Assignment 3

Shreyashi Ganguly

06/03/2020

Question 1: K-Nearest Neighbours

Part (a)

Use n-fold CV to find the best K for predicting cost using K-NN. What are the pros and cons of using n-fold CV, versus say 10-fold CV, for nearest neighbors?

```
[1] "n-fold Cross Validation results:"

      try_K        r2    MSEAve
 [1,]     2 0.4437908 0.3812603
 [2,]     3 0.5237883 0.3264251
 [3,]     4 0.5461043 0.3111283
 [4,]     5 0.5677138 0.2963158
 [5,]     6 0.5674404 0.2965032
 [6,]     7 0.5851995 0.2843300
 [7,]     8 0.5868482 0.2831999
 [8,]     9 0.5854198 0.2841790
 [9,]    10 0.5844870 0.2848184
[10,]    11 0.5792715 0.2883934
[11,]    12 0.5768682 0.2900408
[12,]    13 0.5721236 0.2932931
[13,]    14 0.5718091 0.2935086
[14,]    15 0.5714809 0.2937336
[15,]    16 0.5717559 0.2935451
[16,]    17 0.5688052 0.2955677
[17,]    18 0.5686762 0.2956561
[18,]    19 0.5682352 0.2959584
[19,]    20 0.5650374 0.2981504
```

Observations:

- r2 increases and MSEAve decreases as K increases till K=8 post which the results deretoriate.
- Optimum K=8
- CV R-sq at K=8 : 0.5868482

For nearest neighbours, doing n-fold CV is only marginally more expensive than 10-fold CV. However with n-fold there is no possibility of having replicates of CV, hence not possible to get estimation of standard deviation of CV results.

Also with n-fold, more of the data is available for searching for neighbours in the training data. This is particularly beneficial for this dataset as it has only ~800 rows and 8 variables. Hence n-fold CV is a good choice for K-NN

Part (b)

For the optimal K from part (a), what is the CV estimate of the prediction error standard deviation?

```
   [1] "CV estimate of the prediction error standard deviation for K=8 is
0.528960"
```

Part (c)

What is the predicted cost for a person with age=59, gend=0, intvn=10, drugs=0, ervis=3, comp=0, comorb=4, and dur=300?

```
   Target points completed:


   [1] "Predicted cost for the new data = 2396.013339"
```

Question 2: Generalized Additive Models

Part (a)

Fit a GAM model without interactions, and construct plots of the component functions. Which predictors appear to be the most relevant for predicting cost?

```
   [1] "Summary of GAM model:"


   Family: gaussian
   Link function: identity

   Formula:
   target ~ s(age) + gend + s(intvn) + s(drugs, k = 9) + s(ervis) +
       s(comp, k = 3) + s(comorb) + s(dur)

   Parametric coefficients:
               Estimate Std. Error t value Pr(>|t|)
   (Intercept)  2.73172    0.01654 165.194   <2e-16 ***
   gend        -0.02864    0.01685  -1.699   0.0897 .
   ---
   Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

   Approximate significance of smooth terms:
               edf Ref.df       F  p-value
   s(age)    1.000  1.000   3.651 0.056419 .
```

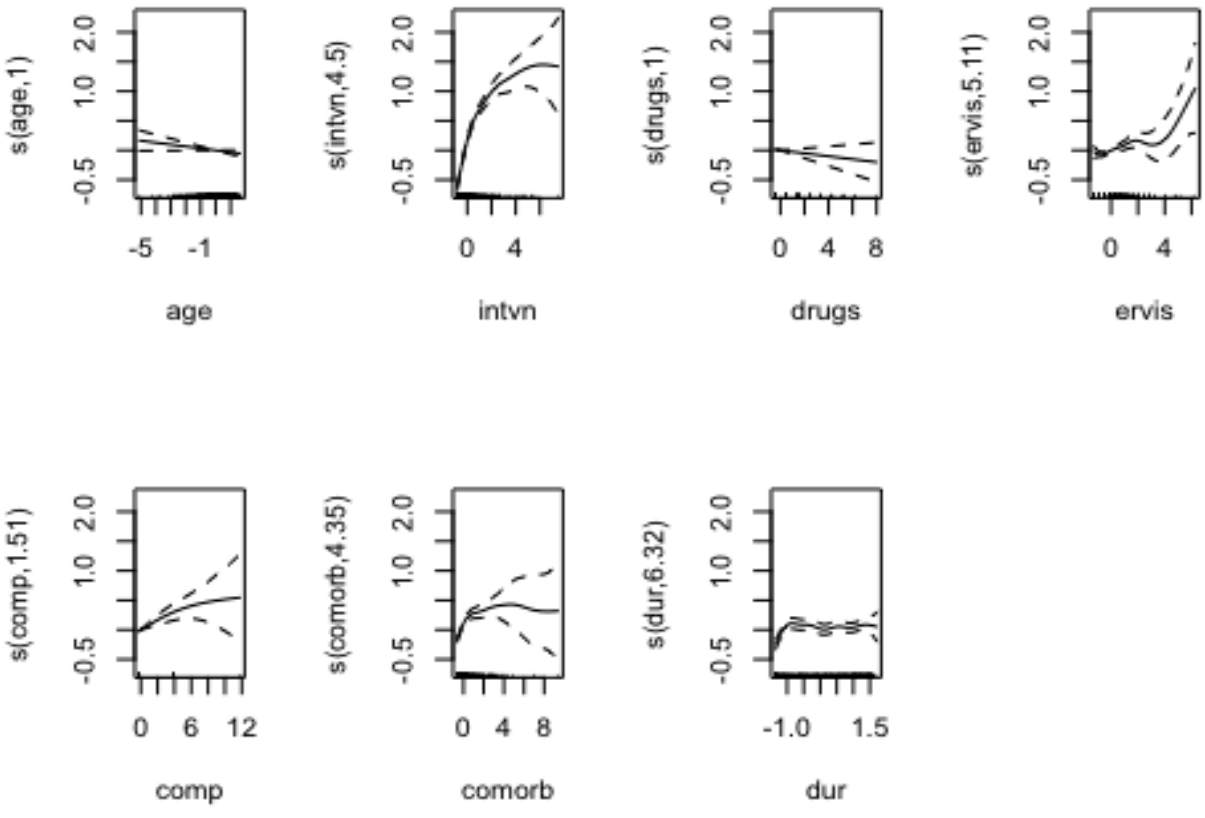
```
s(intvn)  4.500  5.444 137.853  < 2e-16 ***
s(drugs)  1.000  1.000   1.446 0.229498
s(ervis)  5.105  6.132   3.176 0.004760 **
s(comp)   1.506  1.753  11.658 0.000259 ***
s(comorb) 4.352  5.301  15.738 2.82e-15 ***
s(dur)    6.324  7.444   5.184 5.58e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.686   Deviance explained = 69.6%
GCV = 0.22277  Scale est. = 0.21548   n = 788

[1] "estimated smoothing parameters for each constituent function:"

      s(age)     s(intvn)     s(drugs)     s(ervis)      s(comp)
7.113768e+09 1.039629e-01 1.732089e+08 9.001982e-02 1.570520e-02
   s(comorb)       s(dur)
3.975592e-02 7.486655e-02

[1] "Component function plots:"
```



Observations:

Most significant variables for predicting cost

* intvn - positive impact

* ervis - positive impact

* comorb - positive impact

* comp - positive impact

Part (b)

For the model from part (a), what is the CV estimate of the prediction error standard deviation? What are the pros and cons of using n-fold CV, versus say 10- fold CV, for GAMs?

Approach followed - I have used the optimum smoothing parameters estimated for each of the constituent functions from the entire training data in cross validation. That way, each fold of CV is fitting the same model and hence are comparable

```
   [1] "n-fold Cross Validation results:"

                r2    MSEAve       MSEsd
   [1,] 0.6688331 0.2270023 0.004584893

   [1] "CV estimate of the prediction error standard deviation for GAM is
0.476722"
```

Observations:

CV for GAMs is considerably expensive. Hence the compute expense of n-fold is much higher Also with 10-fold, there is possibility of having multiple replicates of CV to better estimate CV error and CV error sd Hence 10-fold is better here

Part (c)

What is the predicted cost for a person with age=59, gend=0, intvn=10, drugs=0, ervis=3, comp=0, comorb=4, and dur=300?

```
   [1] "Predicted cost for new data = 3822.416083"
```

Question 3: Kernel method

Part (a)

Use CV to find the best combination of span and degree (0 for local average, 1 for local linear, and 2 for local quadratic regression) for a kernel method.

```
[1] "n-fold Cross Validation results:"
```

```
      model   r2                  MSEAve
 [1,] "0,0.1" "0.668693351184777" "0.227098141907631"
 [2,] "0,0.2" "0.646955952548317" "0.241998304213154"
 [3,] "0,0.3" "0.620366619468101" "0.260224283554873"
 [4,] "0,0.4" "0.592125492568908" "0.279582504909981"
 [5,] "0,0.5" "0.563954345964007" "0.298892757427577"
 [6,] "0,0.6" "0.537291241692872" "0.317169304122664"
 [7,] "0,0.7" "0.504538426138087" "0.339620116931083"
 [8,] "0,0.8" "0.456363330385368" "0.372642317876306"
 [9,] "0,0.9" "0.376678287516918" "0.427263392454702"
[10,] "1,0.1" "0.666493021735886" "0.228606384290375"
[11,] "1,0.2" "0.678114180807135" "0.220640520516341"
[12,] "1,0.3" "0.678581006439338" "0.220320529251308"
[13,] "1,0.4" "0.677561802108416" "0.2210191551636"
[14,] "1,0.5" "0.676127100728414" "0.222002588543992"
[15,] "1,0.6" "0.674278429670666" "0.223269782437366"
[16,] "1,0.7" "0.671293253083681" "0.225316007765457"
[17,] "1,0.8" "0.665862773703341" "0.22903839541253"
[18,] "1,0.9" "0.656711503429133" "0.235311244094556"
[19,] "2,0.1" "0.566855937654986" "0.296903826375476"
[20,] "2,0.2" "0.639556649109287" "0.247070246078564"
[21,] "2,0.3" "0.659626909512458" "0.233312843800457"
[22,] "2,0.4" "0.667165841648122" "0.228145191759366"
[23,] "2,0.5" "0.670441987171366" "0.225899518261389"
[24,] "2,0.6" "0.672490924413709" "0.224495049494205"
[25,] "2,0.7" "0.67355746986937"  "0.223763973036468"
[26,] "2,0.8" "0.673999048870184" "0.223461287379694"
[27,] "2,0.9" "0.673571865759559" "0.223754105199753"
      MSEsd
 [1,] "0.00134470885375889"
 [2,] "0.00106719575067878"
 [3,] "0.000973747655160708"
 [4,] "0.000980683945542326"
 [5,] "0.00106281001222016"
 [6,] "0.00108067485512067"
 [7,] "0.00108216049490817"
 [8,] "0.00110236060423225"
 [9,] "0.00111494141161481"
[10,] "0.00398114341643728"
[11,] "0.00333724007355024"
[12,] "0.00300138427984417"
```

```
[13,] "0.00275524210782376"
[14,] "0.00248684454243819"
[15,] "0.00231400729652217"
[16,] "0.0020509580675528"
[17,] "0.001819446705078"
[18,] "0.00148329561032093"
[19,] "0.0101937598482355"
[20,] "0.00682241525501506"
[21,] "0.0077306462699776"
[22,] "0.00778604677609779"
[23,] "0.00739630250635718"
[24,] "0.00737887833205413"
[25,] "0.00680521406143718"
[26,] "0.00665199719616083"
[27,] "0.00589485824206024"
```

- Optimum parameters from CV
  - degree = 1
  - span = 0.3

Part (b)

Use Cp to find the best combination of span and degree (0 for local average, 1 for local linear, and 2 for local quadratic regression) for a kernel method. Is this in agreement with what CV said was the best span and degree?

```
[1] "Sigma:"

   degree span     sigma
1       0  0.1 0.5017093
2       0  0.2 0.5588878
3       0  0.3 0.5990548
4       0  0.4 0.6500097
5       0  0.5 0.6670780
6       0  0.6 0.7039970
7       0  0.7 0.7110960
8       0  0.8 0.7268927
9       0  0.9 0.7374671
10      1  0.1 0.4671513
11      1  0.2 0.4708937
12      1  0.3 0.4694632
13      1  0.4 0.4696673
14      1  0.5 0.4695767
15      1  0.6 0.4718809
16      1  0.7 0.4749744
17      1  0.8 0.4794140
18      1  0.9 0.4874427
19      2  0.1 0.6802507
20      2  0.2 0.6234012
21      2  0.3 0.5598760
```

```
   22      2  0.4 0.5418840
   23      2  0.5 0.5069268
   24      2  0.6 0.4835245
   25      2  0.7 0.4906549
   26      2  0.8 0.4859783
   27      2  0.9 0.4897579


[1] "Cp results:"

      degree span        Cp
   1       0  0.1 0.2549250
   2       0  0.2 0.3128995
   3       0  0.3 0.3589436
   4       0  0.4 0.4225221
   5       0  0.5 0.4449462
   6       0  0.6 0.4953478
   7       0  0.7 0.5053961
   8       0  0.8 0.5280799
   9       0  0.9 0.5435749
   10      1  0.1 0.2320746
   11      1  0.2 0.2287123
   12      1  0.3 0.2254268
   13      1  0.4 0.2244547
   14      1  0.5 0.2239008
   15      1  0.6 0.2255043
   16      1  0.7 0.2280489
   17      1  0.8 0.2319376
   18      1  0.9 0.2393453
   19      2  0.1 0.4376935
   20      2  0.2 0.3842688
   21      2  0.3 0.3176363
   22      2  0.4 0.2984412
   23      2  0.5 0.2638013
   24      2  0.6 0.2409433
   25      2  0.7 0.2468122
   26      2  0.8 0.2418846
   27      2  0.9 0.2449247
```

- Optimum parameters from Cp
  - degree = 1
  - span = 0.5

The degree is same for CV and Cp.

However, Cp gives a span of 0.5, while CV gives a span of 0.3 as ideal

Part (c)

For the optimal model from part (a), what is the CV estimate of the prediction error standard deviation?

```
   [1] "CV estimate of the prediction error standard deviation for LOESS is
0.468297"
```

Part (d)

What is the predicted cost for a person with age=59, gend=0, intvn=10, drugs=0, ervis=3, comp=0, comorb=4, and dur=300?

```
   [1] "Predicted cost for new data from best model from CV = 2514.942272"

   [1] "Predicted cost for new data from best model from Cp = 2426.476524"
```

Question 4 - PPR

Part (a)

Use CV to find the best number of terms for the PPR model.

```
   [1] "CV Stats"

                    r2    MSEAve       MSEsd
    [1,]  1   0.6807251 0.2188508 0.001055627
    [2,]  2   0.6801512 0.2192442 0.003111371
    [3,]  3   0.6759600 0.2221171 0.003929858
    [4,]  4   0.6694665 0.2265682 0.004006345
    [5,]  5   0.6607227 0.2325618 0.002936395
    [6,]  6   0.6465648 0.2422665 0.007145435
    [7,]  7   0.6281095 0.2549168 0.004401795
    [8,]  8   0.6026607 0.2723610 0.017005725
    [9,]  9   0.5710926 0.2939998 0.024239843
   [10,] 10   0.5384331 0.3163866 0.020278929
   [11,] 11   0.5264117 0.3246268 0.031762710
   [12,] 12   0.5166043 0.3313494 0.036648329
   [13,] 13   0.5165546 0.3313835 0.037694684
   [14,] 14 -10.8852305 8.1468748 0.000000000
   [15,] 15 -10.8852305 8.1468748 0.000000000
   [16,] 20 -10.8852305 8.1468748 0.000000000
   [17,] 25 -10.8852305 8.1468748 0.000000000
   [18,] 30 -10.8852305 8.1468748 0.000000000
   [19,] 35 -10.8852305 8.1468748 0.000000000
   [20,] 40 -10.8852305 8.1468748 0.000000000
```

Optimum number of terms = 1

Part (b)

For the optimal model from part (a), what is the PPR model and the CV estimate of the prediction error standard deviation? Interpret the fitted model.

```
Call:
ppr(formula = target ~ ., data = heart, nterms = 1)

Goodness of fit:
 1 terms
165.5263

Projection direction vectors ('alpha'):
        age        gend       intvn       drugs       ervis        comp
-0.02759078 -0.02686806  0.93275393 -0.07100287  0.10038411  0.15520531
     comorb         dur
 0.28667863  0.08423339

Coefficients of ridge terms ('beta'):
   term 1
0.6888639

[1] "CV estimate of the prediction error standard deviation for PPR is
0.468111"
```

- PPR has only 1 term
- In Term1, intvn, comorb, comp and ervis are most significant - each with positive impact

Part (c)

What is the predicted cost for a person with age=59, gend=0, intvn=10, drugs=0, ervis=3, comp=0, comorb=4, and dur=300?

```
[1] "Predicted cost for new data = 3240.521955"
```

Question 5: Forensic Glass Data

Part (a)

Use CV to find the best nearest neighbor model for classifying the type.

```
[1] "n-fold Cross Validation results:"

      try_K CV.rateAve   CV.rateSD
 [1,]     2  0.1941121 0.009497622
 [2,]     3  0.1516822 0.008295450
 [3,]     4  0.1715888 0.009395253
 [4,]     5  0.1606542 0.005495902
 [5,]     6  0.1543925 0.004991968
 [6,]     7  0.1564486 0.007204666
 [7,]     8  0.1552336 0.005447034
 [8,]     9  0.1622430 0.006265083
 [9,]    10  0.1608411 0.005826512
[10,]    11  0.1679439 0.007276675
[11,]    12  0.1694393 0.006729422
[12,]    13  0.1777570 0.006259390
[13,]    14  0.1739252 0.006826727
[14,]    15  0.1775701 0.006118259
[15,]    16  0.1779439 0.007601944
[16,]    17  0.1763551 0.006030960
[17,]    18  0.1753271 0.005191515
[18,]    19  0.1730841 0.005167424
[19,]    20  0.1716822 0.004201897
```

Observations:

- Optimum K=3
- Average CV misclass rate with 3 neighbours = 15.17%

Part (b)

Use CV to find the best GAM for classifying the type. You can do this using the binomial family in the gam() function.

Approach - Since using multiple replicates of CV, finding smoothing parameters from whole training data and using that to obtain the CV misclass rate

```
[1] "Smoothing parameters from whole training data:"

       s(RI)        s(Na)        s(Mg)        s(Al)        s(Si)
2.559800e-01 4.645713e-04 1.196338e+04 4.322553e-04 6.293638e+03
        s(K)        s(Ca)        s(Ba)        s(Fe)
1.642143e-04 1.027133e-06 9.754248e-03 2.942407e-02

[1] "CV stats"
```

```
      CV.rateAve  CV.rateSD
[1,]  0.1565421 0.01211554
```

Part (c)

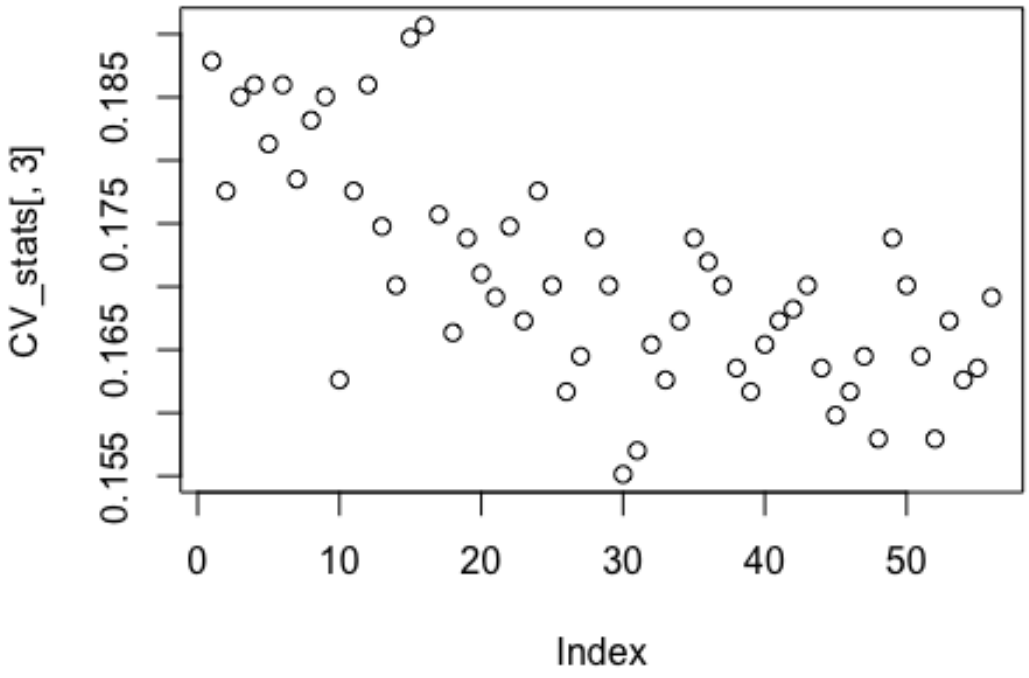
Compare the models in parts (a) and (b) with the best neural network model.

```
[1] "CV stats:"

                CV.rateAve   CV.rateSD
 [1,]  2 0.001  0.1878505 0.014173599
 [2,]  3 0.005  0.1775701 0.023596974
 [3,]  5 0.010  0.1850467 0.013862053
 [4,] 10 0.020  0.1859813 0.011156389
 [5,] 15 0.030  0.1813084 0.016652831
 [6,] 20 0.050  0.1859813 0.013381141
 [7,] 25 0.100  0.1785047 0.006092713
 [8,] 30 0.001  0.1831776 0.015983768
 [9,]  2 0.005  0.1850467 0.017041666
[10,]  3 0.010  0.1626168 0.015638505
[11,]  5 0.020  0.1775701 0.017793862
[12,] 10 0.030  0.1859813 0.012966763
[13,] 15 0.050  0.1747664 0.013862053
[14,] 20 0.100  0.1700935 0.018574399
[15,] 25 0.001  0.1897196 0.013050692
[16,] 30 0.005  0.1906542 0.028115157
[17,]  2 0.010  0.1757009 0.020528862
[18,]  3 0.020  0.1663551 0.015708165
[19,]  5 0.030  0.1738318 0.010130600
[20,] 10 0.050  0.1710280 0.012185425
[21,] 15 0.100  0.1691589 0.018808048
[22,] 20 0.001  0.1747664 0.020528862
[23,] 25 0.005  0.1672897 0.011156389
[24,] 30 0.010  0.1775701 0.024726648
[25,]  2 0.020  0.1700935 0.005327923
[26,]  3 0.030  0.1616822 0.015708165
[27,]  5 0.050  0.1644860 0.014553656
[28,] 10 0.100  0.1738318 0.008988497
[29,] 15 0.001  0.1700935 0.017359043
[30,] 20 0.005  0.1551402 0.017608826
[31,] 25 0.010  0.1570093 0.006269349
[32,] 30 0.020  0.1654206 0.018278138
[33,]  2 0.030  0.1626168 0.015285446
[34,]  3 0.050  0.1672897 0.012538699
[35,]  5 0.100  0.1738318 0.007678354
[36,] 10 0.001  0.1719626 0.011635420
[37,] 15 0.005  0.1700935 0.014628482
[38,] 20 0.010  0.1635514 0.009345794
[39,] 25 0.020  0.1616822 0.008488739
[40,] 30 0.030  0.1654206 0.007819253
```

```
[41,]  2 0.050  0.1672897 0.018218307
[42,]  3 0.100  0.1682243 0.011913597
[43,]  5 0.001  0.1700935 0.012625473
[44,] 10 0.005  0.1635514 0.016521186
[45,] 15 0.010  0.1598131 0.007678354
[46,] 20 0.020  0.1616822 0.015708165
[47,] 25 0.030  0.1644860 0.011156389
[48,] 30 0.050  0.1579439 0.008359133
[49,]  2 0.100  0.1738318 0.006092713
[50,]  3 0.001  0.1700935 0.009689926
[51,]  5 0.005  0.1644860 0.021515634
[52,] 10 0.010  0.1579439 0.008988497
[53,] 15 0.020  0.1672897 0.012538699
[54,] 20 0.030  0.1626168 0.011635420
[55,] 25 0.050  0.1635514 0.006608475
[56,] 30 0.100  0.1691589 0.005118902

[1] "CV misclass rate"
```



Best NN architecture considering CV average misclass rate and standard deviation of error estimation

* Number of nodes = 30

* Decay = 0.01

CV misclass rate for this model = 15.04%

Conclusion

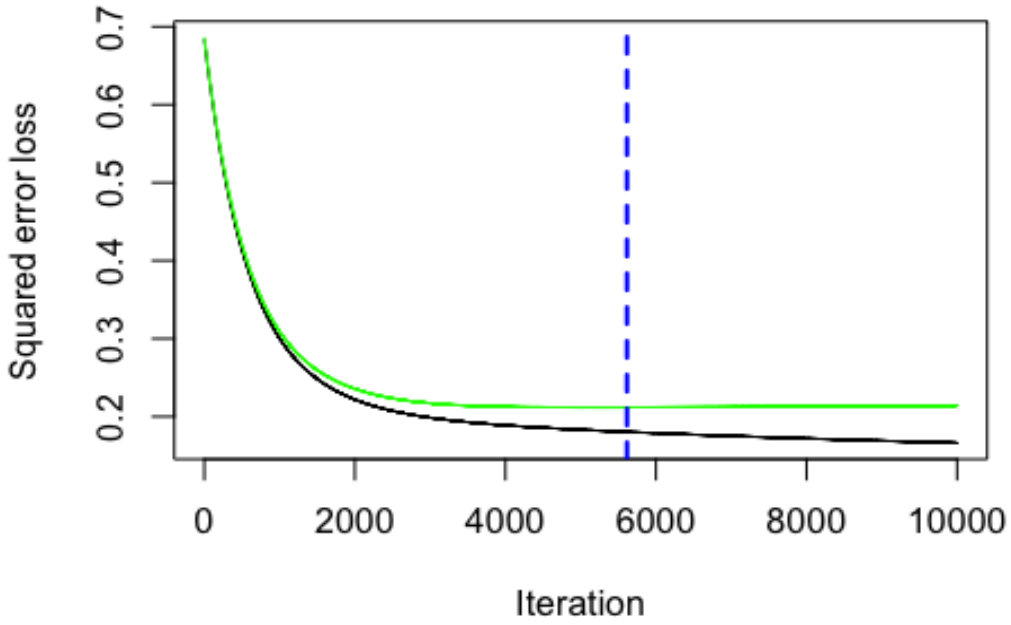
- KNN gives a CV misclassification rate of 15.27% which is only slightly worse than GAM,
- GAM showed a CV misclassification rate of 15.23%
- Neural Network outperforms both marginally with a CV misclassification rate of 15.04%

Question 6: Boosted Trees

Part (a)

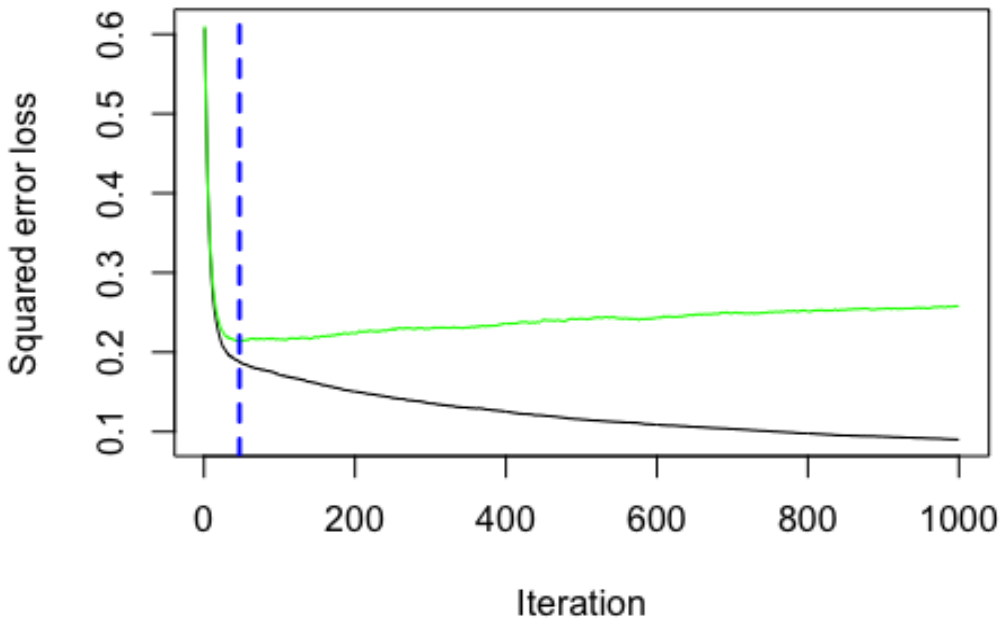
Find the best boosted tree for predicting cost.

```
   [1] "Shrinkage = 0.001"
```



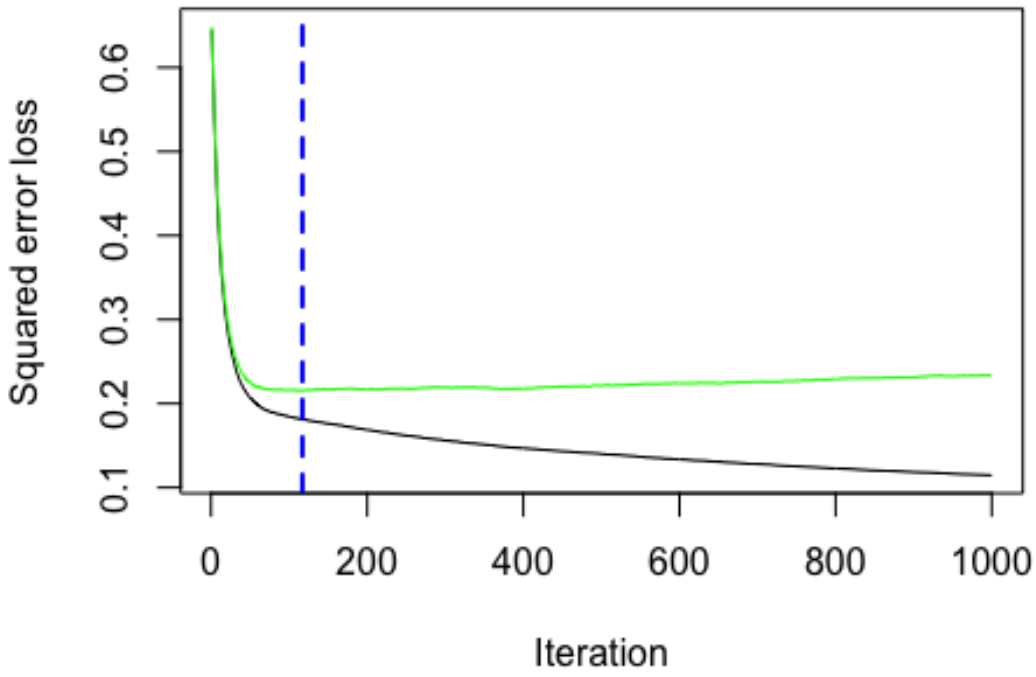
```
   [1] "best tree = 5615"
   [1] "R square = 0.690980"
   [1] "Shrinkage = 0.001, Trees = 5615, CV R-sq = 0.690980"
```

```
[1] "Shrinkage = 0.1"
```



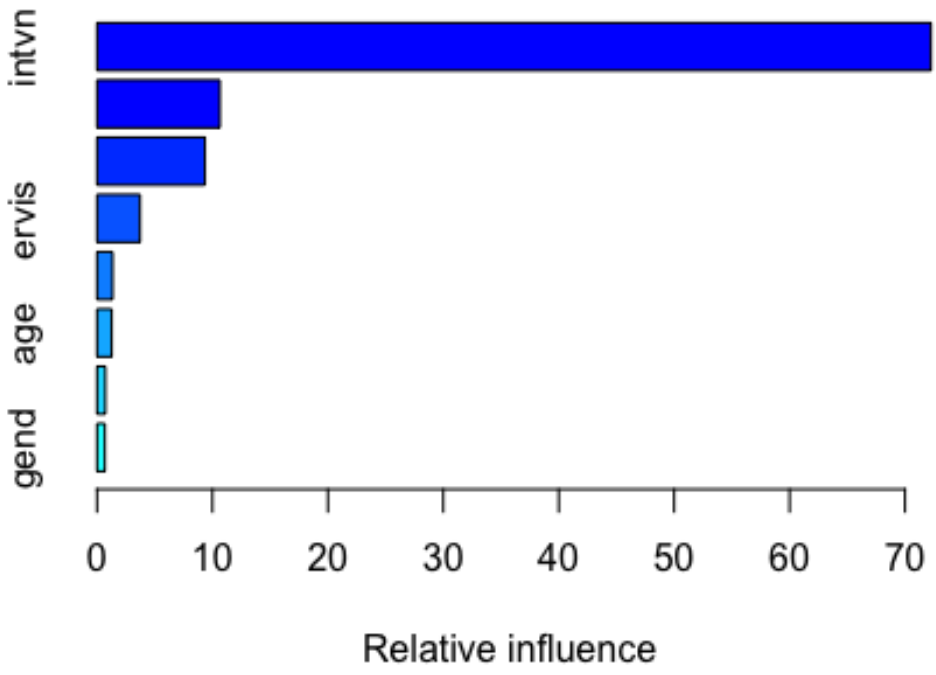

```
   [1] "best tree = 47"
   [1] "R square = 0.688432"
   [1] "Shrinkage = 0.1, Trees = 47, CV R-sq = 0.688432"
```

```
[1] "Shrinkage = 0.05"
```




```
[1] "best tree = 117"
[1] "R square = 0.685960"
[1] "Shrinkage = 0.05, Trees = 117, CV R-sq = 0.685960"
```

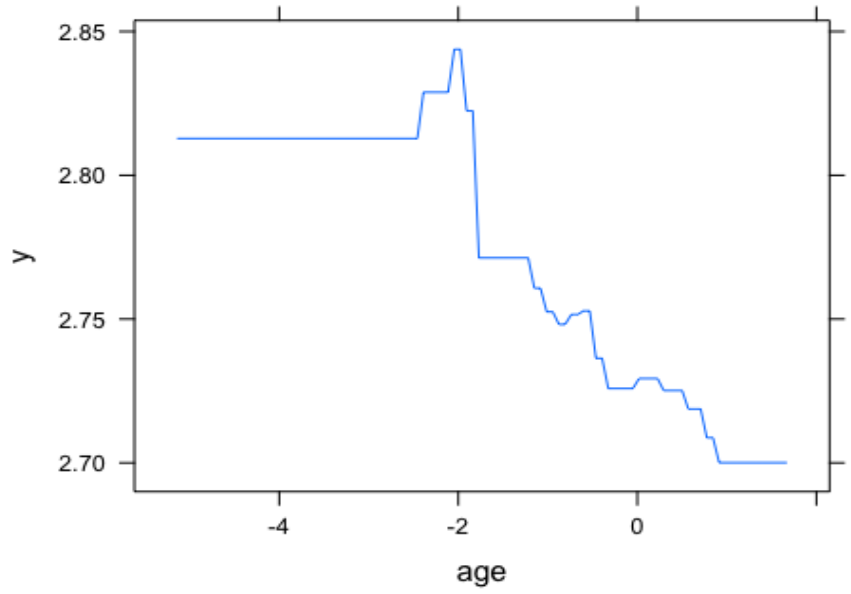
```
[1] "Summary with 117 trees"
```

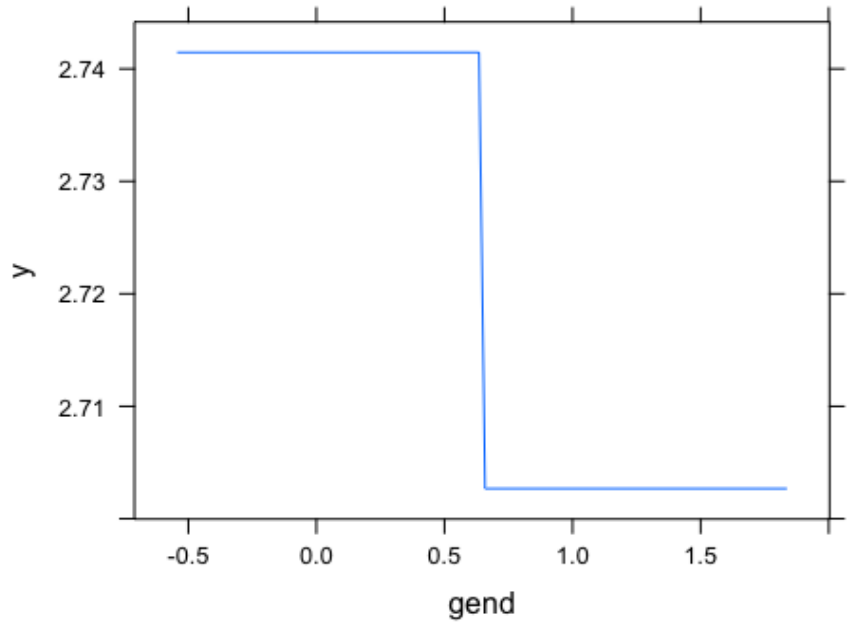


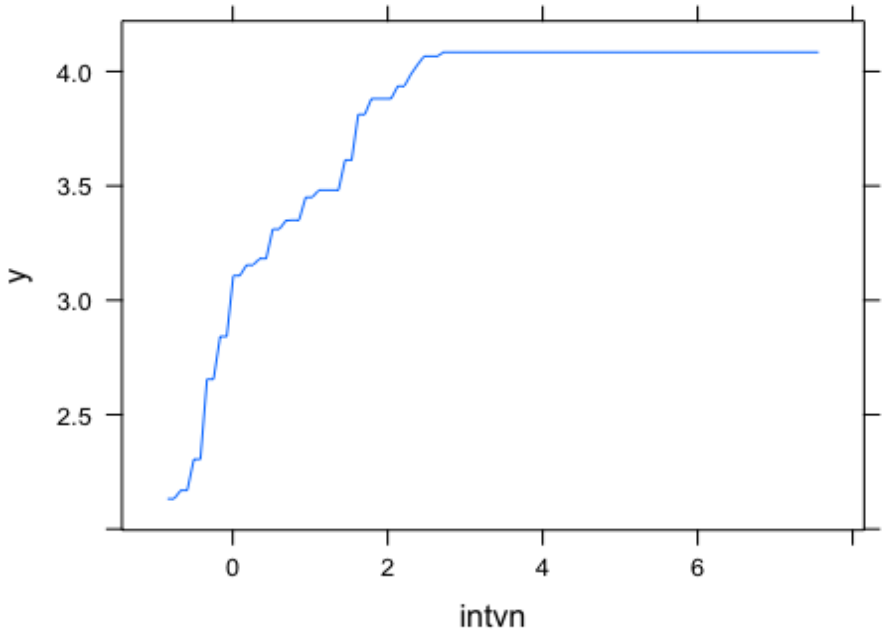

```
            var    rel.inf
intvn     intvn 72.2447611
comorb   comorb 10.6445826
dur         dur  9.3505612
ervis     ervis  3.7265553
comp       comp  1.3503282
age         age  1.2731327
drugs     drugs  0.7258004
gend       gend  0.6842784
```

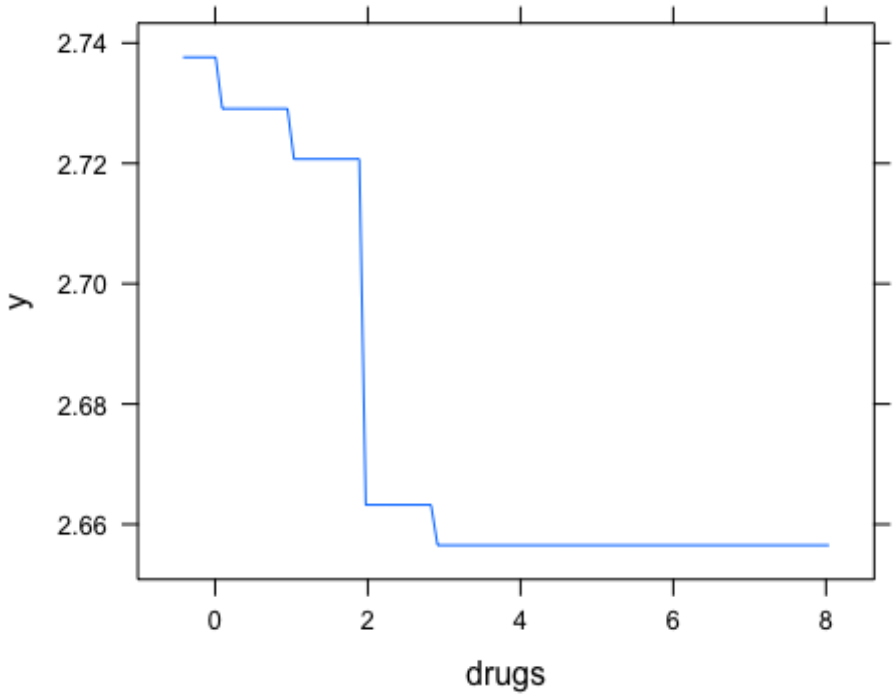
Part (b)

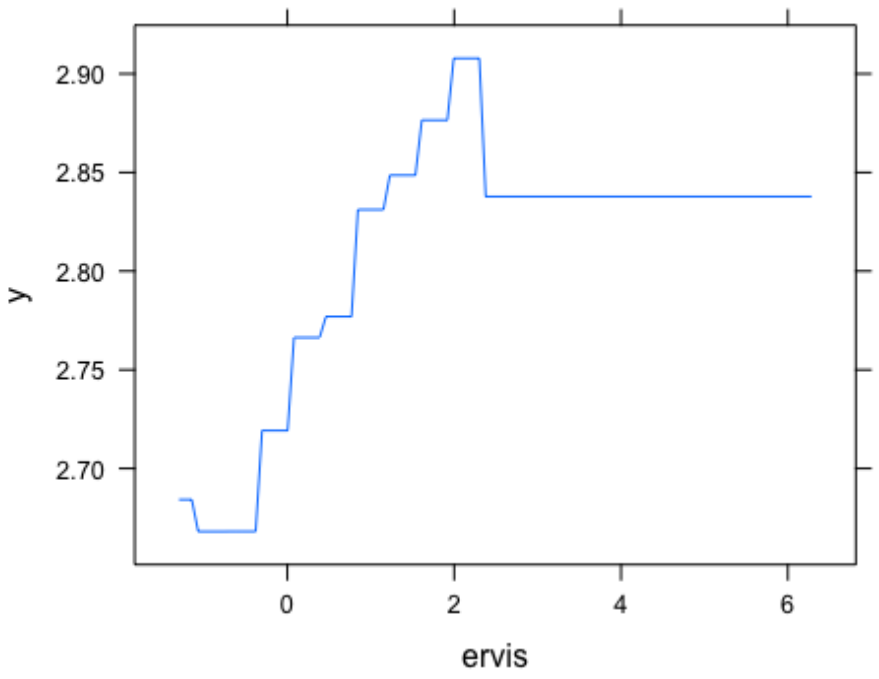
Provide an interpretation of which predictors appear to be the most important and what are their effects. Does this agree with the results from the other methods.

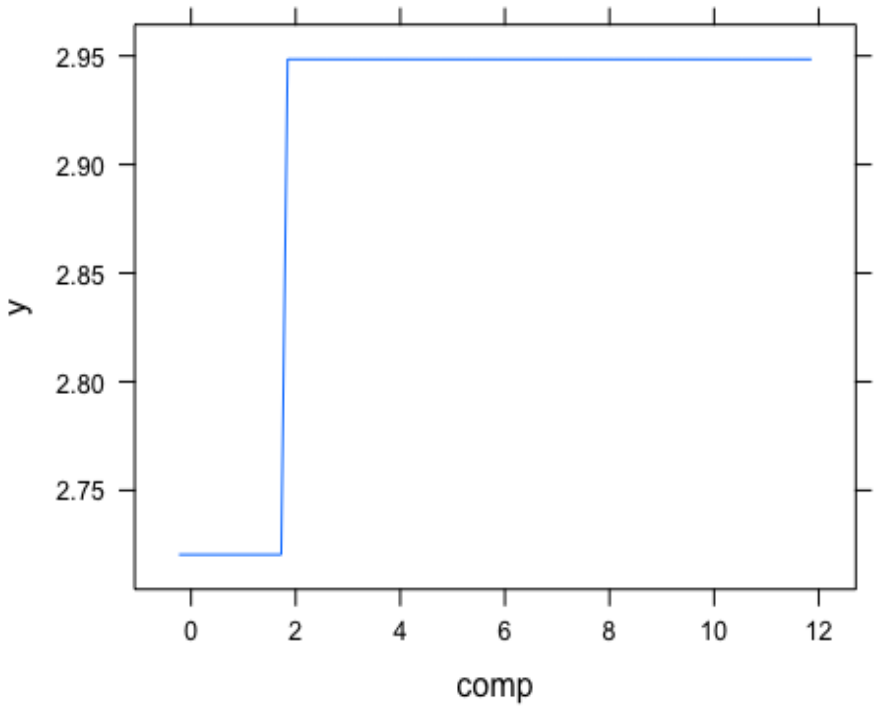


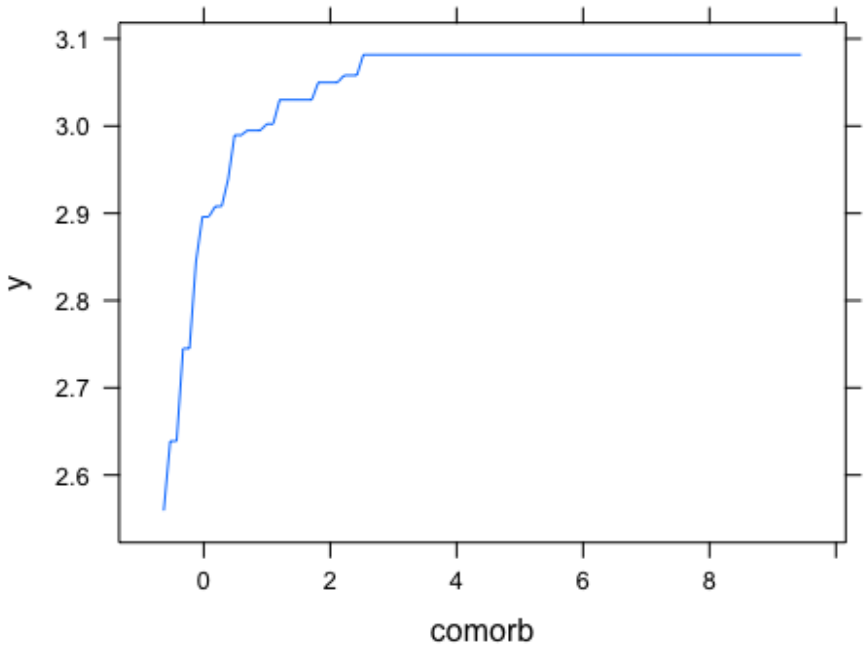


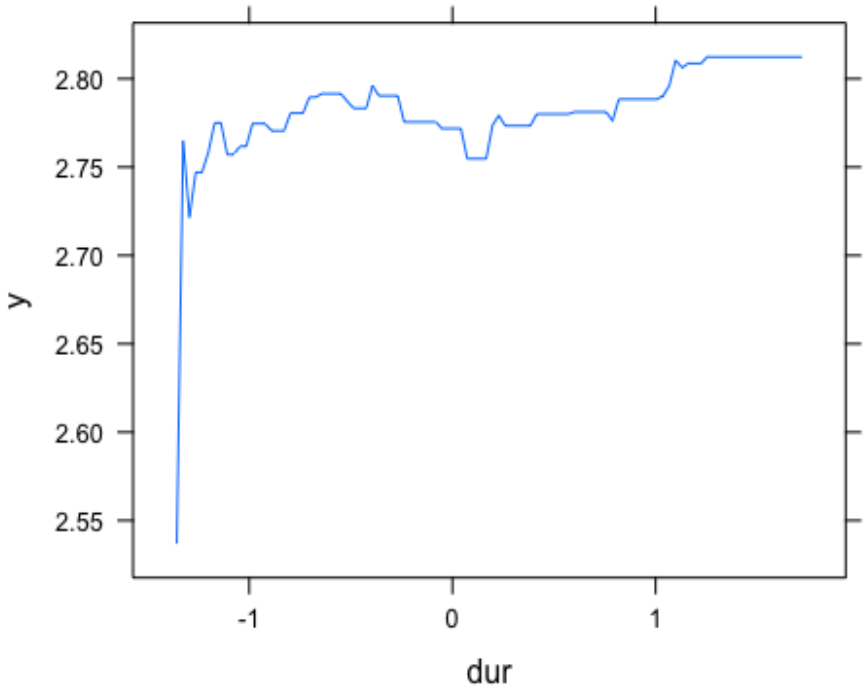

4.0
3.5
3.0
2.5
y
0
2
4
6
intvn


2.74
2.72
2.70
2.68
2.66
y
0
2
4
6
8
drugs


2.90
2.85
2.80
2.75
2.70
y
0
2
4
6
ervis


2.95
2.90
2.85
2.80
2.75
y
0
2
4
6
8
10
12
comp


3.1
3.0
2.9
2.8
2.7
2.6
y
0
2
4
6
8
comorb


2.80
2.75
2.70
2.65
2.60
2.55
y
-1
0
1
dur

Observations:

- intvn is the most significant variable with positive effect on cost
- comorb is the second most important predictor again with positive effect on cost
- duration comes up as the third most significant variable. It's impact increases very sharply at first and then remains stable

This agrees mostly with the GAM model output which had intvn, ervis, comorb and comp as the most influential variables PPR suggested intvn, comorb, comp and ervis are the most significant in that order ervis seems to lose importance in the GBM model

Part (c)

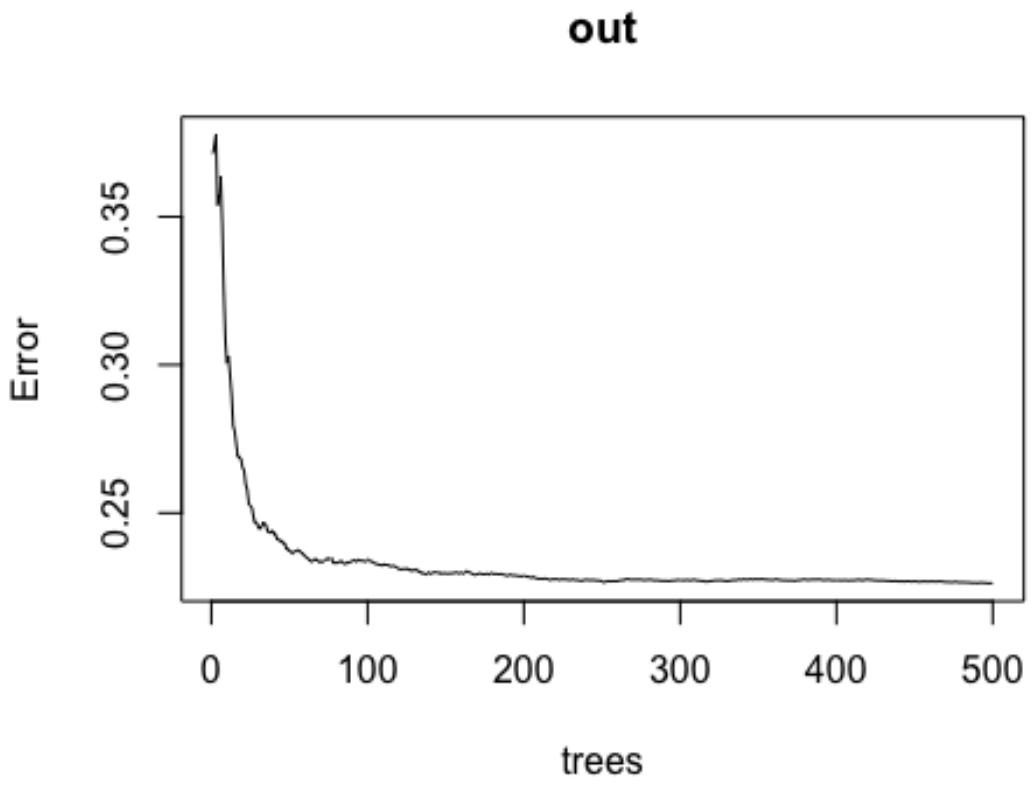
What is the predicted cost for a person with age=59, gend=0, intvn=10, drugs=0, ervis=3, comp=0, comorb=4, and dur=300?

```
[1] "Predicted cost for new data = 3217.921101"
```

Question 7: Random Forest

Part (a)

Fit a random forest model with mtry=3 and ntree = 500.




```
   Call:
    randomForest(formula = target ~ ., data = heart, mtry = 3, ntree = 500,
importance = TRUE)
                  Type of random forest: regression
                        Number of trees: 500
   No. of variables tried at each split: 3

              Mean of squared residuals: 0.2262699
                        % Var explained: 66.95
```

OOB R-square obtained from repeatedly fitting random forest - 67.28, 66.66, 66.99, 66.97, 67.14

The OOB R-square is ~67%.

We see this variation in R-square values due to,

- sampling variance of the bootstrapped samples

- randomly choosing 3 predictors for each split

Part (b)

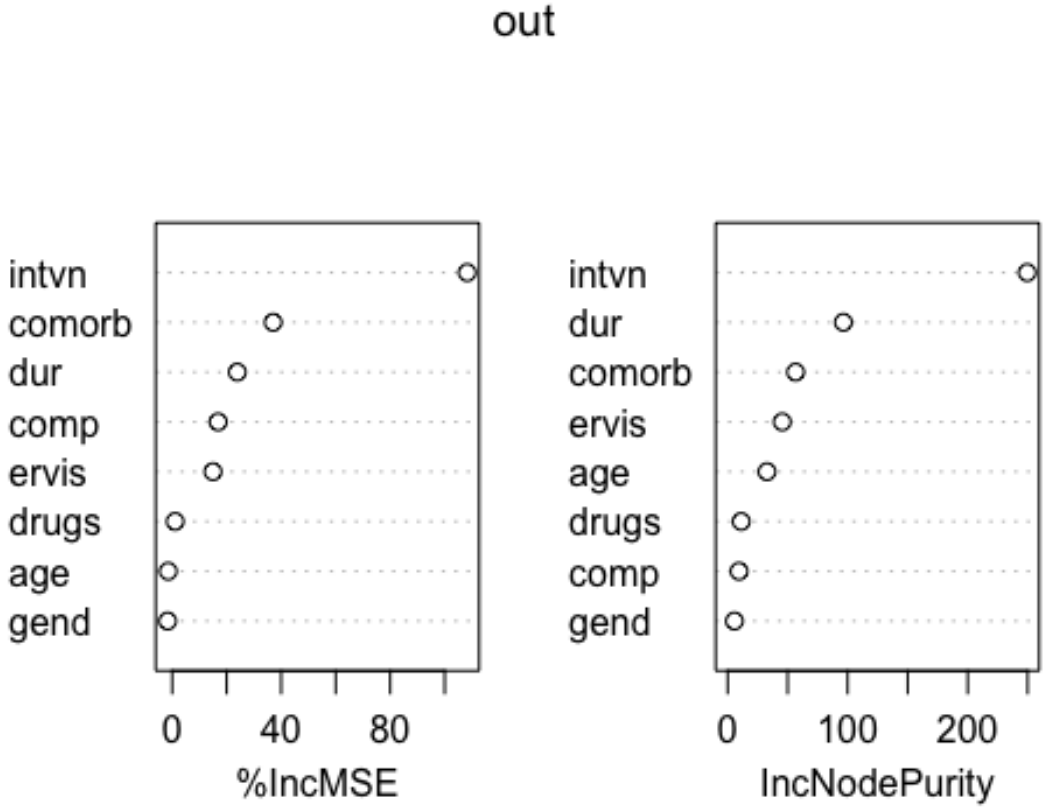
ntree (500) was chosen large enough as the OOB MSE stabilizes after around 200 trees, thereby showing that no further decrease in error could be achieved by increasing ntree.

If the OOB MSE still showed a decreasing trend at the ntree value, we would have to increase ntree further

Part (c)

Based on the numerical variable importance measure produced by the importance.randomForest function, what are the most important variables? Does this agree with what the boosted tree says is the most important?

```
            %IncMSE IncNodePurity
age      -1.450787      32.77620
gend     -1.657730       5.65742
intvn   108.253161     249.60831
drugs     1.087556      11.32377
ervis    14.970618      45.48031
comp     16.855782       9.52505
comorb   37.061376      56.62791
dur      23.910864      96.44062
```

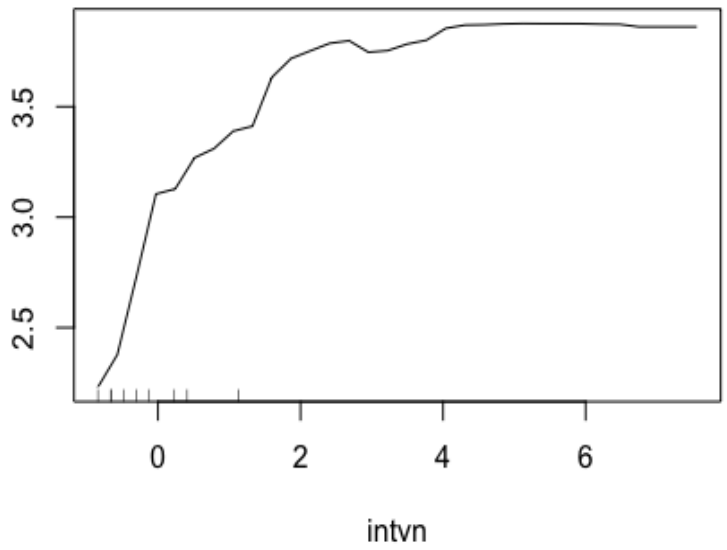


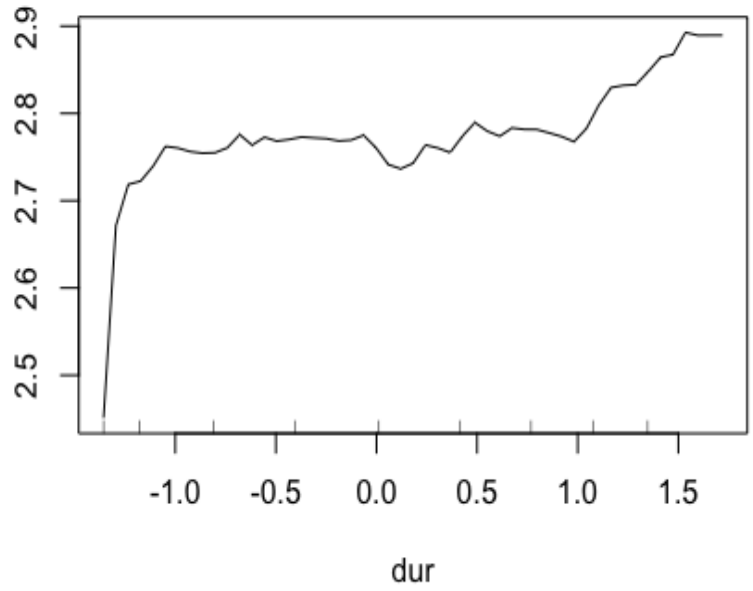
- Most important predictors for Random Forest - intvn, dur, comorb, ervis
- Most significant predictors for GBM - intvn, comorb, dur, ervis

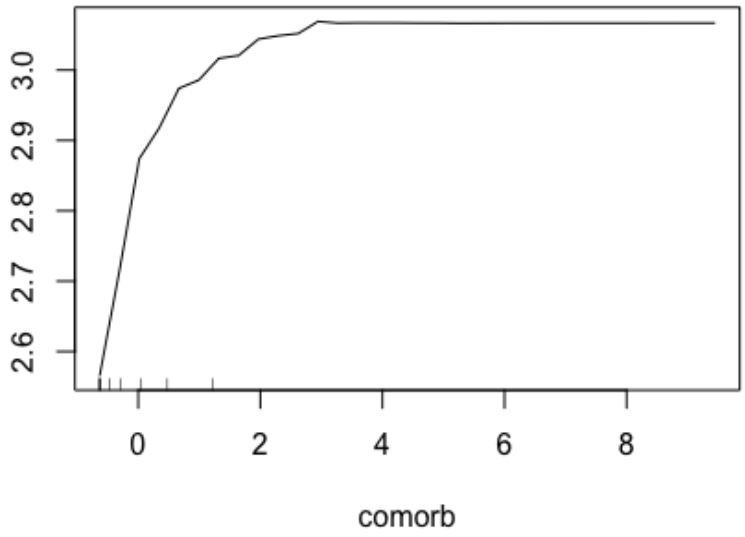
intvn always comes up as the most significant variables, followed by comorb, ervis and duration with some shuffling in the order or importance

Part (d)

Use the partialPlot.randomForest function to produce marginal plots (aka partial dependence plots) for each of the three predictors. Interpret the plots, in terms of whether the predictors have positive, negative, nonlinear, linear, etc. effects on the response.


2.5
3.0
3.5
0
2
4
6
intvn


2.5
2.6
2.7
2.8
2.9
-1.0
-0.5
0.0
0.5
1.0
1.5
dur



Observations

- intvn - fairly linear with positive impact where data is mostly concentrated, flatens out after
- dur - shows some nonlinearity, with a sharp increase in the beginning, followed by a plateau and then a sharp jump at the end. Overall positive impact
- comorb - linear for majority of the data, positive impact, flattens out after

The trends seen here are the same as that seen for the boosted tree partial dependence plots

Part (e)

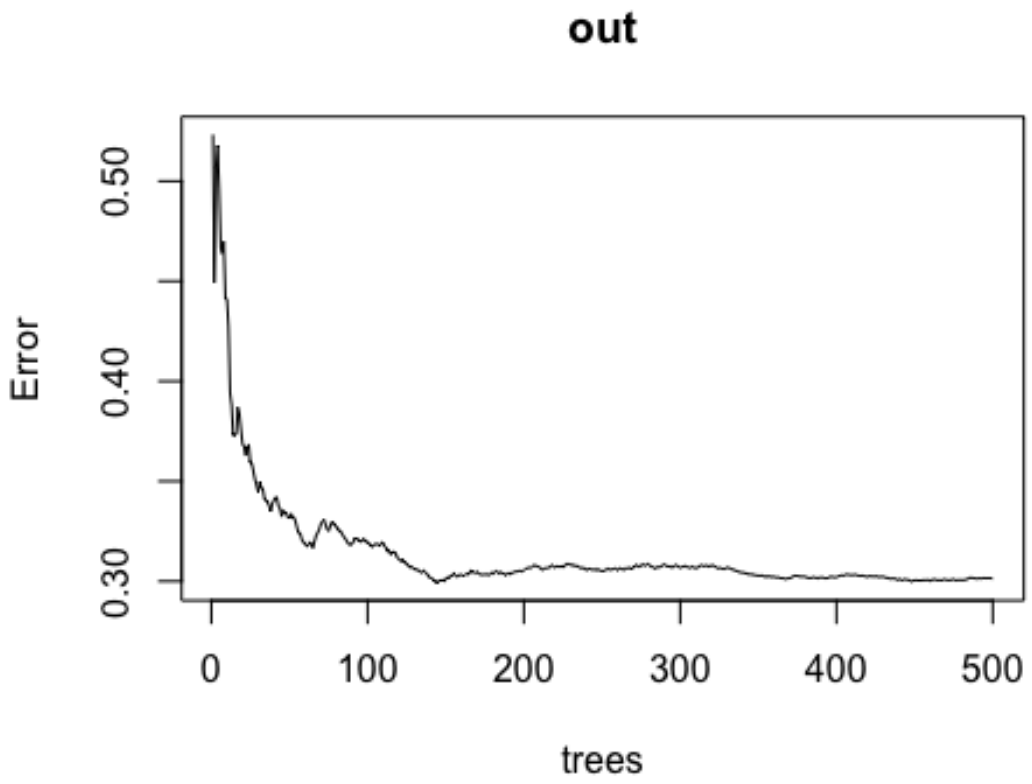
What is the predicted response for a person with age=59, gend=0, intvn=10, drugs=0, ervis=3, comp=0, comorb=4, and dur=300?

```
[1] "Predicted cost for new data with Random Forest = 3517.530599"
```

Part (f)

Repeat part (a) but for mtry=1 and mtry=2. Which of the three random forests (for the three different mtry values) is the best model in terms of predictive performance? You can use the OOB r2 as the measure of predictive performance. Explain what mtry is

```
[1] "mtry=1, ntree=500"
```

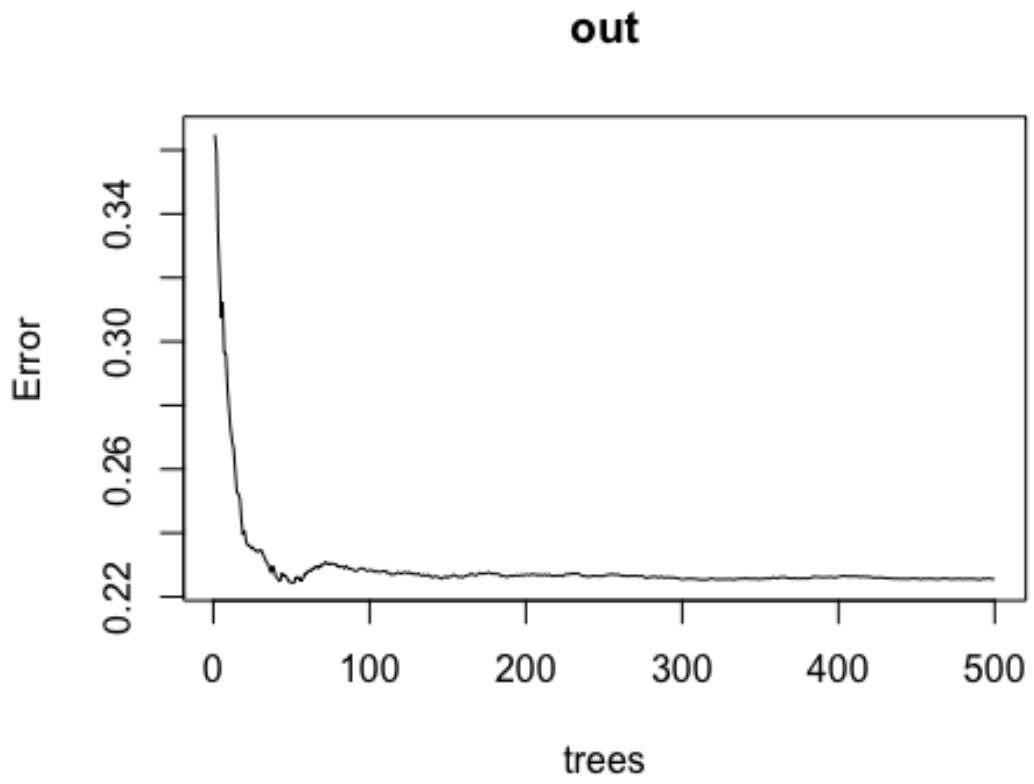



```
   Call:
    randomForest(formula = target ~ ., data = heart, mtry = 1, ntree = 500,
importance = TRUE)
                  Type of random forest: regression
                        Number of trees: 500
   No. of variables tried at each split: 1

              Mean of squared residuals: 0.3013454
                        % Var explained: 55.98

   [1] "OOB R-square for mtry=1 : 55.980000"

   [1] "mtry=2, ntree=500"
```




```
   Call:
    randomForest(formula = target ~ ., data = heart, mtry = 2, ntree = 500,
importance = TRUE)
                  Type of random forest: regression
                        Number of trees: 500
   No. of variables tried at each split: 2

              Mean of squared residuals: 0.2255524
                    % Var explained: 67.05

   [1] "OOB R-square for mtry=1 : 67.050000"
```

Observations:

mtry = 3 gives the best model in terms of OOB R-square, though it is only marginally better than mtry=2.

mtry=1 gives the worst model

mtry tells RandomForest the number of predictors to randomly pick at each split

This ensures some variablity among the different trees fit to the bootstrapped samples Since there are only 8 predictors in the data, letting the algorithm choose mtry=2 (25% of variables) and mtry=3 (37.5%) of the variables for each split was giving comparable results

Part (g)

Out of all the models that you fit in Problems 1-4 and 6-7, and also the tree and neural network model that you fit in HW2, which model appears to be the best for predicting cost? Explain.

The models fitted so far and the cross-validation R-square obtained for each are as follows:
1. 8-NN : 0.5868

2. GAM : 0.6677

3. loess : 0.6793

4. PPR : 0.6809

5. GBM : 0.6914

6. Random Forest : 0.672

7. Decision Tree : 0.6539

8. Neural Network : 0.5919

Hence judging from the cross validation R-square values, GBM gives the best model for predicting cost

Appendix

This section is to be used for including your R code. The following lines of code will take care of it. Please make sure to comment your code appropriately - in particular, demarcating codes belonging to different questions. Among other things, it will be easier for you to debug your own code.

```
# Loading R packages
library(readxl)
library(yaImpute)
library(nnet)
library(mgcv)
library(gbm)
library(randomForest)

# Function to create k-folds for cross validation
CVInd <- function(n,K) {  #n is sample size; K is number of parts; returns K-length list of indices for each part
    m<-floor(n/K)  #approximate size of each part
    r<-n-m*K
    I<-sample(n,n)  #random reordering of the indices
    Ind<-list()  #will be list of indices for all K parts
    length(Ind)<-K
    for (k in 1:K) {
        if (k <= r) kpart <- ((m+1)*(k-1)+1):((m+1)*k)
        else kpart<-((m+1)*r+m*(k-r-1)+1):((m+1)*r+m*(k-r))
        Ind[[k]] <- I[kpart]  #indices for kth part of data
    }
    Ind
}

############ QUESTION 1 begins here ###################
# loading data for question 1
heart <- read.csv("/Users/shreyashiganguly/Documents/Northwestern_MSiA/Winter 2020/Predictive Analytics II/HW3/Ischemic.csv", header = TRUE)
heart$target <- log10(heart$cost)
heart <- heart[,-1]
heart[1:8]<-sapply(heart[1:8], function(x) (x-mean(x))/sd(x)) #standardize predictors
set.seed(12345)
Nrep<-1 #since doing n-fold CV
n.models = 19 #number of different models to fit
n=nrow(heart)
K<-n  #n-fold CV on each replicate
y<-heart$target
yhat=matrix(0,n,n.models)
MSE<-matrix(0,Nrep,n.models)
SD <- matrix(0,Nrep,n.models)
```

```r
for (j in 1:Nrep) {
    Ind<-CVInd(n,K)
    for (k in 1:K) {
        train<-as.matrix(heart[-Ind[[k]],1:8])
        test<-as.matrix(heart[Ind[[k]],1:8])
        ytrain<-as.array(heart[-Ind[[k]],9])
        nn = 20
        out<-ann(train,test,nn,verbose=F)
        ind<-matrix(out$knnIndexDist[,1:nn],nrow=1,ncol=20)
        counter = 1
        for (i in 2:20) {
            yhat[Ind[[k]],counter]<-mean(ytrain[ind[,1:i]])
            counter = counter + 1
        }
    } #end of k loop
    MSE[j,]=apply(yhat,2,function(x) sum((y-x)^2))/n
    SD[j,]=apply(yhat,2,function(x) sd(y-x))
} #end of j loop
MSEAve<- apply(MSE,2,mean) #averaged mean square CV error
r2<-1-MSEAve/var(y)  #CV r^2
try_K <- c(2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20)
CV_stats <- cbind(try_K,r2,MSEAve)
print("n-fold Cross Validation results:")
print(CV_stats)
sprintf("CV estimate of the prediction error standard deviation for K=8 is
%f",SD[1,7])
train<-as.matrix(heart[,1:8])
ytrain<-heart[,9]
new_data <- as.matrix(cbind(age=(59-58.72)/6.754118,
                            gend=(0-0.2284)/0.4200854,
                            intvn=(10-4.707)/5.594661,
                            drugs=(0-0.4467)/1.063882,
                            ervis=(3-3.425)/2.637474,
                            comp=(0-0.05711)/0.248068,
                            comorb=(4-3.767)/5.95099,
                            dur=(300-164.03)/120.9159))
out<-ann(train,new_data,8)
ind<-as.matrix(out$knnIndexDist[,1:8])
predited_value <- mean(ytrain[ind])
sprintf("Predicted cost for the new data = %f",10^(predited_value))
############ QUESTION 2 begins here ###################
out<-
gam(target~s(age)+gend+s(intvn)+s(drugs,k=9)+s(ervis)+s(comp,k=3)+s(comorb)+s
(dur),
         data=heart,
         family=gaussian(),
         sp=c(-1,-1,-1,-1,-1,-1,-1,-1))
print("Summary of GAM model:")
summary(out)
print("estimated smoothing parameters for each constituent function:")
```

```r
out$sp  ##estimated smoothing parameters for each constituent function

print("Component function plots:")
par(mfrow=c(2,4))
plot(out)
Nrep<-20 #number of replicates
n.models = 1 #number of different models to fit
n=nrow(heart)
K<-10  #10-fold CV on each replicate
y<-heart$target
yhat=matrix(0,n,n.models)
MSE<-matrix(0,Nrep,n.models)
SD <- matrix(0,Nrep,n.models)

for (j in 1:Nrep) {
    Ind<-CVInd(n,K)
    for (k in 1:K) {
        #using sp from model trained on entire training data
        out<-
gam(target~s(age)+gend+s(intvn)+s(drugs,k=7)+s(ervis)+comp+s(comorb)+s(dur),
                data=heart[-Ind[[k]],],
                family=gaussian(),
                sp=c(7.113768e+09,1.039629e-01,1.732089e+08,9.001982e-
02,1.570520e-02,3.975592e-02,7.486655e-02))
        yhat[Ind[[k]],1]<-as.numeric(predict(out,heart[Ind[[k]],]))
    } #end of k loop
    MSE[j,]=apply(yhat,2,function(x) sum((y-x)^2))/n
    SD[j,]=apply(yhat,2,function(x) sd(y-x))
} #end of j loop

MSEAve<- apply(MSE,2,mean) #averaged mean square CV error
MSEsd <- apply(MSE,2,sd) #SD of mean square CV error
r2<-1-MSEAve/var(y) #CV r^2

CV_stats <- cbind(r2,MSEAve,MSEsd)
print("n-fold Cross Validation results:")
print(CV_stats)

sprintf("CV estimate of the prediction error standard deviation for GAM is
%f",mean(SD))

new_data <- as.data.frame(cbind(age=(59-58.72)/6.754118,
                                gend=(0-0.2284)/0.4200854,
                                intvn=(10-4.707)/5.594661,
                                drugs=(0-0.4467)/1.063882,
                                ervis=(3-3.425)/2.637474,
                                comp=(0-0.05711)/0.248068,
                                comorb=(4-3.767)/5.95099,
                                dur=(300-164.03)/120.9159))
```

```r
sprintf("Predicted cost for new data = %f",10^predict(out,new_data))
########### QUESTION 3 begins here ###################
heart1 <- heart[,c(3,5,7,8,9)]
Nrep<-10 #number of replicates
n.models = 27 #number of different models to fit
n=nrow(heart)
K<-10 #10-fold CV on each replicate
y<-heart1$target
yhat=matrix(0,n,n.models)
MSE<-matrix(0,Nrep,n.models)
SD <- matrix(0,Nrep,n.models)

for (j in 1:Nrep) {
    Ind<-CVInd(n,K)
    for (k in 1:K) {
        c = 1
        for (d in 0:2) {
            for (s in seq(from=0.1, to=0.9, by=0.1)) {
                out <- loess(target~.,
                             heart1[-Ind[[k]],],
                             degree=d,
                             span=s,
                             control=loess.control(surface="direct"))
                print(summary(out))
                yhat[Ind[[k]],c]<-as.numeric(predict(out,heart1[Ind[[k]],]))
                c = c+1
            }
        }
    } #end of k loop
    MSE[j,]=apply(yhat,2,function(x) sum((y-x)^2))/n
    SD[j,]=apply(yhat,2,function(x) sd(y-x))
} #end of j loop

MSEAve<- apply(MSE,2,mean) #averaged mean square CV error
MSEsd <- apply(MSE,2,sd)  #SD of mean square CV error
model <-
c("0,0.1","0,0.2","0,0.3","0,0.4","0,0.5","0,0.6","0,0.7","0,0.8","0,0.9",

"1,0.1","1,0.2","1,0.3","1,0.4","1,0.5","1,0.6","1,0.7","1,0.8","1,0.9",

"2,0.1","2,0.2","2,0.3","2,0.4","2,0.5","2,0.6","2,0.7","2,0.8","2,0.9")

r2<-1-MSEAve/var(y) #CV r^2

CV_stats <- cbind(model,r2,MSEAve,MSEsd)

print("n-fold Cross Validation results:")
CV_stats
```

```r
#degree = 0
sigma <- data.frame()

for (lambda in seq(from=0.1, to=0.9, by=0.1)) {
    out<-loess(target~., heart1,degree=0, span=lambda);
    sigma <- rbind(sigma,c(0,lambda,out$s))
}

#degree = 1
for (lambda in seq(from=0.1, to=0.9, by=0.1)) {
    out<-loess(target~., heart1,degree=1, span=lambda);
    sigma <- rbind(sigma,c(1,lambda,out$s))
}

#degree = 2
for (lambda in seq(from=0.1, to=0.9, by=0.1)) {
    out<-loess(target~., heart1,degree=2, span=lambda);
    sigma <- rbind(sigma,c(2,lambda,out$s))
}
colnames(sigma) <- c('degree','span','sigma')
print("Sigma:")
print(sigma)

#low bias estimate of sigma = 0.4671513
sig_hat<-0.4671513

CP = data.frame()
for (lambda in seq(from=0.1, to=0.9, by=0.1)) {
    out<-loess(target~., heart1,degree=0, span=lambda);
    SSE<-sum((heart1[,5]-out$fitted)^2);
    Cp <- (SSE+2*out$trace.hat*sig_hat^2)/nrow(heart1);
    CP <- rbind(CP,c(0,lambda,Cp))
}

for (lambda in seq(from=0.1, to=0.9, by=0.1)) {
    out<-loess(target~., heart1,degree=1, span=lambda);
    SSE<-sum((heart1[,5]-out$fitted)^2);
    Cp <- (SSE+2*out$trace.hat*sig_hat^2)/nrow(heart1);
    CP <- rbind(CP,c(1,lambda,Cp))
}

for (lambda in seq(from=0.1, to=0.9, by=0.1)) {
    out<-loess(target~., heart1,degree=2, span=lambda);
    SSE<-sum((heart1[,5]-out$fitted)^2);
    Cp <- (SSE+2*out$trace.hat*sig_hat^2)/nrow(heart1);
    CP <- rbind(CP,c(2,lambda,Cp))
}

colnames(CP) <- c('degree','span','Cp')
```

```r
print("Cp results:")
print(CP)
sprintf("CV estimate of the prediction error standard deviation for LOESS is
%f",mean(SD[,12]))
#using parameters from CV: degree=1, span=0.3
out <- loess(target~.,
             heart1,
             degree=1,
             span=0.3,
             control=loess.control(surface="direct"))
sprintf("Predicted cost for new data from best model from CV =
%f",10^predict(out,new_data))

#using parameters from Cp: degree=1, span=0.5
out <- loess(target~.,
             heart1,
             degree=1,
             span=0.5,
             control=loess.control(surface="direct"))
sprintf("Predicted cost for new data from best model from Cp =
%f",10^predict(out,new_data))
############ QUESTION 4 begins here ###################
Nrep<-5 #number of replicates of CV
K<-10  #K-fold CV on each replicate
n.models = 20 #number of different models to fit
n=nrow(heart)
y<-heart$target
yhat=matrix(0,n,n.models)
MSE<-matrix(0,Nrep,n.models)
SD <- matrix(0,Nrep,n.models)

for (j in 1:Nrep) {
    Ind<-CVInd(n,K)
    for (k in 1:K) {
        c = 1
        for (d in c(1,2,3,4,5,7,10,15,20,25,30,35,40)) {
            out <- ppr(target~.,
                       data=heart[-Ind[[k]],],
                       nterms=d)
            yhat[Ind[[k]],c]<-as.numeric(predict(out,heart[Ind[[k]],]))
            c = c+1
        }
    } #end of k loop
    MSE[j,]=apply(yhat,2,function(x) sum((y-x)^2))/n
    SD[j,]=apply(yhat,2,function(x) sd(y-x))
} #end of j loop
MSEAve<- apply(MSE,2,mean) #averaged mean square CV error
MSEsd <- apply(MSE,2,sd)  #SD of mean square CV error
r2<-1-MSEAve/var(y) #CV r^2
CV_stats <-
```

```
cbind(c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,20,25,30,35,40),r2,MSEAve,MSEsd)

print("CV Stats")
CV_stats
out<-ppr(target~.,
         data=heart,
         nterms=1)
summary(out)

sprintf("CV estimate of the prediction error standard deviation for PPR is
%f",mean(SD[,1]))
sprintf("Predicted cost for new data = %f",10^predict(out,new_data))
########### QUESTION 5 begins here ###################
fgl <- read.csv("/Users/shreyashiganguly/Documents/Northwestern_MSiA/Winter
2020/Predictive Analytics II/HW3/FGL.csv")
unique(fgl$type)
fgl$Win <- ifelse(fgl$type=="WinF",1,
                  ifelse(fgl$type=="WinNF",1,0))

fgl <- fgl[,-10] #dropping type
fgl[1:9]<-sapply(fgl[1:9], function(x) (x-mean(x))/sd(x)) #standardize
predictors
set.seed(12345)
Nrep<-50 #CV replicates
n.models = 19 #number of different models to fit
n=nrow(fgl)
K<-10  #10-fold CV on each replicate
y<-fgl$Win
yhat=matrix(0,n,n.models)
CV.rate<-matrix(0,Nrep,n.models)

for (j in 1:Nrep) {
    Ind<-CVInd(n,K)
    for (k in 1:K) {
        train<-as.matrix(fgl[-Ind[[k]],1:9])
        test<-as.matrix(fgl[Ind[[k]],1:9])
        ytrain<-as.array(fgl[-Ind[[k]],10])
        nn = 20
        out<-ann(train,test,nn)
        ind<-as.matrix(out$knnIndexDist[,1:nn])
        counter = 1
        for (i in 2:20) {
            yhat[Ind[[k]],counter]<-ifelse(apply(ind[,1:i],1,
                                                 function(x)
sum(ytrain[x]==1)/length(ytrain[x]))>0.5,1,0)
            counter = counter + 1
        }
    } #end of k loop
    CV.rate[j,]=apply(yhat,2,function(x) sum(y != x)/n)
```

```
} #end of j loop
CV.rateAve<- apply(CV.rate,2,mean) #averaged CV misclass rate
CV.rateSD <- apply(CV.rate,2,sd)  #SD of CV misclass rate
try_K <- seq(from=2, to=20, by=1)
CV_stats <- cbind(try_K,CV.rateAve,CV.rateSD)

print("n-fold Cross Validation results:")
print(CV_stats)
#On whole training
out<-gam(Win~s(RI)+s(Na)+s(Mg)+s(Al)+s(Si)+s(K)+s(Ca)+s(Ba)+s(Fe),
         data=fgl,
         family=binomial("logit"),
         sp=c(-1,-1,-1,-1,-1,-1,-1,-1,-1))
print("Smoothing parameters from whole training data:")
print(out$sp)

#Using sp from whole training data in CV
Nrep<-10 #CV replicates
n.models = 1 #number of different models to fit
n=nrow(fgl)
K<-10  #10-fold CV on each replicate
y<-fgl$Win
yhat=matrix(0,n,n.models)
CV.rate<-matrix(0,Nrep,n.models)

for (j in 1:Nrep) {
    Ind<-CVInd(n,K)
    for (k in 1:K) {
        out<-gam(Win~s(RI)+s(Na)+s(Mg)+s(Al)+s(Si)+s(K)+s(Ca)+s(Ba)+s(Fe),
                 data=fgl[-Ind[[k]],],
                 family=binomial("logit"),
                 sp=c(2.559800e-01,4.645713e-04,1.196338e+04,4.322553e-
04,6.293638e+03,1.642143e-04,
                      1.027133e-06,9.754248e-03,2.942407e-02))
        yhat[Ind[[k]],1]<-
as.matrix(ifelse(as.numeric(predict(out,fgl[Ind[[k]],],type='response'))>0.5,
1,0))
    } #end of k loop
    CV.rate[j,]=apply(yhat,2,function(x) sum(y != x)/n)
} #end of j loop

CV.rateAve<- apply(CV.rate,2,mean) #averaged CV misclass rate
CV.rateSD <- apply(CV.rate,2,sd)   #SD of CV misclass rate
CV_stats <- cbind(CV.rateAve,CV.rateSD)
print("CV stats")
CV_stats
#Fitting best Neural Network
fgl$Win <- as.factor(fgl$Win)
Nrep<-5 #number of replicates of CV
```

```
K<-10  #K-fold CV on each replicate
n.models = 56 #number of different models to fit
n=nrow(fgl)
y<-fgl[,10]
yhat=matrix(0,n,n.models)
CV.rate<-matrix(0,Nrep,n.models)
for (j in 1:Nrep) {
    Ind<-CVInd(n,K)
    for (k in 1:K) {
        counter = 1
        for (size in c(2,3,5,10,15,20,25,30)) {
            for (decay in c(0.001,0.005,0.01,0.02,0.03,0.05,0.1)) {
                out<-nnet(Win~.,fgl[-Ind[[k]],],linout=F,skip=F,
                          size=size,decay=decay,
                          maxit=1000,trace=F)
                yhat[Ind[[k]],counter]<-
predict(out,fgl[Ind[[k]],],type="class")
                counter = counter+1
            }
        }
    } #end of k loop
    CV.rate[j,]=apply(yhat,2,function(x) sum(y != x)/n)
} #end of j loop

CV.rateAve<- apply(CV.rate,2,mean) #averaged CV misclass rate
CV.rateSD <- apply(CV.rate,2,sd)  #SD of CV misclass rate
CV_stats <-
cbind(c(2,3,5,10,15,20,25,30),c(0.001,0.005,0.01,0.02,0.03,0.05,0.1),CV.rateA
ve,CV.rateSD)
print("CV stats:")
CV_stats

print("CV misclass rate")
plot(CV_stats[,3])
########### QUESTION 6 begins here ###################
set.seed(12345)
print("Shrinkage = 0.001")
gbm1 <- gbm(target~., data=heart, var.monotone=rep(0,8),
            distribution="gaussian",
            n.trees=10000,
            shrinkage=0.001,
            interaction.depth=3,
            bag.fraction = .5,
            train.fraction = 1,
            n.minobsinnode = 10,
            cv.folds = 10,
            keep.data=TRUE, verbose=FALSE)
best.iter <- gbm.perf(gbm1,method="cv")
sprintf("best tree = %d",best.iter)
```

```r
r2 <- 1 - gbm1$cv.error[best.iter]/var(heart$target)
sprintf("R square = %f",r2)
sprintf("Shrinkage = 0.001, Trees = %d, CV R-sq = %f",best.iter,r2)

print("Shrinkage = 0.1")
gbm1 <- gbm(target~., data=heart, var.monotone=rep(0,8),
            distribution="gaussian",
            n.trees=1000,
            shrinkage=0.1,
            interaction.depth=3,
            bag.fraction = .5,
            train.fraction = 1,
            n.minobsinnode = 10,
            cv.folds = 10,
            keep.data=TRUE, verbose=FALSE)
best.iter <- gbm.perf(gbm1,method="cv")
sprintf("best tree = %d",best.iter)

r2 <- 1 - gbm1$cv.error[best.iter]/var(heart$target)
sprintf("R square = %f",r2)
sprintf("Shrinkage = 0.1, Trees = %d, CV R-sq = %f",best.iter,r2)

print("Shrinkage = 0.05")
gbm1 <- gbm(target~., data=heart, var.monotone=rep(0,8),
            distribution="gaussian",
            n.trees=1000,
            shrinkage=0.05,
            interaction.depth=3,
            bag.fraction = .5,
            train.fraction = 1,
            n.minobsinnode = 10,
            cv.folds = 10,
            keep.data=TRUE, verbose=FALSE)
best.iter <- gbm.perf(gbm1,method="cv")
sprintf("best tree = %d",best.iter)
r2 <- 1 - gbm1$cv.error[best.iter]/var(heart$target)
sprintf("R square = %f",r2)
sprintf("Shrinkage = 0.05, Trees = %d, CV R-sq = %f",best.iter,r2)

sprintf("Summary with %d trees",best.iter)
summary(gbm1,n.trees=best.iter)
#Marginal Plots
plot(gbm1, i.var = 1, n.trees = best.iter) #Age
plot(gbm1, i.var = 2, n.trees = best.iter) #Gender
plot(gbm1, i.var = 3, n.trees = best.iter) #Intvn
plot(gbm1, i.var = 4, n.trees = best.iter) #drugs
plot(gbm1, i.var = 5, n.trees = best.iter) #ervis
plot(gbm1, i.var = 6, n.trees = best.iter) #comp
plot(gbm1, i.var = 7, n.trees = best.iter) #comorb
```

```r
plot(gbm1, i.var = 8, n.trees = best.iter) #dur
#On whole training
out <- gbm(target~., data=heart, var.monotone=rep(0,8),
            distribution="gaussian",
            n.trees=113,
            shrinkage=0.05,
            interaction.depth=3,
            bag.fraction = .5,
            train.fraction = 1,
            n.minobsinnode = 10,
            keep.data=TRUE, verbose=FALSE)

new_data <- as.data.frame(cbind(age=(59-58.72)/6.754118,
                                gend=(0-0.2284)/0.4200854,
                                intvn=(10-4.707)/5.594661,
                                drugs=(0-0.4467)/1.063882,
                                ervis=(3-3.425)/2.637474,
                                comp=(0-0.05711)/0.248068,
                                comorb=(4-3.767)/5.95099,
                                dur=(300-164.03)/120.9159))
pred <- predict.gbm(out,new_data,n.trees=89,type='response')
sprintf("Predicted cost for new data = %f",10^pred)
########### QUESTION 7 begins here ###################
set.seed(12345)
out <- randomForest(target~., data=heart,
                    mtry=3, ntree = 500,
                    importance = TRUE)

plot(out)  #plots OOB mse vs # trees
print(out)
importance(out)
varImpPlot(out)
#intvn - fairly linear with positive impact (data concentration)
partialPlot(out,
            pred.data=heart,
            x.var = names(heart)[3], xlab = names(heart)[3], main=NULL)

#dur - shows some nonlinearity, with a sharp increase in the beginning,
followed by a plateau and then a sharp jump at the end
#overall positive impact
partialPlot(out,
            pred.data=heart,
            x.var = names(heart)[8], xlab = names(heart)[8], main=NULL)

#comorb - linear for majority of the data, positive impact (data
concentration)
partialPlot(out,
            pred.data=heart,
            x.var = names(heart)[7], xlab = names(heart)[7], main=NULL)
```

```r
#The plots look fairly similar to the partial dependence plots from the GBM
model

sprintf("Predicted cost for new data with Random Forest =
%f",10^predict(out,new_data))
set.seed(12345)
print("mtry=1, ntree=500")
out <- randomForest(target~., data=heart,
                    mtry=1, ntree = 500,
                    importance = TRUE)

plot(out)  #plots OOB mse vs # trees
print(out)
sprintf("OOB R-square for mtry=1 : %f",55.98)

print("mtry=2, ntree=500")
out <- randomForest(target~., data=heart,
                    mtry=2, ntree = 500,
                    importance = TRUE)

plot(out)  #plots OOB mse vs # trees
print(out)
sprintf("OOB R-square for mtry=1 : %f",67.05)
```