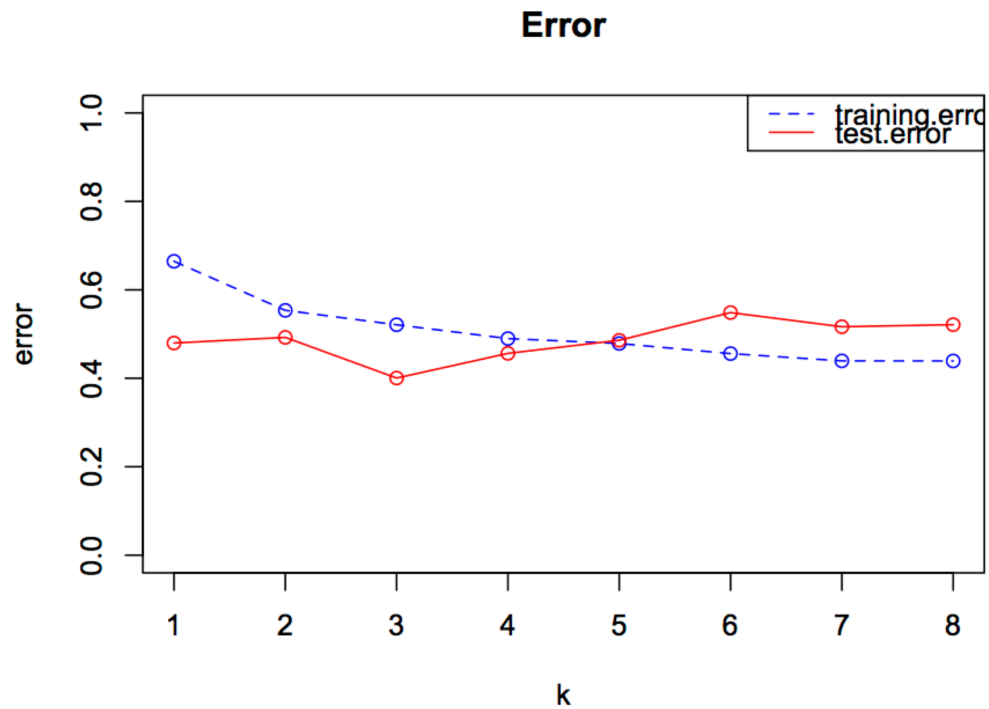
Statistical Data Mining I

Homework 4

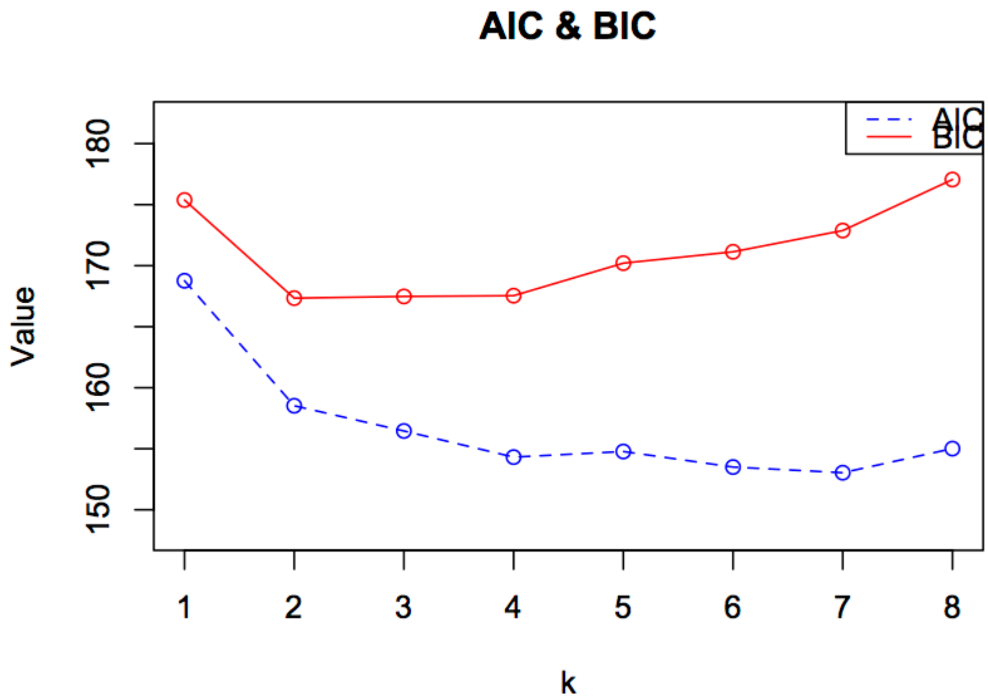
Student Number:71

Problem 1:

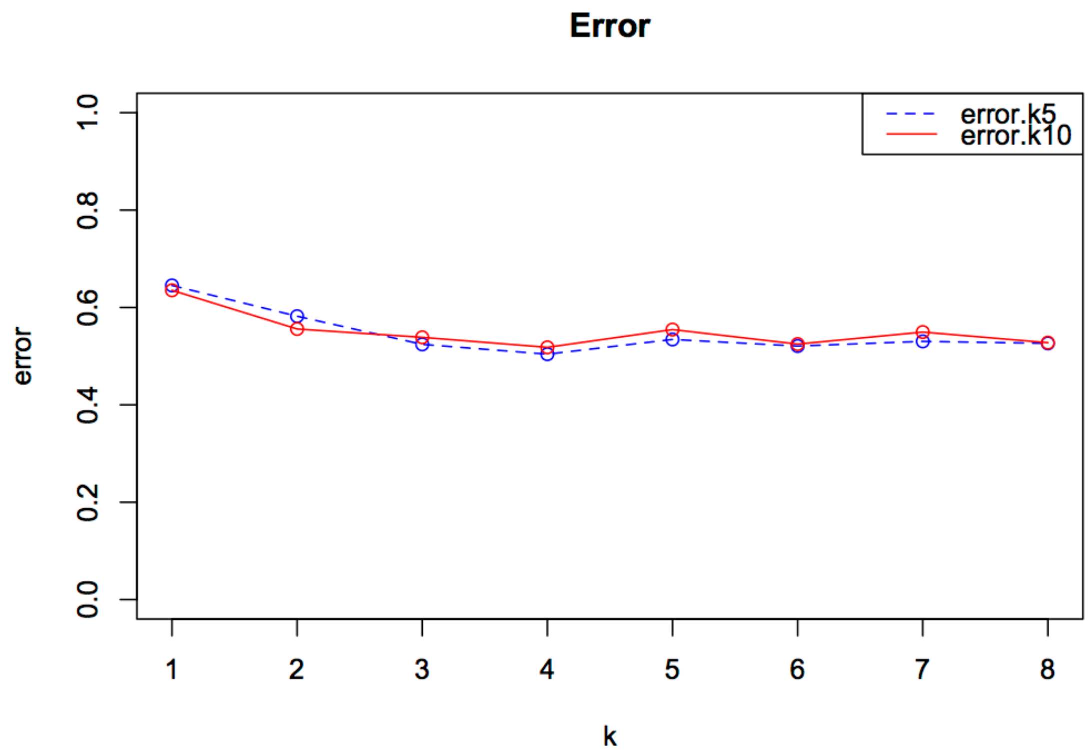
- Carried out best subset linear regression analysis and result plot is below:



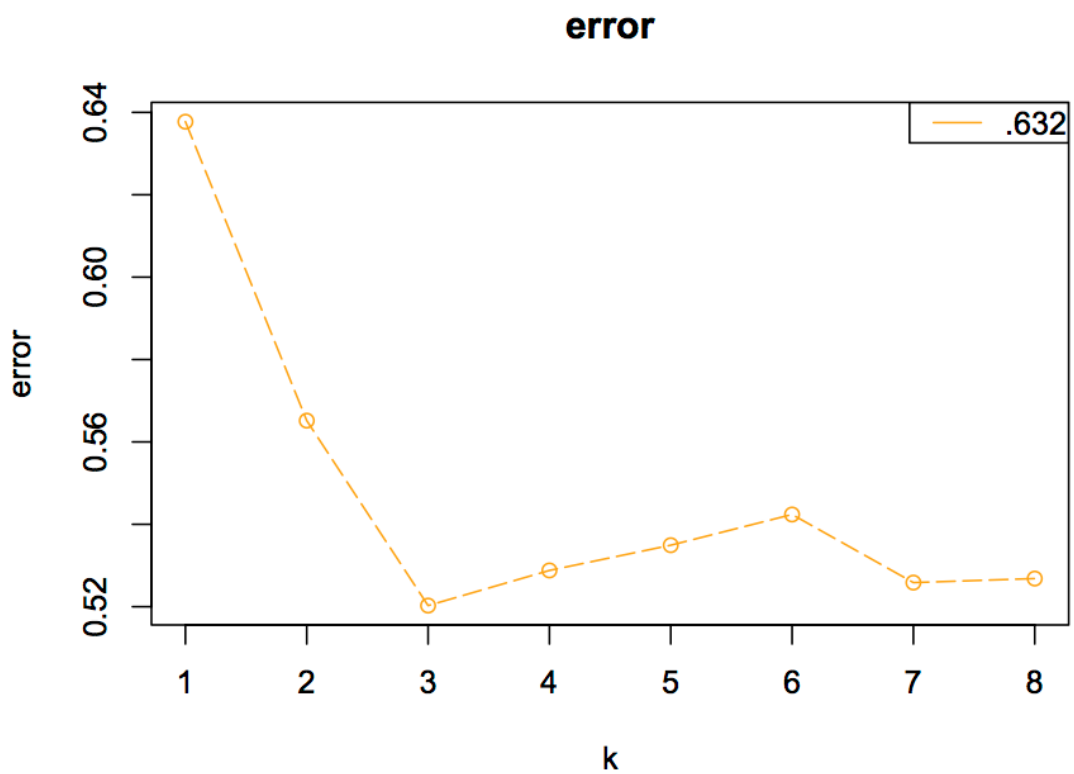
- We can see that error for k<5 is good from the plot above since error from this is less.



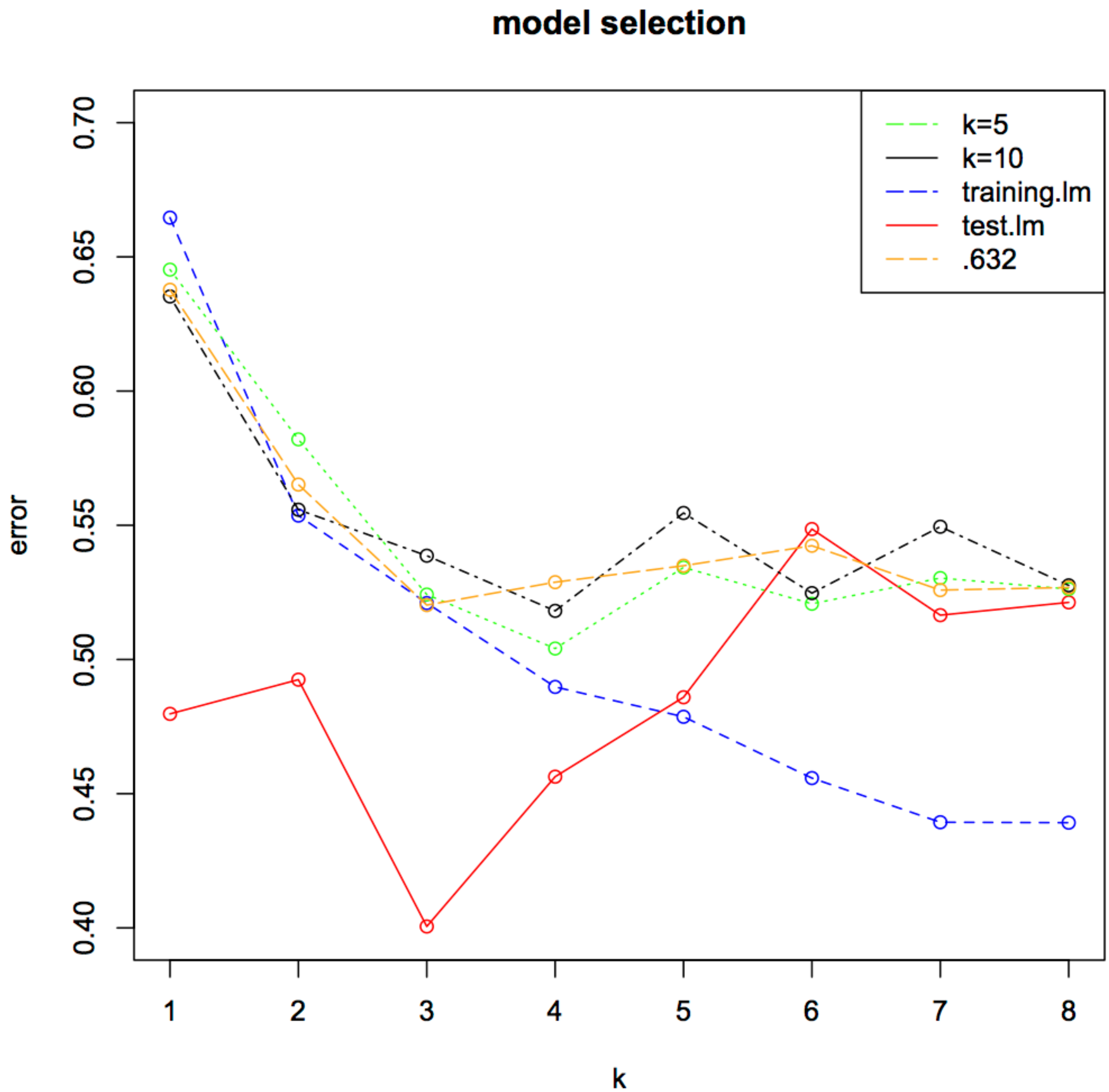
- AIC and BIC say the same thing that k=2 performs the best.



- When five- and tenfold cross-validation was performed I found that both are equally good.



- Bootstrap .632 is best for k=3.



- Linear model for k=3 performs best amonst all the models.

Problem 3:

- We will use the "Weekly" data set from the "ISLR" package to predict the "Direction" variable.
- First conducted logistic regression:

```
logit.pred
      0   1
  0  11 244
  1   8 28
```

- We have a classification error of *r 1 - (11 + 282) / (11 + 244 + 8 + 282)*

- Boosting:

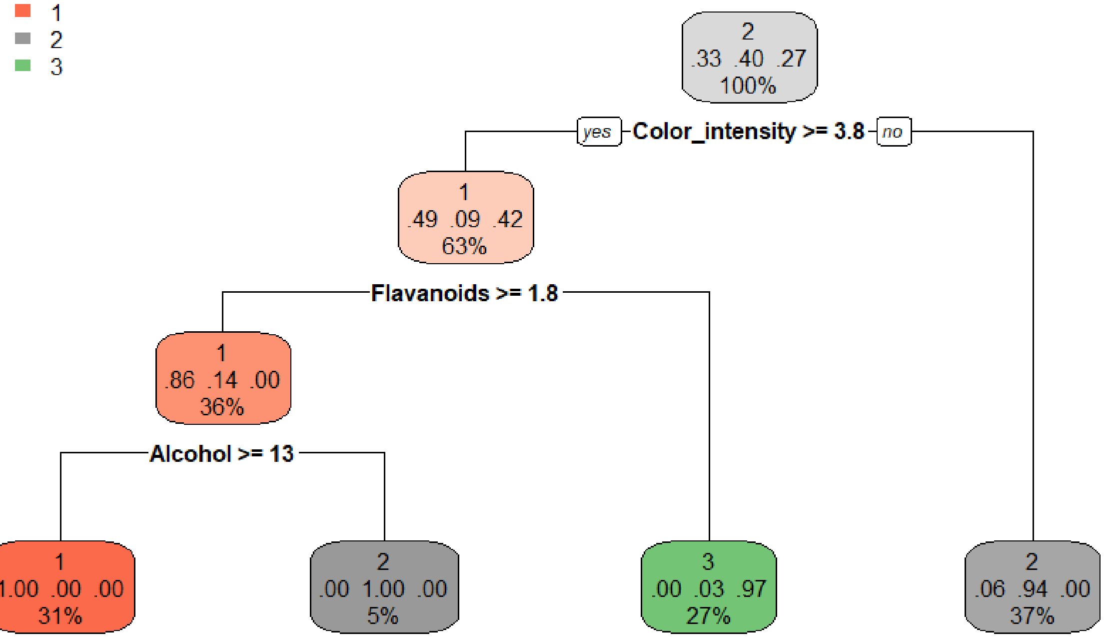
```
boost.pred
      0   1
  0 166  89
  1 181 109
```

- We have a classification error of *r 1 - (166 + 109) / (166 + 89 + 181 + 109)*.

- Bagging:

```
bag.pred
      0   1
  0  85 170
  1  71 219
```

Problem 2:



train accuracy is

97.1831 test accuracy

is 88.88889

In the training set

node4 = 47, node5 = 7, node6 = 51, node7 =

37 In the testing set

node4 = 12, node5 = 1, node6 = 11, node7 = 12

- We have a classification error of *r 1 - (85 + 71) / (85 + 170 + 71 + 219)*.

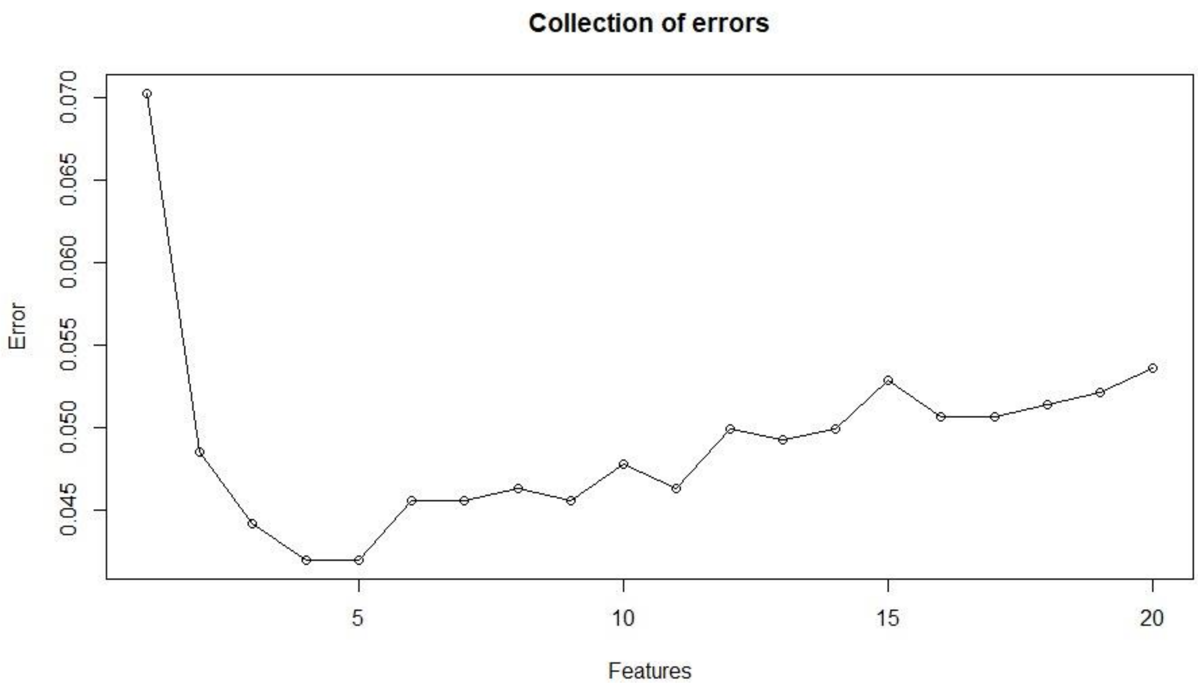
- Random forest:

```
       rf.pred
          0   1
     0   69 186
     1   62 228
```

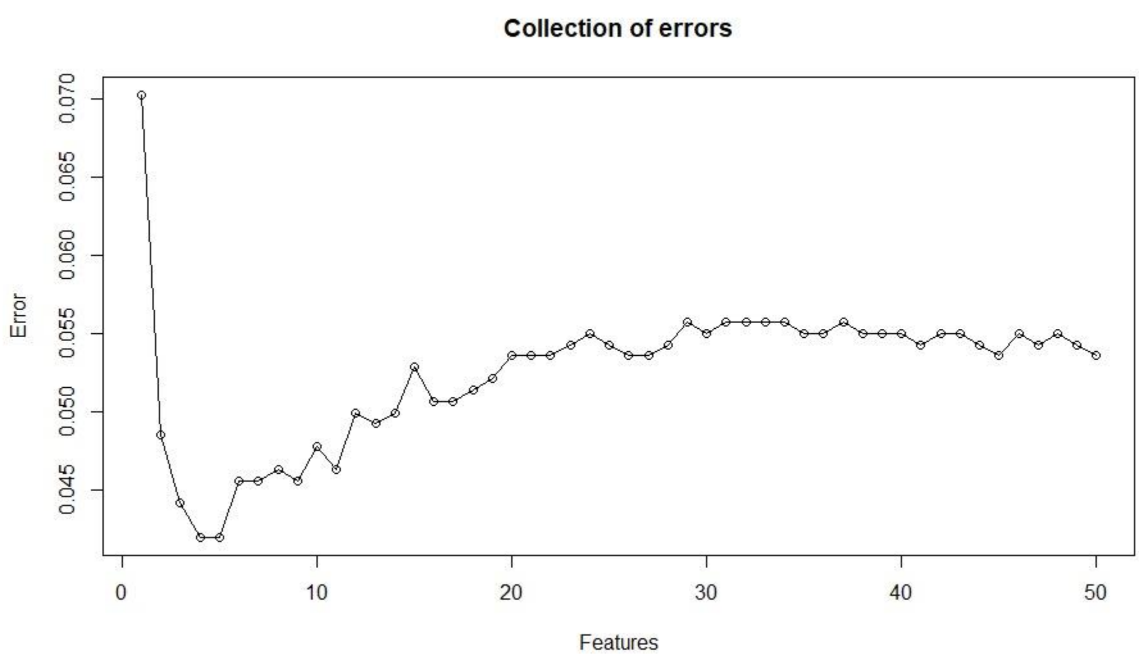
- We have a classification error of *r 1 - (69 + 228) / (69 + 186 + 62 + 228)*.

- We may conclude that random forests gave the lowest classification error.
- Advantages that committee machines have related to the data set that we selected 'Weekly Data' is Decision boundary in the data points can be easily fitted in the decision trees because there is linear separation that can be modeled using decision trees.

Problem 4:



- It is best when you take 5 features



Problem 5:

Here we are using the spam dataset which has a 4600 data in it. For this problem or major attribute will be spam values. So we converted the values in terms of 1's and 0's for or model. We are defining our vectors and the neural network first, then we will training it with the training dataset we have generated from the original data.

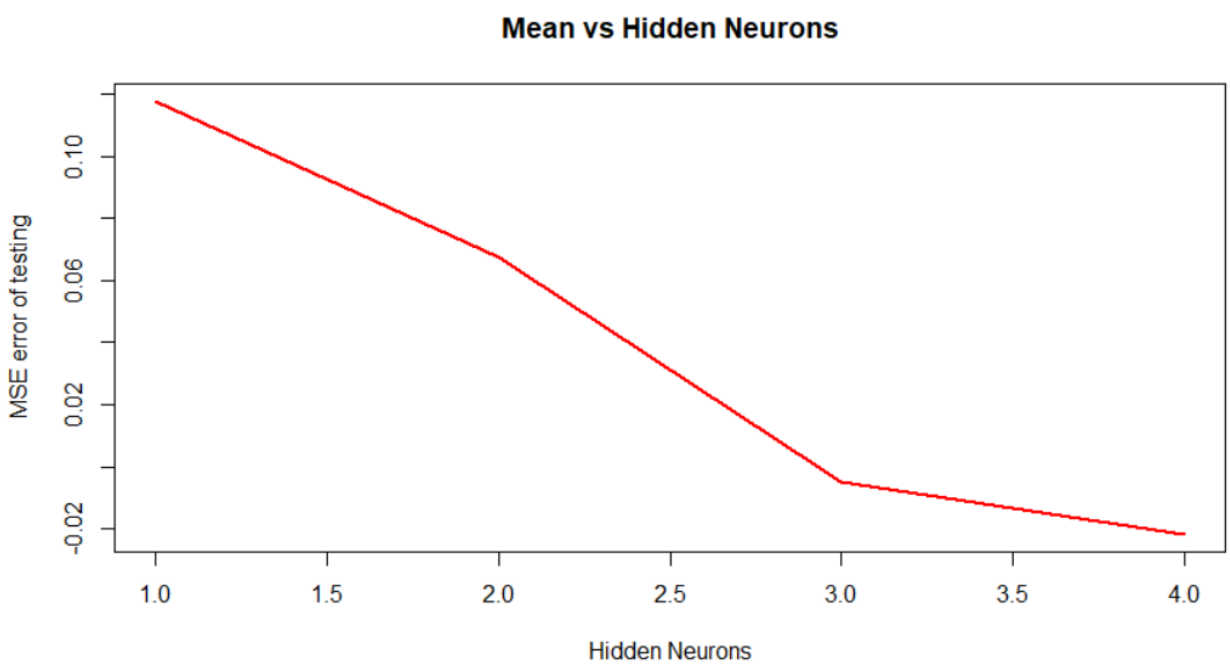
Training data used is 60% of the original data. Other data are being used for testing and optimizing the model.

We are also finding the cross validation error to do the comparison of the model.

After building the network and training and testing it with the dataset we build, below are the error values.

```
> training_error
[1] 0.06695652174 0.05282608696 0.03978260870 0.04195652174
> testing_error
[1]  0.117826086957  0.067608695652 -0.004710144928 -0.021739130435
```

Next we'll see the relation between mean values and the number of hidden neurons in the network.



Below is the error values we got by running the network on out test data set

```
> error
[1] 0.0252173913
```

True vs predicted values count:

```
> table(true,prediction)
    prediction
true    0    1
   0 1303  117
   1   59  821
> 
```

Comparison with additive model:

Below is the summary of the addictive data:

```
> summary(model.additive)

Call: gam(formula = form, family = binomial, data = train.data)
Deviance Residuals:
            Min              1Q          Median              3Q 
-3.858772204460 -0.116152973636 -0.000004197826  0.082366499582 
            Max 
 4.064390696971 

(Dispersion Parameter for binomial family taken to be 1)

    Null Deviance: 3106.090074 on 2299 degrees of freedom
Residual Deviance: 747.3533146 on 2242 degrees of freedom
AIC: 863.3533146 

Number of Local Scoring Iterations: 13 

Anova for Parametric Effects
            Df    Sum Sq   Mean Sq F value    Pr(>F)   
A.1          1    0.0335  0.033518 0.00842 0.9268990   
A.2          1    1.6841  1.684068 0.42302 0.5155017   
A.3          1    3.5848  3.584819 0.90046 0.3427598   
A.4          1    0.0248  0.024805 0.00623 0.9370909   
A.5          1   23.6973 23.697289 5.95247 0.0147737 * 
A.6          1   13.1410 13.140989 3.30086 0.0693771 . 
A.7          1   24.1431 24.143076 6.06445 0.0138677 * 
A.8          1    6.1022  6.102188 1.53280 0.2158229   
A.9          1    6.9300  6.929999 1.74073 0.1871808   
A.10         1    1.6555  1.655528 0.41585 0.5190817   
A.11         1    1.8974  1.897406 0.47661 0.4900353   
A.12         1    4.6236  4.623556 1.16138 0.2812951   
A.13         1    0.0307  0.030732 0.00772 0.9299952   
A.14         1    0.2932  0.293248 0.07366 0.7861049   
A.15         1    2.4893  2.489333 0.62529 0.4291716   
A.16         1   27.0967 27.096731 6.80637 0.0091437 **
A.17         1    4.2461  4.246055 1.06656 0.3018356   
A.18         1    4.1057  4.105690 1.03130 0.3099631   
A.19         1    7.7416  7.741556 1.94459 0.1633096   
A.20         1    0.0262  0.026204 0.00658 0.9353451   
```

```
A.24          1   18.4845 18.484523 4.64309 0.0312850 *
A.25          1   26.4510 26.451016 6.64418 0.0100114 *
A.26          1    1.5573  1.557257 0.39116 0.5317517
A.27          1    4.5231  4.523120 1.13615 0.2865814
A.28          1    1.0114  1.011362 0.25404 0.6142927
A.29          1    2.3876  2.387587 0.59973 0.4387615
A.30          1    0.0020  0.001951 0.00049 0.9823390
A.31          1    0.1386  0.138554 0.03480 0.8520259
A.32          1    0.0473  0.047321 0.01189 0.9131923
A.33          1    0.6889  0.688857 0.17303 0.6774705
A.34          1    0.8250  0.825021 0.20724 0.6489870
A.35          1    0.0154  0.015371 0.00386 0.9504594
A.36          1    5.3889  5.388887 1.35362 0.2447694
A.37          1    0.2922  0.292236 0.07341 0.7864655
A.38          1    3.2134  3.213396 0.80717 0.3690558
A.39          1    2.3029  2.302907 0.57846 0.4469955
A.40          1    0.0001  0.000099 0.00002 0.9960297
A.41          1    0.4472  0.447161 0.11232 0.7375480
A.42          1    4.3726  4.372563 1.09834 0.2947447
A.43          1    0.2847  0.284690 0.07151 0.7891743
A.44          1   12.4167 12.416715 3.11893 0.0775237 .
A.45          1    6.3972  6.397164 1.60689 0.2050611
A.46          1   13.0031 13.003058 3.26621 0.0708547 .
A.47          1    1.5326  1.532570 0.38496 0.5350220
A.48          1    1.0035  1.003509 0.25207 0.6156711
A.49          1    1.5783  1.578312 0.39645 0.5289909
A.50          1    3.7724  3.772403 0.94758 0.3304404
A.51          1    0.0013  0.001286 0.00032 0.9856609
A.52          1    9.1235  9.123495 2.29171 0.1302073
A.53          1   12.5398 12.539759 3.14984 0.0760702 .
A.54          1    1.1899  1.189864 0.29888 0.5846401
A.55          1   32.2054 32.205396 8.08961 0.0044923 **
A.56          1    0.5269  0.526877 0.13235 0.7160471
A.57          1   10.7030 10.702976 2.68846 0.1012178
Residuals 2242 8925.5856  3.981082
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Below is the performance output of the additive model:

```
> error
[1] 0.0252173913
> table(actual,predicted)
      predicted
actual    0    1
     0 1303  117
     1   59  821
```

So here we can conclude that the additive model fared moderately compared to the other model. But the ANN and the CV results of error were better than the additive model.

Problem 7:

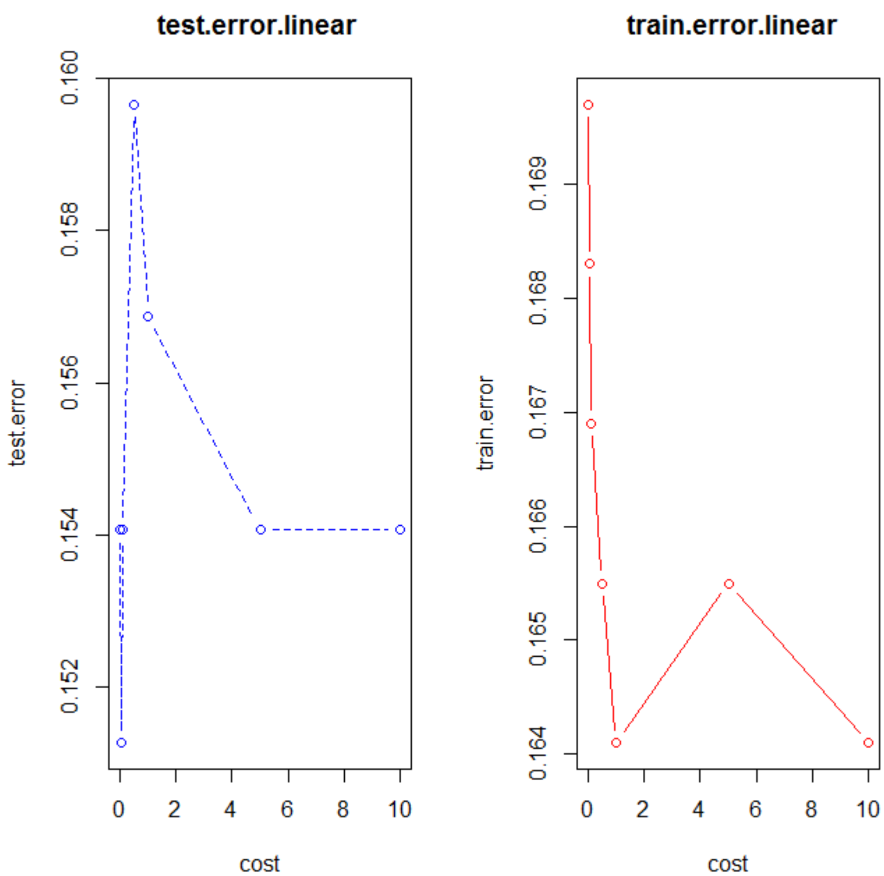
First, I used 1/3rd data as test data and the other data as training data.

A)

I set the cost equals (0.01,0.05,0.1,0.5,1,5,10) and the kernel is the linear. The test error and the train error are on the following table

linear.test.errlinear.train.errcost

[1,]	0.1540616246	0.1697054698	0.01
[2,]	0.1512605042	0.1683029453	0.05
[3,]	0.1540616246	0.1669004208	0.10
[4,]	0.1596638655	0.1654978962	0.50
[5,]	0.1568627451	0.1640953717	1.00
[6,]	0.1540616246	0.1654978962	5.00
[7,]	0.1540616246	0.1640953717	10.00

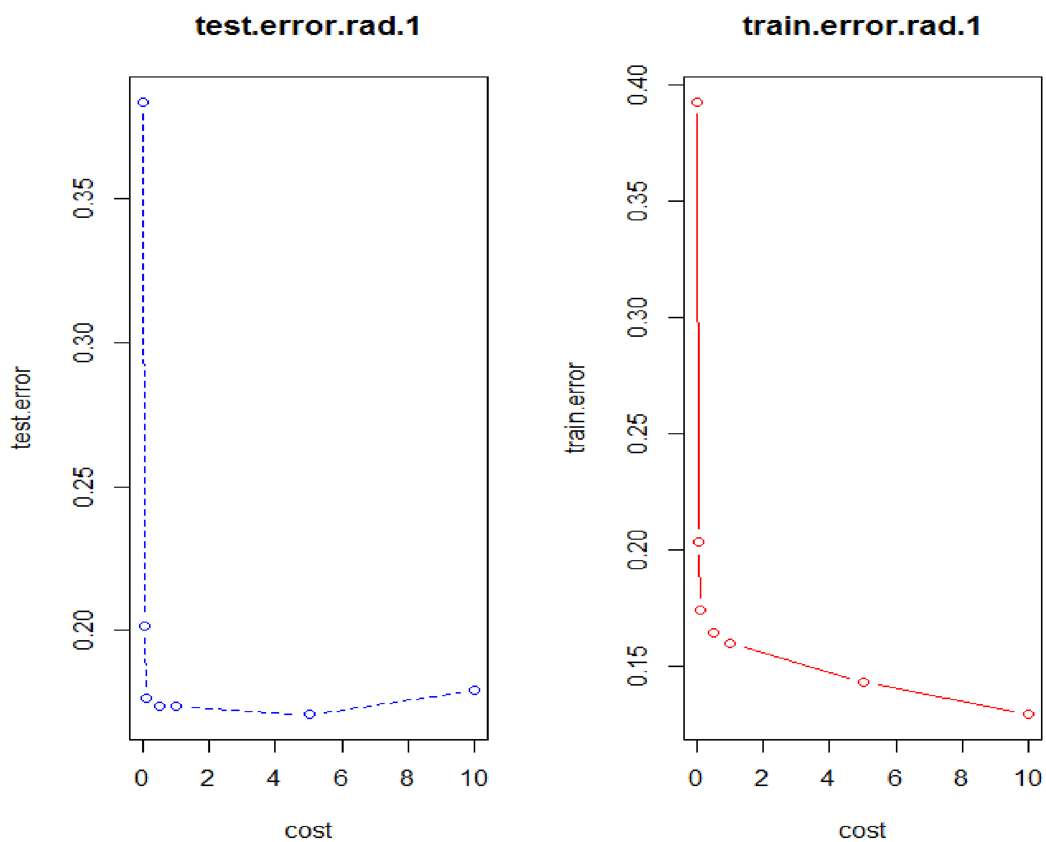


From the figure, we can see that when cost equals to 0.05 we have the minimum test error which is 0.1512605042 so the model whose cost is 0.05 is the best model. But from the table we can see that the test error is smaller than train error. I don't know the reason.

B)

I set the cost equals (0.01,0.05,0.1,0.5,1,5,10) and the kernel is the radial. The test error and the train error are on the following table

	rad.test.err.1	rad.train.err.1	cost
[1,]	0.3837535014	0.3927068724	0.01
[2,]	0.2016806723	0.2033660589	0.05
[3,]	0.1764705882	0.1739130435	0.10
[4,]	0.1736694678	0.1640953717	0.50
[5,]	0.1736694678	0.1598877980	1.00
[6,]	0.1708683473	0.1430575035	5.00
[7,]	0.1792717087	0.1290322581	10.00

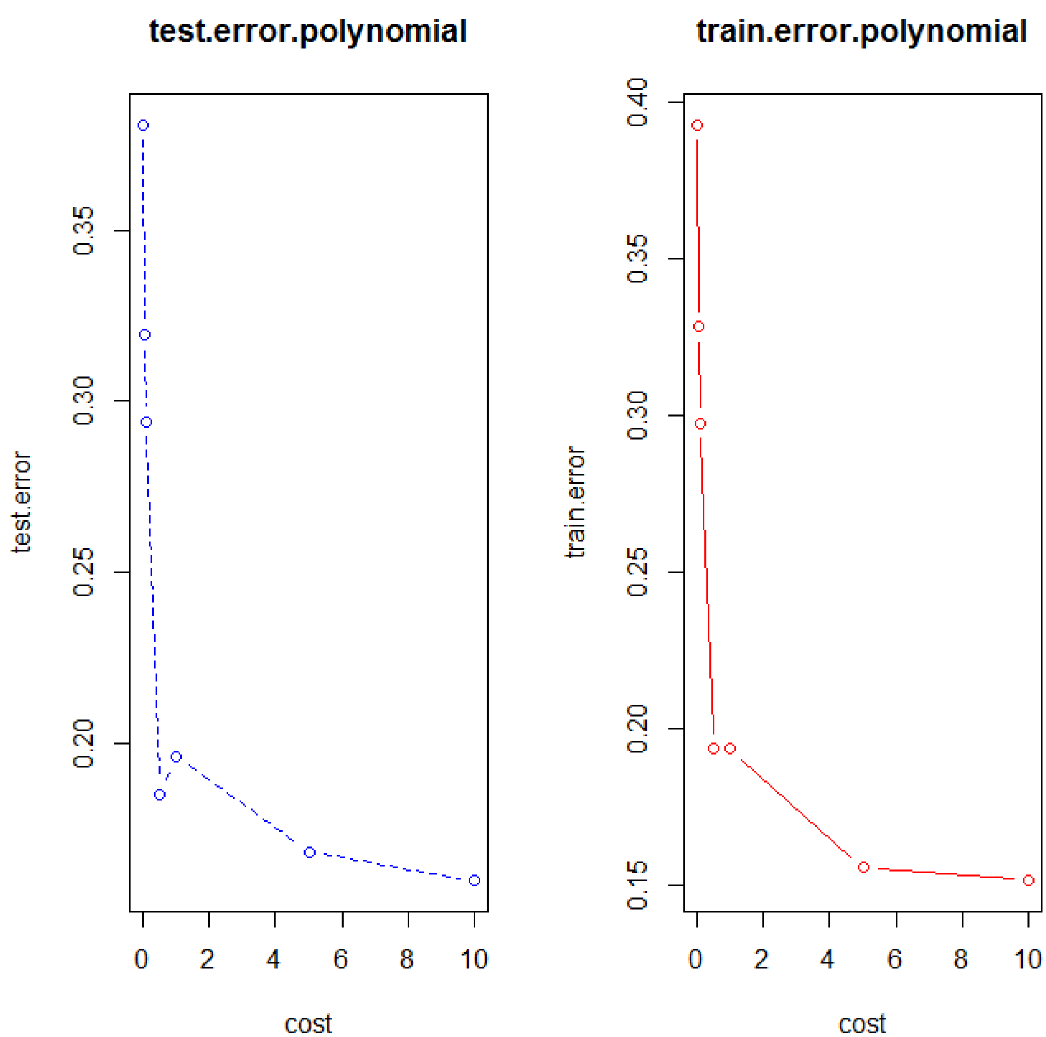


From the figure, we can see that when cost equals to 5 we have the minimum test error which is 0.1708683473 so the model whose cost is 5 is the best model.

I set the cost equals (0.01,0.05,0.1,0.5,1,5,10) and the kernel is the polynomial. The test error and the train error are on the following table

polynomial.test.err polynomial.train.err cost

[1,]	0.3809523810	0.3927068724	0.01
[2,]	0.3193277311	0.3281907433	0.05
[3,]	0.2941176471	0.2973352034	0.10
[4,]	0.1848739496	0.1935483871	0.50
[5,]	0.1960784314	0.1935483871	1.00
[6,]	0.1680672269	0.1556802244	5.00
[7,]	0.1596638655	0.1514726508	10.00



From the figure, we can see that when cost equals to 10 we have the minimum test error which is 0.1596638655 so the model whose cost is 10 is the best model.

Overall, radial basis kernel seems to be producing minimum misclassification error on both train and test data.