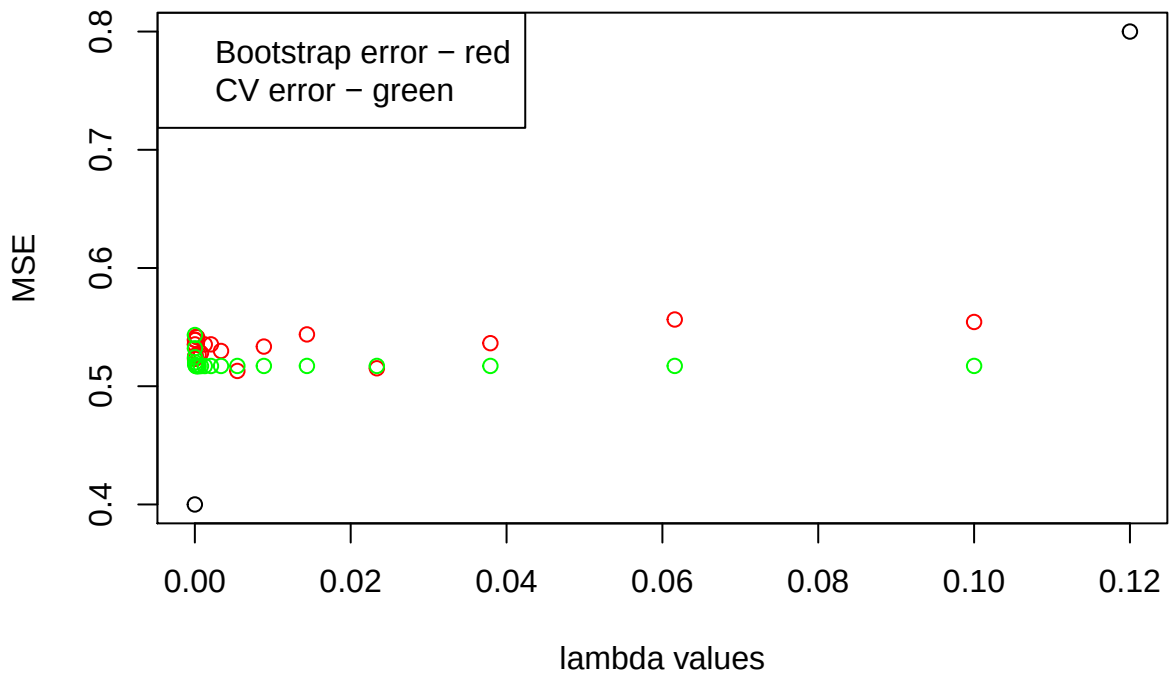
```
## [1] 0.003359818
```

```
#10 fold cross validation
cv.ridge = cv.glmnet(X, y, lambda = l_grid, alpha=0, type.measure="mse")
```

```r
show(cv.ridge$lambda.min)
```

```
## [1] 0.003359818
```

```r
plot(c(0,0.12),c(0.4,0.8), xlab="lambda values", ylab="MSE")
points(l_grid, bootstrap.mse, col="red")
points(l_grid, cv.ridge$cvm, col="green")
legend("topleft",c("Bootstrap error - red", "CV error - green"), col=c("red","green"))
```



The best lambda from bootstrapping is 0.0034 and from cross-validation 0.0144, this varies some for different runs. We can see from the plot that bootstrap gives a higher MSE some values of lambda. We also see that CV generaly have less variation for different lambdas than bootstrap.

5

First I am smootging TPSA, MLOGP, RDCHI, SAacc and GATS1p since these are the values from the earlier task with highest correlation. They I am smoothing the other values, H050, nN and c040 and adding 6 degrees of freedom for a more complex model.

```r
# gam spline
library(gam)
```

```
## Loading required package: splines
## Loading required package: foreach
## Loaded gam 1.16.1
```

```
mod_gam1 = gam(LC50 ~ s(TPSA) + s(SAacc) + H050 + s(MLOGP) + s(RDCHI) + s(GATS1p) + nN + c040, data = t
```

```
## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument
## ignored
```

```
summary(mod_gam1)
```

```
##
## Call: gam(formula = LC50 ~ s(TPSA) + s(SAacc) + H050 + s(MLOGP) + s(RDCHI) +
##     s(GATS1p) + nN + c040, data = train)
## Deviance Residuals:
##     Min      1Q  Median      3Q     Max
## -2.5660 -0.7410 -0.0621  0.5725  4.8614
##
## (Dispersion Parameter for gaussian family taken to be 1.3575)
##
##     Null Deviance: 1068.462 on 364 degrees of freedom
## Residual Deviance: 462.9011 on 340.9998 degrees of freedom
## AIC: 1172.555
##
## Number of Local Scoring Iterations: 4
##
## Anova for Parametric Effects
##             Df Sum Sq Mean Sq  F value    Pr(>F)
## s(TPSA)      1   0.06    0.06   0.0423 0.8372129
## s(SAacc)     1  71.78   71.78  52.8747 2.452e-12 ***
## H050         1  70.07   70.07  51.6204 4.266e-12 ***
## s(MLOGP)     1 324.00  324.00 238.6786 < 2.2e-16 ***
## s(RDCHI)     1   4.79    4.79   3.5297 0.0611321 .
## s(GATS1p)    1  20.29   20.29  14.9466 0.0001324 ***
## nN           1  13.69   13.69  10.0819 0.0016338 **
## c040         1   2.04    2.04   1.4993 0.2216238
## Residuals 341 462.90    1.36
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##             Npar Df Npar F     Pr(F)
## (Intercept)
## s(TPSA)            3 1.4072  0.240470
## s(SAacc)           3 7.4490 7.615e-05 ***
## H050
## s(MLOGP)           3 2.0953  0.100599
## s(RDCHI)           3 4.6676  0.003276 **
## s(GATS1p)          3 0.8113  0.488315
## nN
## c040
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
mod_gam2 = gam(LC50 ~ TPSA + SAacc + s(H050, df=6) + MLOGP + RDCHI + GATS1p + s(nN, df=6) + s(c040, df=
```

```
## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument
## ignored
```

```r
summary(mod_gam2)
```

```
##
## Call: gam(formula = LC50 ~ TPSA + SAacc + s(H050, df = 6) + MLOGP +
##     RDCHI + GATS1p + s(nN, df = 6) + s(c040, df = 6), data = train)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.92759 -0.75344 -0.02662  0.57482  5.14638
##
## (Dispersion Parameter for gaussian family taken to be 1.3879)
##
##     Null Deviance: 1068.462 on 364 degrees of freedom
## Residual Deviance: 474.6516 on 342.0002 degrees of freedom
## AIC: 1179.704
##
## Number of Local Scoring Iterations: 12
##
## Anova for Parametric Effects
##                   Df Sum Sq Mean Sq  F value    Pr(>F)
## TPSA               1   0.13    0.13   0.0972 0.7554180
## SAacc              1  75.07   75.07  54.0877 1.430e-12 ***
## s(H050, df = 6)    1  30.73   30.73  22.1398 3.686e-06 ***
## MLOGP              1 388.09  388.09 279.6321 < 2.2e-16 ***
## RDCHI              1   3.54    3.54   2.5521 0.1110726
## GATS1p             1  19.51   19.51  14.0595 0.0002079 ***
## s(nN, df = 6)      1  22.50   22.50  16.2120 6.979e-05 ***
## s(c040, df = 6)    1   6.95    6.95   5.0051 0.0259172 *
## Residuals        342 474.65    1.39
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                  Npar Df Npar F    Pr(F)
## (Intercept)
## TPSA
## SAacc
## s(H050, df = 6)       5 2.9248 0.013356 *
## MLOGP
## RDCHI
## GATS1p
## s(nN, df = 6)         5 3.9221 0.001801 **
## s(c040, df = 6)       4 2.1527 0.074020 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

My first smoothing model have a median residuals of -0.1132. It has much less significant values for nonparametric effects with s(SAacc) as the most influencal one.

My second smoothing model have a median residuals of -0.1108, this is less than model 1. This model have more significant nonparametric effects than the first, but only the variables with a higher degree of freedom. For the parametric effect the models have more similar p-values.

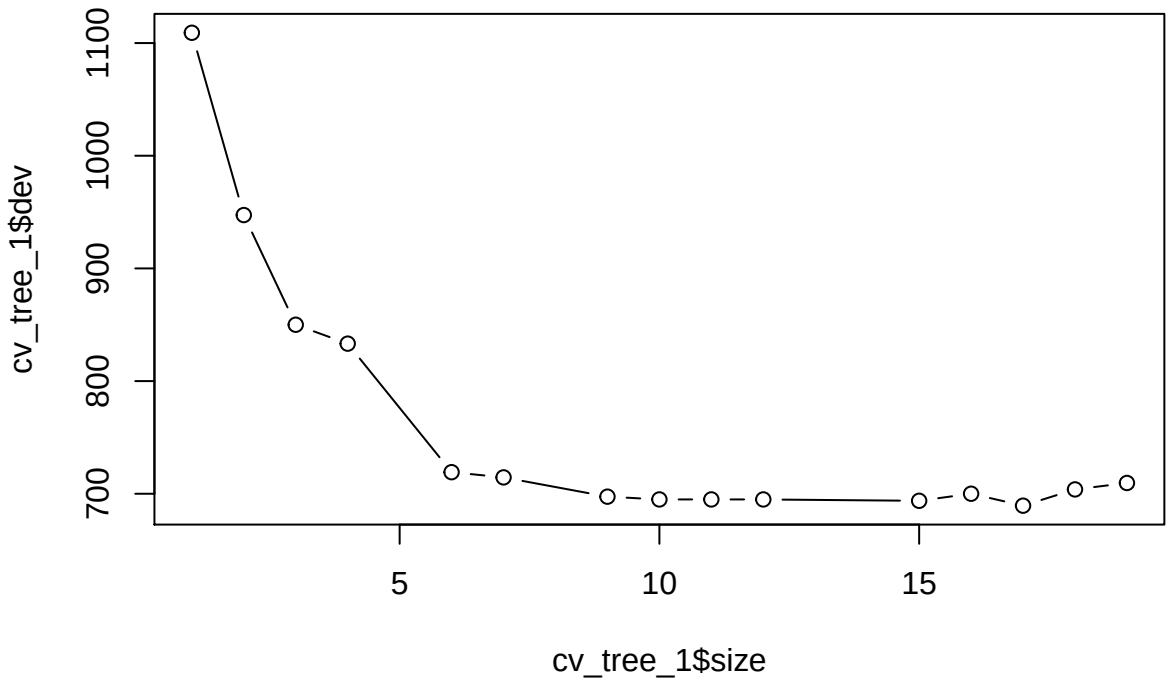
6

I use library tree for creating the tree and cv.tree to select the best number of nodes on the tree. I then use prune.tree for pruning.

```
library(tree)
tree_1 = tree(LC50~., train)
summary(tree_1)
```

```
##
## Regression tree:
## tree(formula = LC50 ~ ., data = train)
## Variables actually used in tree construction:
## [1] "MLOGP"  "TPSA"   "nN"     "H050"   "SAacc"  "GATS1p" "c040"
## Number of terminal nodes:  19
## Residual mean deviance:  0.9794 = 338.9 / 346
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -3.24900 -0.57410 -0.03792  0.00000  0.55030  3.79600
```

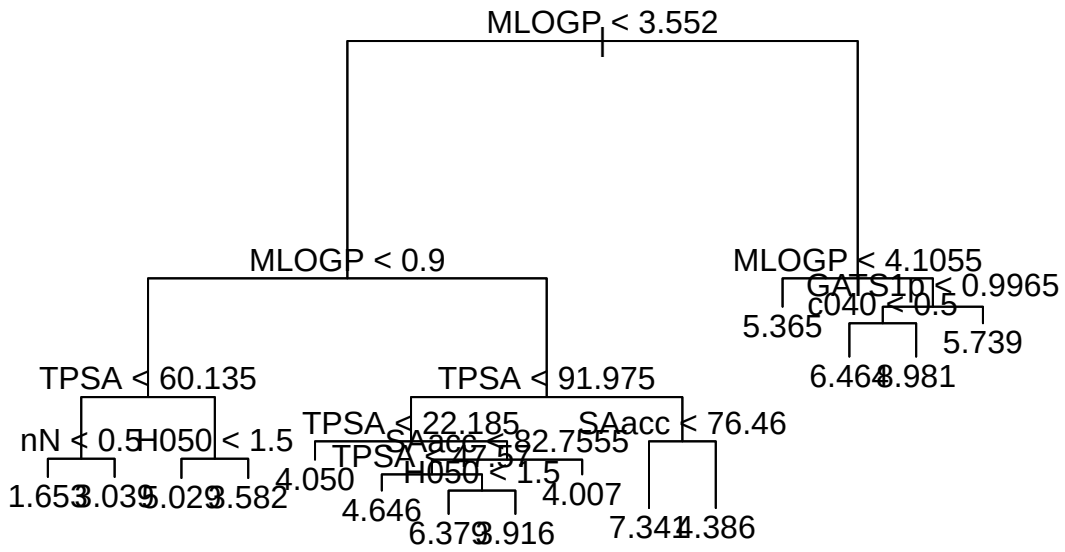
```
#plot(tree_1)
#text(tree_1, pretty = 0)

cv_tree_1 = cv.tree(tree_1)
plot(cv_tree_1$size, cv_tree_1$dev, type = 'b')
```




```
summary(cv_tree_1)
```

```
##        Length Class  Mode
## size   15     -none- numeric
## dev    15     -none- numeric
## k      15     -none- numeric
## method  1     -none- character
```

```
prune_tree_1 = prune.tree(tree_1, best = 13)
plot(prune_tree_1)
text(prune_tree_1, pretty = 0)
```




```
mean((train$LC50 - predict(prune_tree_1, newdata = train))^2)
```

```
## [1] 1.084766
```

The cost-complexity pruning led to the selected tree size of 13, using the function cv.tree that uses cross-validation to find the deviance. I can also see on the plot that a tree with size 13 have a low deviance.
Se that a 13 node tree is the best from cv.tree, using plot I visualize the results. Using pruning i get a train mse of 1.24.

7

```
train_error = NULL
test_error = NULL

test_error$lin = error_test_lin/200
test_error$dict = error_test_dict/200
test_error$gam1 = mean((test$LC50 - predict(mod_gam1, newdata = test[, -9]))^2)
test_error$gam2 = mean((test$LC50 - predict(mod_gam2, newdata = test[, -9]))^2)
```

```
test_error$tree = mean((test$LC50 - predict(prune_tree_1, newdata = test))^2)


train_error$lin = error_train_lin/200
train_error$dict = error_train_dict/200
train_error$gam1 = mean((train$LC50 - predict(mod_gam1, newdata = train[, -9]))^2)
train_error$gam2 = mean((train$LC50 - predict(mod_gam2, newdata = train[, -9]))^2)
train_error$tree = mean((train$LC50 - predict(prune_tree_1, newdata = train))^2)

cbind(error)
```

```
##            error
## train_lin  1.415377
## test_lin   1.462759
## train_dict 1.466875
## test_dict  1.51623
```

```
cbind(train_error, test_error)
```

```
##      train_error test_error
## lin  1.421794    1.477858
## dict 1.521795    1.479954
## gam1 1.268208    1.470066
## gam2 1.300419    1.540866
## tree 1.084766    1.682413
```

The lowest test error is when using the first gam model. The best train error is also from that model. The linear effect test error is slightlty higher than the gam model.

Problem 2

1

Using sample.split to ensure about equal amount of women with diabeties.

```
library(mlbench)
library(caTools)
library(class)
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
## Registered S3 method overwritten by 'cli':
##   method     from
##   print.tree tree
```

```
data(PimaIndiansDiabetes)
train_ind_dia = sample.split(PimaIndiansDiabetes$diabetes, SplitRatio = 0.67)
test_ind_dia = !train_ind_dia
print(dim(PimaIndiansDiabetes[train_ind_dia,-9]))
```

```
## [1] 515   8
```

```
trControl_5 = trainControl(method  = "cv",  number  = 5)
trControl_loo = trainControl(method  = "cv",  number  = nrow(PimaIndiansDiabetes[train_ind_dia,]))
fit_5 = train(x = PimaIndiansDiabetes[train_ind_dia,-9],
              y = PimaIndiansDiabetes[train_ind_dia,9],
```

```
              method     = "knn",
              tuneGrid   = expand.grid(k = 1:10),
              trControl  = trControl_5,
              metric     = "Accuracy")
fit_loo = train(x = PimaIndiansDiabetes[train_ind_dia,-9],
                y = PimaIndiansDiabetes[train_ind_dia,9],
                method     = "knn",
                tuneGrid   = expand.grid(k = 1:10),
                trControl  = trControl_loo,
                metric     = "Accuracy")
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```
summary(fit_5)
```

```
##             Length Class      Mode
## learn       2      -none-     list
## k           1      -none-     numeric
## theDots     0      -none-     list
## xNames      8      -none-     character
## problemType 1      -none-     character
## tuneValue   1      data.frame list
## obsLevels   2      -none-     character
## param       0      -none-     list
```

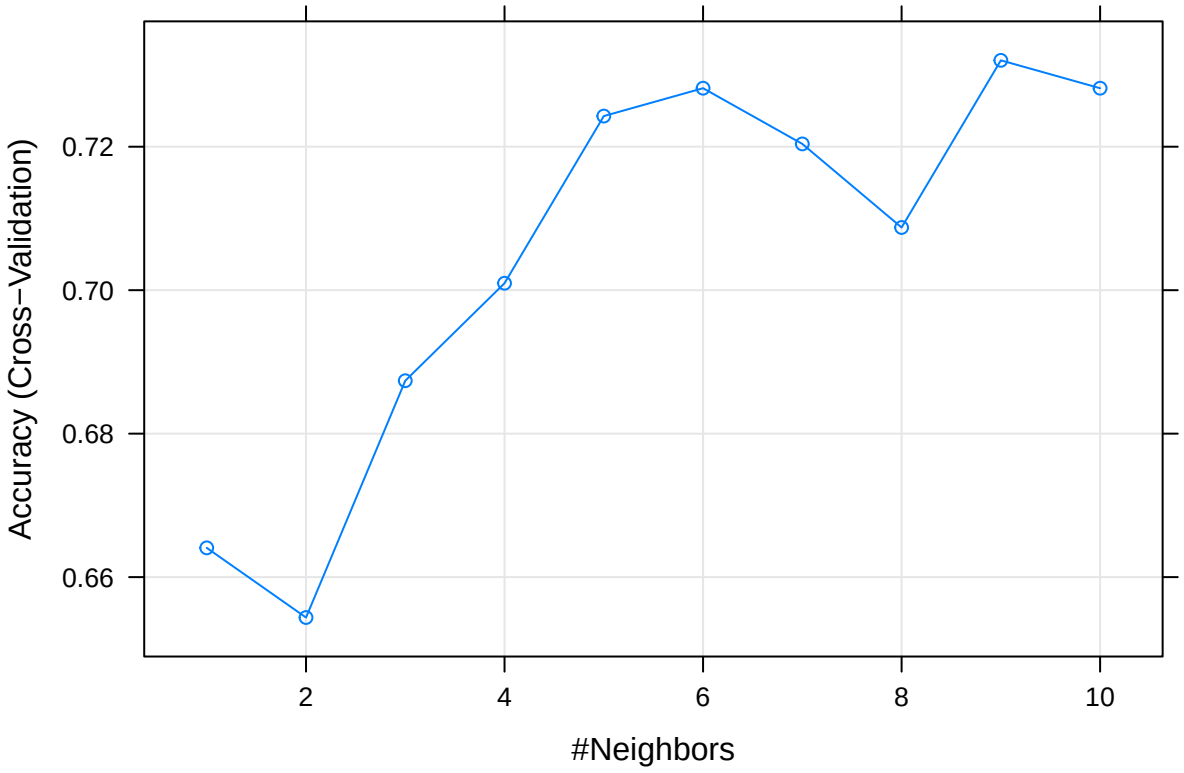
```
summary(fit_loo)
```

```
##             Length Class      Mode
## learn       2      -none-     list
## k           1      -none-     numeric
## theDots     0      -none-     list
## xNames      8      -none-     character
## problemType 1      -none-     character
## tuneValue   1      data.frame list
## obsLevels   2      -none-     character
## param       0      -none-     list
```

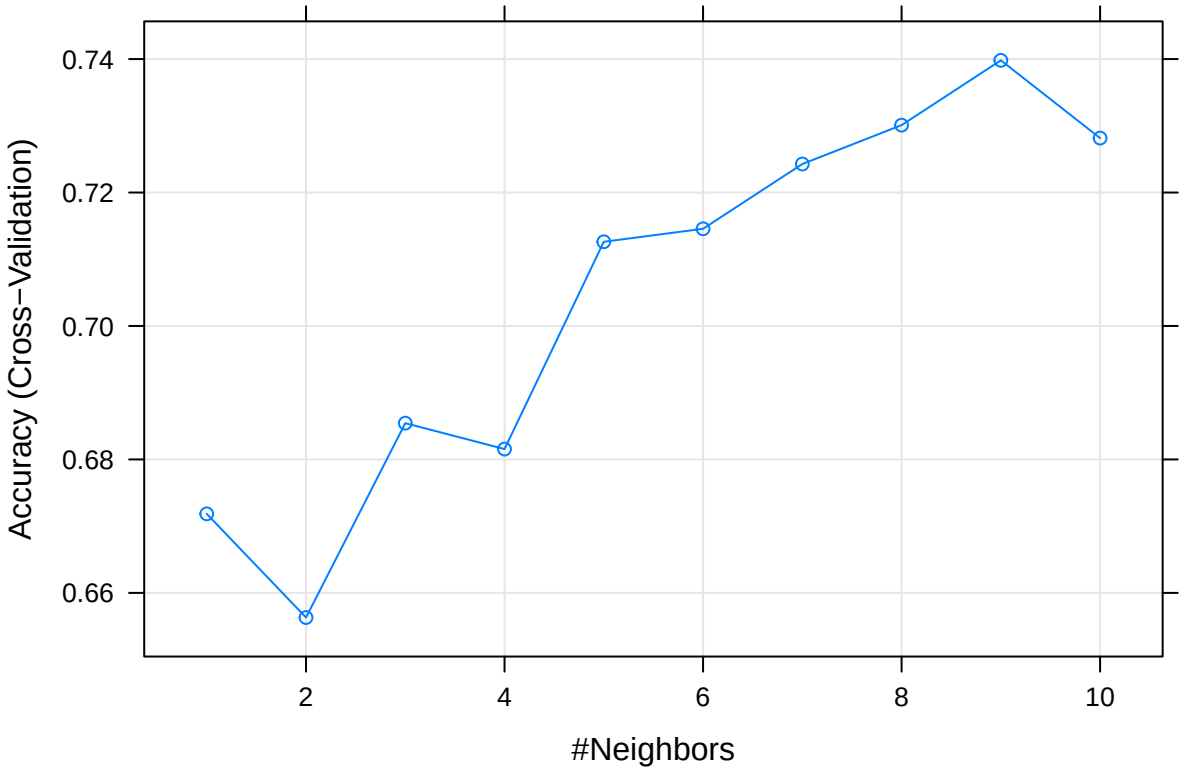
```
test_y = ifelse(PimaIndiansDiabetes[test_ind_dia,9] =="neg",0,1)
pred_5 = predict(fit_5, newdata=PimaIndiansDiabetes[test_ind_dia,-9])
pred_5 = ifelse(pred_5 =="neg",0,1)
test_error_5 = mean((test_y - pred_5)^2)

pred_loo = predict(fit_loo, newdata=PimaIndiansDiabetes[test_ind_dia,-9])
pred_loo = ifelse(pred_loo =="neg",0,1)
test_error_loo = mean((test_y - pred_loo)^2)
cbind(test_error_5, test_error_loo)
```

```
##      test_error_5 test_error_loo
## [1,]     0.229249       0.229249
```

```
plot(fit_5)
```


Accuracy (Cross−Validation)
0.72
0.70
0.68
0.66
2
4
6
8
10
#Neighbors

```
plot(fit_loo)
```



Loo cross-validation gives a lower test error.

2

Using gam function with select = TRUE for model selectin.

https://www.rdocumentation.org/packages/mgcv/versions/1.8-33/topics/gam.selection :

"The second approach leaves the original smoothing penalty unchanged, but constructs an additional penalty for each smooth, which penalizes only functions in the null space of the original penalty (the 'completely smooth' functions). Hence, if all the smoothing parameters for a term tend to infinity, the term will be selected out of the model. This latter approach is more expensive computationally, but has the advantage that it can be applied automatically to any smooth term. The select argument to gam turns on this method."

```
library(mgcv)
```

```
## Loading required package: nlme
```

```
## This is mgcv 1.8-31. For overview type 'help("mgcv-package")'.
```

```
##
## Attaching package: 'mgcv'
```

```
## The following objects are masked from 'package:gam':
##
##     gam, gam.control, gam.fit, s
```

```
train_2 = PimaIndiansDiabetes[train_ind_dia,]
test_2 = PimaIndiansDiabetes[-train_ind_dia,]
train_2_a = train_2
test_2_a = test_2
```

```
train_2[,9] = ifelse(PimaIndiansDiabetes[train_ind_dia,9] =="neg",0,1)
test_2[,9] = ifelse(PimaIndiansDiabetes[-train_ind_dia,9] =="neg",0,1)
gam_model = gam(diabetes ~ s(pregnant) + s(glucose) + s(pressure) + s(triceps) + s(insulin)
                + s(mass) + s(pedigree) + s(age), data = train_2, select = TRUE)
summary(gam_model)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## diabetes ~ s(pregnant) + s(glucose) + s(pressure) + s(triceps) +
##     s(insulin) + s(mass) + s(pedigree) + s(age)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.34951    0.01691   20.67   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                   edf Ref.df      F  p-value
## s(pregnant) 3.946e-01      9  0.064 0.209680
## s(glucose)  2.258e+00      9 13.193  < 2e-16 ***
## s(pressure) 9.218e-01      9  0.791 0.005215 **
## s(triceps)  2.442e-09      9  0.000 0.680987
## s(insulin)  3.228e-01      9  0.037 0.318273
## s(mass)     1.905e+00      9  2.272 3.69e-06 ***
## s(pedigree) 1.625e+00      9  1.472 0.000261 ***
## s(age)      2.396e+00      9  3.127 9.95e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.354   Deviance explained = 36.6%
## GCV = 0.15038  Scale est. = 0.14722   n = 515
```

```
mean((train_2[,9] - predict(gam_model, newdata = train_2[, -9]))^2)
```

```
## [1] 0.1441252
```

```
mean((test_2[,9] - predict(gam_model, newdata = test_2[, -9]))^2)
```

```
## [1] 0.1482968
```

From my result it seems all variables is in the best model have all variables exept triceps. The R-squared is not that high, so this method might not be the best for creating a model for this data. The test and train error is lower here than when using knn. Here the test error is slightly lower than train error. Where train error is 0.2251 and test error 0.2246.

3

```
errors2 = NULL
#classification tree
library(rpart)
#grow tree
```

```r
tree = rpart(diabetes ~., data=train_2, method="class")
printcp(tree)
```

```
##
## Classification tree:
## rpart(formula = diabetes ~ ., data = train_2, method = "class")
##
## Variables actually used in tree construction:
## [1] age      glucose  insulin  mass     pedigree pressure
##
## Root node error: 180/515 = 0.34951
##
## n= 515
##
##         CP nsplit rel error  xerror     xstd
## 1 0.261111      0   1.00000 1.00000 0.060115
## 2 0.094444      1   0.73889 0.82778 0.057169
## 3 0.019444      2   0.64444 0.75556 0.055579
## 4 0.016667      4   0.60556 0.71111 0.054486
## 5 0.012963      8   0.53889 0.70556 0.054343
## 6 0.010000     14   0.42222 0.68333 0.053755
```

```r
#choose the complexity parameter with smallest cv-error and prune
prune_tree = prune(tree, cp= tree$cptable[which.min(tree$cptable[,"xerror"]),"CP"])
summary(prune_tree)
```

```
## Call:
## rpart(formula = diabetes ~ ., data = train_2, method = "class")
##   n= 515
##
##           CP nsplit rel error    xerror       xstd
## 1 0.26111111      0 1.0000000 1.0000000 0.06011496
## 2 0.09444444      1 0.7388889 0.8277778 0.05716860
## 3 0.01944444      2 0.6444444 0.7555556 0.05557928
## 4 0.01666667      4 0.6055556 0.7111111 0.05448593
## 5 0.01296296      8 0.5388889 0.7055556 0.05434275
## 6 0.01000000     14 0.4222222 0.6833333 0.05375507
##
## Variable importance
##  glucose     mass      age  insulin pressure pedigree pregnant  triceps
##       34       15       15       12        7        7        6        4
##
## Node number 1: 515 observations,    complexity param=0.2611111
##   predicted class=0  expected loss=0.3495146  P(node) =1
##     class counts:   335   180
##    probabilities: 0.650 0.350
##   left son=2 (324 obs) right son=3 (191 obs)
##   Primary splits:
##       glucose  < 127.5  to the left,  improve=45.426620, (0 missing)
##       mass     < 26.45  to the left,  improve=24.664530, (0 missing)
##       age      < 30.5   to the left,  improve=24.536520, (0 missing)
##       pregnant < 6.5    to the left,  improve=12.080650, (0 missing)
##       insulin  < 121    to the left,  improve= 9.046429, (0 missing)
##   Surrogate splits:
```

```
##       insulin  < 125.5  to the left,  agree=0.701, adj=0.194, (0 split)
##       pressure < 81     to the left,  agree=0.660, adj=0.084, (0 split)
##       mass     < 39.75  to the left,  agree=0.652, adj=0.063, (0 split)
##       age      < 34.5   to the left,  agree=0.650, adj=0.058, (0 split)
##       triceps  < 40.5   to the left,  agree=0.637, adj=0.021, (0 split)
##
## Node number 2: 324 observations,    complexity param=0.01296296
##   predicted class=0  expected loss=0.1882716  P(node) =0.6291262
##     class counts:   263    61
##    probabilities: 0.812 0.188
##   left son=4 (93 obs) right son=5 (231 obs)
##   Primary splits:
##       mass     < 26.45  to the left,  improve=8.221201, (0 missing)
##       age      < 28.5   to the left,  improve=8.000309, (0 missing)
##       glucose  < 108.5  to the left,  improve=7.013954, (0 missing)
##       pedigree < 0.676  to the left,  improve=3.855864, (0 missing)
##       pregnant < 5.5    to the left,  improve=3.076923, (0 missing)
##   Surrogate splits:
##       pressure < 15     to the left,  agree=0.731, adj=0.065, (0 split)
##       age      < 21.5   to the left,  agree=0.725, adj=0.043, (0 split)
##       pedigree < 0.0845 to the left,  agree=0.719, adj=0.022, (0 split)
##
## Node number 3: 191 observations,    complexity param=0.09444444
##   predicted class=1  expected loss=0.3769634  P(node) =0.3708738
##     class counts:    72   119
##    probabilities: 0.377 0.623
##   left son=6 (51 obs) right son=7 (140 obs)
##   Primary splits:
##       mass     < 29.85  to the left,  improve=11.679180, (0 missing)
##       glucose  < 166.5  to the left,  improve=10.206200, (0 missing)
##       age      < 24.5   to the left,  improve= 6.404662, (0 missing)
##       pedigree < 0.429  to the left,  improve= 4.690502, (0 missing)
##       pregnant < 6.5    to the left,  improve= 3.602326, (0 missing)
##   Surrogate splits:
##       age      < 21.5   to the left,  agree=0.743, adj=0.039, (0 split)
##       pedigree < 0.1315 to the left,  agree=0.738, adj=0.020, (0 split)
##
## Node number 4: 93 observations
##   predicted class=0  expected loss=0.01075269  P(node) =0.1805825
##     class counts:    92     1
##    probabilities: 0.989 0.011
##
## Node number 5: 231 observations,    complexity param=0.01296296
##   predicted class=0  expected loss=0.2597403  P(node) =0.4485437
##     class counts:   171    60
##    probabilities: 0.740 0.260
##   left son=10 (117 obs) right son=11 (114 obs)
##   Primary splits:
##       age      < 28.5   to the left,  improve=8.203638, (0 missing)
##       glucose  < 99.5   to the left,  improve=7.213591, (0 missing)
##       pedigree < 0.6865 to the left,  improve=4.448325, (0 missing)
##       pregnant < 5.5    to the left,  improve=3.675897, (0 missing)
##       insulin  < 8      to the right, improve=3.261008, (0 missing)
##   Surrogate splits:
```

```
##       pregnant < 3.5    to the left,  agree=0.766, adj=0.526, (0 split)
##       pressure < 69     to the left,  agree=0.649, adj=0.289, (0 split)
##       insulin  < 8      to the right, agree=0.649, adj=0.289, (0 split)
##       triceps  < 5      to the right, agree=0.619, adj=0.228, (0 split)
##       glucose  < 98.5   to the left,  agree=0.610, adj=0.211, (0 split)
##
## Node number 6: 51 observations,    complexity param=0.01666667
##   predicted class=0  expected loss=0.3333333  P(node) =0.09902913
##     class counts:    34    17
##    probabilities: 0.667 0.333
##   left son=12 (12 obs) right son=13 (39 obs)
##   Primary splits:
##       age      < 22.5   to the left,  improve=3.487179, (0 missing)
##       glucose  < 145.5  to the left,  improve=2.946667, (0 missing)
##       mass     < 23.2   to the left,  improve=1.803030, (0 missing)
##       pedigree < 0.3185 to the left,  improve=1.759834, (0 missing)
##       pregnant < 1.5    to the left,  improve=1.400257, (0 missing)
##   Surrogate splits:
##       pregnant < 1.5    to the left,  agree=0.843, adj=0.333, (0 split)
##       glucose  < 131.5  to the left,  agree=0.824, adj=0.250, (0 split)
##       insulin  < 169    to the right, agree=0.804, adj=0.167, (0 split)
##       pressure < 60.5   to the left,  agree=0.784, adj=0.083, (0 split)
##       mass     < 23.2   to the left,  agree=0.784, adj=0.083, (0 split)
##
## Node number 7: 140 observations,    complexity param=0.01944444
##   predicted class=1  expected loss=0.2714286  P(node) =0.2718447
##     class counts:    38   102
##    probabilities: 0.271 0.729
##   left son=14 (95 obs) right son=15 (45 obs)
##   Primary splits:
##       glucose  < 165.5  to the left,  improve=5.560902, (0 missing)
##       pressure < 61     to the right, improve=3.239197, (0 missing)
##       pedigree < 0.438  to the left,  improve=2.699799, (0 missing)
##       age      < 24.5   to the left,  improve=2.287862, (0 missing)
##       pregnant < 7.5    to the left,  improve=2.064190, (0 missing)
##   Surrogate splits:
##       insulin  < 452.5  to the left,  agree=0.686, adj=0.022, (0 split)
##       mass     < 30.2   to the right, agree=0.686, adj=0.022, (0 split)
##       pedigree < 1.3925 to the left,  agree=0.686, adj=0.022, (0 split)
##       age      < 58.5   to the left,  agree=0.686, adj=0.022, (0 split)
##
## Node number 10: 117 observations
##   predicted class=0  expected loss=0.1282051  P(node) =0.2271845
##     class counts:   102    15
##    probabilities: 0.872 0.128
##
## Node number 11: 114 observations,    complexity param=0.01296296
##   predicted class=0  expected loss=0.3947368  P(node) =0.2213592
##     class counts:    69    45
##    probabilities: 0.605 0.395
##   left son=22 (67 obs) right son=23 (47 obs)
##   Primary splits:
##       glucose  < 110.5  to the left,  improve=5.166603, (0 missing)
##       pedigree < 0.514  to the left,  improve=4.926676, (0 missing)
```

```
##       insulin  < 145    to the left,  improve=4.879398, (0 missing)
##       mass     < 27.7   to the right, improve=3.968967, (0 missing)
##       pressure < 87     to the right, improve=1.702930, (0 missing)
##   Surrogate splits:
##       insulin  < 90.5   to the left,  agree=0.658, adj=0.170, (0 split)
##       mass     < 28.95  to the right, agree=0.632, adj=0.106, (0 split)
##       pressure < 25     to the right, agree=0.614, adj=0.064, (0 split)
##       triceps  < 15.5   to the right, agree=0.605, adj=0.043, (0 split)
##       pedigree < 1.105  to the left,  agree=0.605, adj=0.043, (0 split)
##
## Node number 12: 12 observations
##   predicted class=0  expected loss=0  P(node) =0.02330097
##     class counts:    12     0
##    probabilities: 1.000 0.000
##
## Node number 13: 39 observations,    complexity param=0.01666667
##   predicted class=0  expected loss=0.4358974  P(node) =0.07572816
##     class counts:    22    17
##    probabilities: 0.564 0.436
##   left son=26 (13 obs) right son=27 (26 obs)
##   Primary splits:
##       age      < 50.5   to the right, improve=3.102564, (0 missing)
##       pedigree < 0.3135 to the left,  improve=2.687509, (0 missing)
##       glucose  < 166.5  to the left,  improve=1.985939, (0 missing)
##       pressure < 79     to the right, improve=1.641026, (0 missing)
##       insulin  < 14.5   to the left,  improve=1.312821, (0 missing)
##   Surrogate splits:
##       pregnant < 4.5    to the right, agree=0.718, adj=0.154, (0 split)
##       glucose  < 138    to the left,  agree=0.692, adj=0.077, (0 split)
##       mass     < 22.75  to the left,  agree=0.692, adj=0.077, (0 split)
##       pedigree < 0.159  to the left,  agree=0.692, adj=0.077, (0 split)
##
## Node number 14: 95 observations,    complexity param=0.01944444
##   predicted class=1  expected loss=0.3684211  P(node) =0.184466
##     class counts:    35    60
##    probabilities: 0.368 0.632
##   left son=28 (21 obs) right son=29 (74 obs)
##   Primary splits:
##       insulin  < 219    to the right, improve=4.796112, (0 missing)
##       pressure < 61     to the right, improve=4.088575, (0 missing)
##       pregnant < 7.5    to the left,  improve=3.805597, (0 missing)
##       age      < 24.5   to the left,  improve=3.386845, (0 missing)
##       mass     < 30.85  to the right, improve=2.371446, (0 missing)
##
## Node number 15: 45 observations
##   predicted class=1  expected loss=0.06666667  P(node) =0.08737864
##     class counts:     3    42
##    probabilities: 0.067 0.933
##
## Node number 22: 67 observations
##   predicted class=0  expected loss=0.2686567  P(node) =0.1300971
##     class counts:    49    18
##    probabilities: 0.731 0.269
##
```

```
## Node number 23: 47 observations,    complexity param=0.01296296
##   predicted class=1  expected loss=0.4255319  P(node) =0.09126214
##     class counts:    20    27
##    probabilities: 0.426 0.574
##   left son=46 (30 obs) right son=47 (17 obs)
##   Primary splits:
##       pedigree < 0.512  to the left,  improve=5.049312, (0 missing)
##       pressure < 85     to the right, improve=3.562507, (0 missing)
##       age      < 31.5   to the left,  improve=1.295465, (0 missing)
##       pregnant < 1.5    to the right, improve=1.292237, (0 missing)
##       insulin  < 145    to the left,  improve=1.292237, (0 missing)
##   Surrogate splits:
##       triceps  < 36.5   to the left,  agree=0.723, adj=0.235, (0 split)
##       insulin  < 145    to the left,  agree=0.723, adj=0.235, (0 split)
##       mass     < 41     to the left,  agree=0.723, adj=0.235, (0 split)
##       pregnant < 10.5   to the left,  agree=0.681, adj=0.118, (0 split)
##       glucose  < 111.5  to the right, agree=0.660, adj=0.059, (0 split)
##
## Node number 26: 13 observations
##   predicted class=0  expected loss=0.1538462  P(node) =0.02524272
##     class counts:    11     2
##    probabilities: 0.846 0.154
##
## Node number 27: 26 observations,    complexity param=0.01666667
##   predicted class=1  expected loss=0.4230769  P(node) =0.05048544
##     class counts:    11    15
##    probabilities: 0.423 0.577
##   left son=54 (13 obs) right son=55 (13 obs)
##   Primary splits:
##       pedigree < 0.3085 to the left,  improve=3.769231, (0 missing)
##       pregnant < 5      to the left,  improve=2.053419, (0 missing)
##       glucose  < 160    to the left,  improve=2.053419, (0 missing)
##       pressure < 71     to the right, improve=1.617308, (0 missing)
##       age      < 38     to the left,  improve=1.110608, (0 missing)
##   Surrogate splits:
##       triceps  < 7      to the left,  agree=0.692, adj=0.385, (0 split)
##       insulin  < 14.5   to the left,  agree=0.692, adj=0.385, (0 split)
##       glucose  < 151.5  to the left,  agree=0.654, adj=0.308, (0 split)
##       pressure < 74.5   to the right, agree=0.654, adj=0.308, (0 split)
##       age      < 35     to the right, agree=0.654, adj=0.308, (0 split)
##
## Node number 28: 21 observations,    complexity param=0.01666667
##   predicted class=0  expected loss=0.3333333  P(node) =0.0407767
##     class counts:    14     7
##    probabilities: 0.667 0.333
##   left son=56 (14 obs) right son=57 (7 obs)
##   Primary splits:
##       age      < 37     to the left,  improve=3.0476190, (0 missing)
##       pregnant < 4.5    to the left,  improve=2.1987180, (0 missing)
##       mass     < 39.3   to the right, improve=1.1217950, (0 missing)
##       glucose  < 149    to the right, improve=1.0606060, (0 missing)
##       insulin  < 314    to the left,  improve=0.3888889, (0 missing)
##   Surrogate splits:
##       pregnant < 4.5    to the left,  agree=0.952, adj=0.857, (0 split)
```

```
##       insulin  < 246    to the right, agree=0.762, adj=0.286, (0 split)
##       glucose  < 145.5  to the right, agree=0.714, adj=0.143, (0 split)
##       mass     < 34.8   to the right, agree=0.714, adj=0.143, (0 split)
##       pedigree < 0.4395 to the left,  agree=0.714, adj=0.143, (0 split)
##
## Node number 29: 74 observations
##   predicted class=1  expected loss=0.2837838  P(node) =0.1436893
##     class counts:    21    53
##    probabilities: 0.284 0.716
##
## Node number 46: 30 observations,    complexity param=0.01296296
##   predicted class=0  expected loss=0.4  P(node) =0.05825243
##     class counts:    18    12
##    probabilities: 0.600 0.400
##   left son=92 (10 obs) right son=93 (20 obs)
##   Primary splits:
##       pressure < 77     to the right, improve=4.800000, (0 missing)
##       insulin  < 11     to the right, improve=1.653589, (0 missing)
##       mass     < 34.65  to the right, improve=1.653589, (0 missing)
##       triceps  < 26.5   to the right, improve=1.650000, (0 missing)
##       age      < 35     to the left,  improve=1.314027, (0 missing)
##   Surrogate splits:
##       glucose  < 125.5  to the right, agree=0.733, adj=0.2, (0 split)
##       pedigree < 0.4165 to the right, agree=0.733, adj=0.2, (0 split)
##       age      < 48     to the right, agree=0.733, adj=0.2, (0 split)
##       mass     < 36.95  to the right, agree=0.700, adj=0.1, (0 split)
##
## Node number 47: 17 observations
##   predicted class=1  expected loss=0.1176471  P(node) =0.03300971
##     class counts:     2    15
##    probabilities: 0.118 0.882
##
## Node number 54: 13 observations
##   predicted class=0  expected loss=0.3076923  P(node) =0.02524272
##     class counts:     9     4
##    probabilities: 0.692 0.308
##
## Node number 55: 13 observations
##   predicted class=1  expected loss=0.1538462  P(node) =0.02524272
##     class counts:     2    11
##    probabilities: 0.154 0.846
##
## Node number 56: 14 observations
##   predicted class=0  expected loss=0.1428571  P(node) =0.02718447
##     class counts:    12     2
##    probabilities: 0.857 0.143
##
## Node number 57: 7 observations
##   predicted class=1  expected loss=0.2857143  P(node) =0.01359223
##     class counts:     2     5
##    probabilities: 0.286 0.714
##
## Node number 92: 10 observations
##   predicted class=0  expected loss=0  P(node) =0.01941748
```

```
##     class counts:    10     0
##    probabilities: 1.000 0.000
##
## Node number 93: 20 observations,    complexity param=0.01296296
##   predicted class=1  expected loss=0.4  P(node) =0.03883495
##     class counts:     8    12
##    probabilities: 0.400 0.600
##   left son=186 (10 obs) right son=187 (10 obs)
##   Primary splits:
##       age      < 35     to the left,  improve=3.6000000, (0 missing)
##       mass     < 34.65  to the right, improve=2.1274730, (0 missing)
##       triceps  < 16     to the right, improve=1.3500000, (0 missing)
##       pressure < 67     to the right, improve=0.6000000, (0 missing)
##       pregnant < 3.5    to the left,  improve=0.2666667, (0 missing)
##   Surrogate splits:
##       triceps  < 6      to the right, agree=0.75, adj=0.5, (0 split)
##       insulin  < 52.5   to the right, agree=0.70, adj=0.4, (0 split)
##       mass     < 33.3   to the right, agree=0.70, adj=0.4, (0 split)
##       pregnant < 4.5    to the left,  agree=0.65, adj=0.3, (0 split)
##       pedigree < 0.2715 to the right, agree=0.65, adj=0.3, (0 split)
##
## Node number 186: 10 observations
##   predicted class=0  expected loss=0.3  P(node) =0.01941748
##     class counts:     7     3
##    probabilities: 0.700 0.300
##
## Node number 187: 10 observations
##   predicted class=1  expected loss=0.1  P(node) =0.01941748
##     class counts:     1     9
##    probabilities: 0.100 0.900
```

```
errors2$train_tree = mean((train_2[,9] - predict(prune_tree, newdata = train_2[, -9]))^2)
errors2$test_tree = mean((test_2[,9] - predict(prune_tree, newdata = test_2[, -9]))^2)

#bagging
library(ipred)
library(adabag)
```

```
## Loading required package: doParallel
## Loading required package: iterators
## Loading required package: parallel
##
## Attaching package: 'adabag'
## The following object is masked from 'package:ipred':
##
##     bagging
```

```
bag = bagging(diabetes ~., data=train_2_a)
#Using the pruning option
bagging.pred = predict.bagging(bag, newdata=test_2_a[, -9], newmfinal=3)
errors2$train_bag = mean(bag$class != train_2_a[,9])
errors2$test_bag = mean(bagging.pred$class != test_2_a[,9])
```

```r
#random forrest
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
rf = randomForest(x = train_2[,-9], y = train_2[,9],
                  xtest = test_2[,-9], ytest = test_2[,9], mtry = sqrt(ncol(train_2)))
```

```
## Warning in randomForest.default(x = train_2[, -9], y = train_2[, 9], xtest =
## test_2[, : The response has five or fewer unique values. Are you sure you want
## to do regression?
```

```r
rf
```

```
##
## Call:
##  randomForest(x = train_2[, -9], y = train_2[, 9], xtest = test_2[,      -9], ytest = test_2[, 9], m
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 0.1665569
##                     % Var explained: 26.74
##                        Test set MSE: 0.08
##                     % Var explained: 66.57
```

```r
errors2$test_rf = mean(rf$mse)
#Adaboost
library(mboost)
```

```
## Loading required package: stabs
```

```
## This is mboost 2.9-3. See 'package?mboost' and 'news(package  = "mboost")'
## for a complete list of changes.
```

```
##
## Attaching package: 'mboost'
```

```
## The following object is masked from 'package:ipred':
##
##     cv
```

```
## The following object is masked from 'package:ggplot2':
##
##     %+%
```

```
## The following object is masked from 'package:glmnet':
##
##     Cindex
```

```r
AdaBoost = mboost(as.factor(diabetes)~., data = as.data.frame(train_2),
                  family = AdaExp(), baselearner = 'btree',
```

```
                  boost_control(mstop = 400))
summary(AdaBoost)
```

```
##
##   Model-based Boosting
##
## Call:
## mboost(formula = as.factor(diabetes) ~ ., data = as.data.frame(train_2),     na.action = boost_contr
##
##
##   Adaboost Exponential Error
##
## Loss function: exp(-y * f)
##
##
## Number of boosting iterations: mstop = 100
## Step size:  0.1
## Offset:  -0.3105868
## Number of baselearners:  8
##
## Selection frequencies:
##  btree(glucose)     btree(mass)      btree(age) btree(pedigree) btree(pregnant)
##            0.37            0.26            0.22            0.12            0.03
```

```
error_adaBoost = NULL
for (i in 1:400) {
  pred_ada = as.factor(predict(AdaBoost[i], newdata = as.data.frame(test_2[,-9]), type = 'class'))
  error_adaBoost[i] = mean(test_2[,9] != pred_ada)
}
errors2$test_ada = mean(error_adaBoost)
errors2
```

```
## $train_tree
## [1] 0.38518
##
## $test_tree
## [1] 0.3849138
##
## $train_bag
## [1] 0.1184466
##
## $test_bag
## [1] 0.1981747
##
## $test_rf
## [1] 0.1710728
##
## $test_ada
## [1] 0.2203879
```

The classification tree gives the highest test error. The random forrest gives the lowest test error. Bagging gives a lower test and train error than classification tree. AdaBoost have a slightly higher test error than bagging.

4

I would use the method of random forrest beacuse it is easy to implement and gives a low test error. Random forrest can also handle binary, categorical and numerical features and the data do not need to be preprossed a lot like other method. The data don't need to be scaled. The method is also fast. Random forrest gives low bias and some higher variance.

5

```
error_5 = NULL

data(PimaIndiansDiabetes2)
data_3 = na.omit(PimaIndiansDiabetes2)
train_ind_dia_3 = sample.split(data_3$diabetes, SplitRatio = 0.67)
train_3 = data_3[train_ind_dia_3, ]
test_3 = data_3[-train_ind_dia_3, ]

#knn from 2.1
#choosing k = 5 in knn
trControl_5 = trainControl(method  = "cv",  number  = 5)
trControl_loo = trainControl(method  = "cv",  number  = nrow(train_3))
fit_5 = train(x = train_3[,-9],
              y = train_3[,9],
              method     = "knn",
              tuneGrid   = expand.grid(k = 5),
              trControl  = trControl_5,
              metric     = "Accuracy")
fit_loo = train(x = train_3[,-9],
                y = train_3[,9],
                method     = "knn",
                tuneGrid   = expand.grid(k = 5),
                trControl  = trControl_loo,
                metric     = "Accuracy")
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```
summary(fit_5)
```

```
##             Length Class      Mode
## learn       2      -none-     list
## k           1      -none-     numeric
## theDots     0      -none-     list
## xNames      8      -none-     character
## problemType 1      -none-     character
## tuneValue   1      data.frame list
## obsLevels   2      -none-     character
## param       0      -none-     list
```

```
summary(fit_loo)
```

```
##             Length Class      Mode
## learn       2      -none-     list
## k           1      -none-     numeric
## theDots     0      -none-     list
## xNames      8      -none-     character
```

```
## problemType 1      -none-     character
## tuneValue   1      data.frame list
## obsLevels   2      -none-     character
## param       0      -none-     list
```

```
test_y = ifelse(test_3[,9] =="neg",0,1)
pred_5 = predict(fit_5, newdata=test_3[,-9])
pred_5 = ifelse(pred_5 =="neg",0,1)
error_5$knn5 = mean((test_y - pred_5)^2)

pred_loo = predict(fit_loo, newdata=test_3[,-9])
pred_loo = ifelse(pred_loo =="neg",0,1)
error_5$knnloo = mean((test_y - pred_loo)^2)

#gam from 2.2
train_3_a = train_3
test_3_a = test_3
train_3[,9] = ifelse(train_3[,9] =="neg",0,1)
test_3[,9] = ifelse(test_3[,9] =="neg",0,1)
gam_model_2 = gam(diabetes ~ s(pregnant) + s(glucose) + s(pressure) + s(triceps) + s(insulin)
                + s(mass) + s(pedigree) + s(age), data = train_3, select = TRUE)
summary(gam_model)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## diabetes ~ s(pregnant) + s(glucose) + s(pressure) + s(triceps) +
##     s(insulin) + s(mass) + s(pedigree) + s(age)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.34951    0.01691   20.67   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                   edf Ref.df      F  p-value
## s(pregnant) 3.946e-01      9  0.064 0.209680
## s(glucose)  2.258e+00      9 13.193  < 2e-16 ***
## s(pressure) 9.218e-01      9  0.791 0.005215 **
## s(triceps)  2.442e-09      9  0.000 0.680987
## s(insulin)  3.228e-01      9  0.037 0.318273
## s(mass)     1.905e+00      9  2.272 3.69e-06 ***
## s(pedigree) 1.625e+00      9  1.472 0.000261 ***
## s(age)      2.396e+00      9  3.127 9.95e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.354   Deviance explained = 36.6%
## GCV = 0.15038  Scale est. = 0.14722   n = 515
```

```
error_5$train_gam = mean((train_3[,9] - predict(gam_model_2, newdata = train_3[, -9]))^2)
error_5$test_gam = mean((test_3[,9] - predict(gam_model_2, newdata = test_3[, -9]))^2)
```

```r
#classification tree
#grow tree
tree = rpart(diabetes ~., data=train_3, method="class")
printcp(tree)
```

```
##
## Classification tree:
## rpart(formula = diabetes ~ ., data = train_3, method = "class")
##
## Variables actually used in tree construction:
## [1] age      glucose  mass     pedigree
##
## Root node error: 87/263 = 0.3308
##
## n= 263
##
##         CP nsplit rel error  xerror     xstd
## 1 0.321839      0   1.00000 1.00000 0.087704
## 2 0.063218      1   0.67816 0.80460 0.082382
## 3 0.025862      3   0.55172 0.75862 0.080818
## 4 0.011494      7   0.44828 0.83908 0.083472
## 5 0.010000      8   0.43678 0.86207 0.084161
```

```r
#choose the complexity parameter with smallest cv-error and prune
prune_tree = prune(tree, cp= tree$cptable[which.min(tree$cptable[,"xerror"]),"CP"])
summary(prune_tree)
```

```
## Call:
## rpart(formula = diabetes ~ ., data = train_3, method = "class")
##   n= 263
##
##           CP nsplit rel error    xerror       xstd
## 1 0.32183908      0 1.0000000 1.0000000 0.08770389
## 2 0.06321839      1 0.6781609 0.8045977 0.08238166
## 3 0.02586207      3 0.5517241 0.7586207 0.08081797
##
## Variable importance
##  glucose      age  insulin pregnant pedigree pressure  triceps     mass
##       53       14       14       10        4        3        2        1
##
## Node number 1: 263 observations,    complexity param=0.3218391
##   predicted class=0  expected loss=0.3307985  P(node) =1
##     class counts:   176    87
##    probabilities: 0.669 0.331
##   left son=2 (175 obs) right son=3 (88 obs)
##   Primary splits:
##       glucose  < 135    to the left,  improve=28.50704, (0 missing)
##       insulin  < 143    to the left,  improve=15.08791, (0 missing)
##       age      < 28.5   to the left,  improve=12.25486, (0 missing)
##       mass     < 30.05  to the left,  improve=11.18944, (0 missing)
##       pregnant < 7.5    to the left,  improve=11.03113, (0 missing)
##   Surrogate splits:
##       insulin  < 222.5  to the left,  agree=0.757, adj=0.273, (0 split)
```

```
##       age      < 49.5   to the left,  agree=0.719, adj=0.159, (0 split)
##       pregnant < 6.5    to the left,  agree=0.700, adj=0.102, (0 split)
##       pedigree < 1.9935 to the left,  agree=0.677, adj=0.034, (0 split)
##       pressure < 81     to the left,  agree=0.673, adj=0.023, (0 split)
##
## Node number 2: 175 observations
##   predicted class=0  expected loss=0.1657143  P(node) =0.6653992
##     class counts:   146    29
##    probabilities: 0.834 0.166
##
## Node number 3: 88 observations,    complexity param=0.06321839
##   predicted class=1  expected loss=0.3409091  P(node) =0.3346008
##     class counts:    30    58
##    probabilities: 0.341 0.659
##   left son=6 (49 obs) right son=7 (39 obs)
##   Primary splits:
##       glucose  < 157.5  to the left,  improve=6.337710, (0 missing)
##       mass     < 29.8   to the left,  improve=5.121212, (0 missing)
##       age      < 27.5   to the left,  improve=3.116883, (0 missing)
##       pedigree < 0.3125 to the left,  improve=2.660038, (0 missing)
##       pregnant < 6.5    to the left,  improve=2.456033, (0 missing)
##   Surrogate splits:
##       insulin  < 173    to the left,  agree=0.648, adj=0.205, (0 split)
##       triceps  < 35.5   to the left,  agree=0.636, adj=0.179, (0 split)
##       pedigree < 0.577  to the left,  agree=0.636, adj=0.179, (0 split)
##       pregnant < 3.5    to the right, agree=0.614, adj=0.128, (0 split)
##       mass     < 30.45  to the left,  agree=0.614, adj=0.128, (0 split)
##
## Node number 6: 49 observations,    complexity param=0.06321839
##   predicted class=0  expected loss=0.4897959  P(node) =0.1863118
##     class counts:    25    24
##    probabilities: 0.510 0.490
##   left son=12 (31 obs) right son=13 (18 obs)
##   Primary splits:
##       age      < 41     to the left,  improve=4.719187, (0 missing)
##       mass     < 26.4   to the left,  improve=4.587357, (0 missing)
##       pregnant < 7.5    to the left,  improve=3.432653, (0 missing)
##       triceps  < 23.5   to the left,  improve=2.544674, (0 missing)
##       pressure < 75     to the left,  improve=1.239796, (0 missing)
##   Surrogate splits:
##       pregnant < 6.5    to the left,  agree=0.878, adj=0.667, (0 split)
##       pressure < 85.5   to the left,  agree=0.714, adj=0.222, (0 split)
##       triceps  < 42.5   to the left,  agree=0.673, adj=0.111, (0 split)
##       glucose  < 136.5  to the right, agree=0.653, adj=0.056, (0 split)
##       pedigree < 1.1815 to the left,  agree=0.653, adj=0.056, (0 split)
##
## Node number 7: 39 observations
##   predicted class=1  expected loss=0.1282051  P(node) =0.148289
##     class counts:     5    34
##    probabilities: 0.128 0.872
##
## Node number 12: 31 observations
##   predicted class=0  expected loss=0.3225806  P(node) =0.1178707
##     class counts:    21    10
```

```
##    probabilities: 0.677 0.323
##
## Node number 13: 18 observations
##   predicted class=1  expected loss=0.2222222  P(node) =0.06844106
##     class counts:     4    14
##    probabilities: 0.222 0.778
```

```
error_5$train_tree = mean((train_3[,9] - predict(prune_tree, newdata = train_3[, -9]))^2)
error_5$test_tree = mean((test_3[,9] - predict(prune_tree, newdata = test_3[, -9]))^2)

#bagging

bag = bagging(diabetes ~., data=train_3_a)
#Using the pruning option
bagging.pred = predict.bagging(bag, newdata=test_3_a[, -9], newmfinal=3)
error_5$train_bag = mean(bag$class != train_3_a[,9])
error_5$train_bag = mean(bagging.pred$class != test_3_a[,9])

#random forrest
library(randomForest)
rf = randomForest(x = train_3[,-9], y = train_3[,9],
                  xtest = test_3[,-9], ytest = test_3[,9], mtry = sqrt(ncol(train_3)))
```

```
## Warning in randomForest.default(x = train_3[, -9], y = train_3[, 9], xtest =
## test_3[, : The response has five or fewer unique values. Are you sure you want
## to do regression?
```

```
error_5$test_rf = mean(rf$mse)

#Adaboost
library(mboost)
AdaBoost = mboost(as.factor(diabetes)~., data = as.data.frame(train_3),
                  family = AdaExp(), baselearner = 'btree',
                  boost_control(mstop = 400))
summary(AdaBoost)
```

```
##
##   Model-based Boosting
##
## Call:
## mboost(formula = as.factor(diabetes) ~ ., data = as.data.frame(train_3),     na.action = boost_contr
##
##
##   Adaboost Exponential Error
##
## Loss function: exp(-y * f)
##
##
## Number of boosting iterations: mstop = 100
## Step size:  0.1
## Offset:  -0.3522879
## Number of baselearners:  8
##
## Selection frequencies:
##  btree(glucose)     btree(mass) btree(pedigree)      btree(age) btree(pregnant)
```

```
##            0.29            0.21            0.16            0.10            0.08
## btree(pressure)  btree(insulin)
##            0.08            0.08
```

```
error_adaBoost = NULL
for (i in 1:400) {
  pred_ada = as.factor(predict(AdaBoost[i], newdata = as.data.frame(test_3[,-9]), type = 'class'))
  error_adaBoost[i] = mean(test_3[,9] != pred_ada)
}
error_5$adaboost = mean(error_adaBoost)
error_5
```

```
## $knn5
## [1] 0.2276215
##
## $knnloo
## [1] 0.2276215
##
## $train_gam
## [1] 0.1257559
##
## $test_gam
## [1] 0.136812
##
## $train_tree
## [1] 0.3538458
##
## $test_tree
## [1] 0.3548529
##
## $train_bag
## [1] 0.1918159
##
## $test_rf
## [1] 0.1499263
##
## $adaboost
## [1] 0.1871036
```

When removinge false values in the data (such as the bmi of 0) it is reasonable that I get a lower error.

For knn I get a little lower test error loo cv, with this data the test error is about the same for 5 cv as the previous dataset.

For gam the train error is slightly lower here then with the first data. For the test error with gam comared to the previous data it is also lower.

For classification tree the test error and train error is also lower than with the other dataset.

For bagging the error is also lower.

For random forrest the residuals is lower and the test set MSE is much lower.

For AdaBoost the average test error is now 0.186, this is lower than in task 2.3.

The result is not surprisingly better for all the methods when removing missing values instead of setting them to 0.