Project Two

Jacob Hansén (960512-1533),
Kevin Olsson (960406-0336), and
Liam Persson (970502-0858)

November 2019

Contents

1 Part A

1.1 Method and resources

This part of the project was carried out using the program R, version 3.5.2. The packages used were the following:

Package	Purpose
ISLR	Obtain data.
ggplot2	Plotting relevant plots.
MASS	Data operation functionality.
e1071	Support Vector Machine operations.
ellipse	Data mangement.
caret	Classification training.

In this assignment, the basic R functions were used to fit the models. For example, for the Support Vector Machine (SVM) classifier following was run: "classifier = svm(formula = y ., data = trainingset, type= 'C-classification', kernel = 'linear')". Moreover, no noteworthy functions were used to conduct any special operations to the data, other than using certain commands of used packages to plot the data in a desired way, etc.

1.2 Assignment a

The package *ISLR* was used to access the data set "Auto". As in project 1 each observation was then classified with a new variable mpg_bin with the following specifications:

$$mpg\_bin = 1, if mpg \geq 22.75$$
$$mpg\_bin = 0, if mpg < 22.75$$

Moreover, the variable mpg was removed as instructed in the assignment. If this variable were included in the predictive models trained on this data with mpg_bin as the dependent variable, then the variable mpg would be the most deciding variable since mpg_bin depends entirely on mpg. Furthermore the SVM classifier with corresponding combinations of kernels, costs and parameters were evaluated in order to find which one performs best.

Linear	Cost	Train. err	Test. err
1	0.01	0.073	0.077
2	0.1	0.035	0.077
3	1	0	0.026

Table 1

Polynomial	Cost	Train. err	Test. err
1	0.01	0	0.051
2	0.1	0	0.012
3	1	0	0.026

Table 2

Radial Basis Function	Cost	Train. err	Test. err
1	0.01	0	0.0057
2	0.1	0	0.0057
3	1	0	0.0057

Table 3

Sigmoid	Cost	gamma	Train. err	Test. err
1	0.01	1	0.073	0.10
2	0.01	10	0.070	0.12
3	0.01	100	0.080	0.13
4	0.1	1	0.092	0.13
5	0.1	10	0.15	0.23
6	0.1	100	0.14	0.19
7	1	1	0.14	0.15
8	1	10	0.17	0.29
9	1	100	0.17	0.31

Table 4

Insights gained from the results is that Radial Basis Functions (RBF) yield the lowest test error. It is also made evident that the use of a higher cost in the linear SVM results in a lower training error. Intuitively, this is logical as the margins are strictly fit to the training data and no or less miss-classifications are tolerated. This complexity, wanting to choose a model that both accurately captures the regularities in its training data, but also generalizes well to unseen data plagues all of statistical learning and is generally called bias-variance trade off.

1.3 Assignment b

For all of the different methods and parameters we can see the same pattern for r. If we increase r, the test and training error is decreasing. This is easy to understand intuitively. If we shift the data point a longer distance in relation to the points we haven't moved it is obvious that they should be easier to separate.

Furthermore it is evident that the change of the value k, that decides ratio of training or testing set, from 0.5 to 0.8 yields lower training error and test error

picture.png picture.png

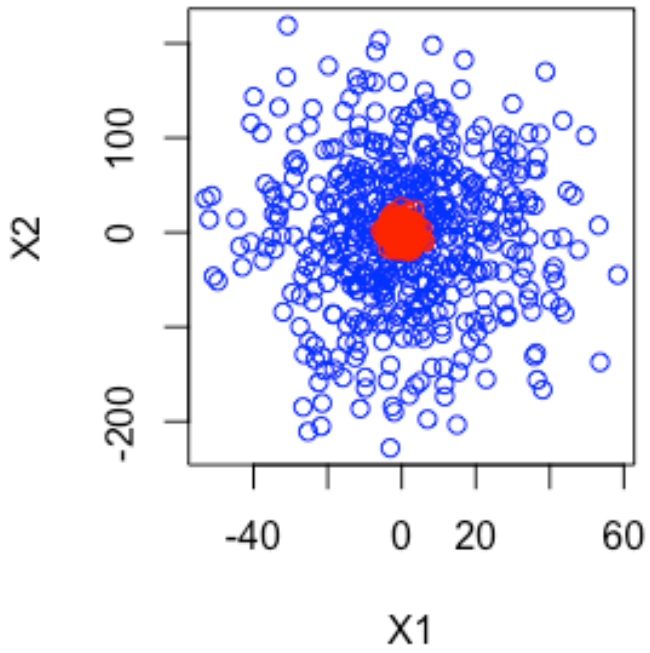

Figure 1: Scatter plot after radial shift of data points

for all the methods and parameters. However, it is important to note that it is due to a large difference between number of data points from the two sets which will create a tendency to classify more points to the larger set. For example if we have 99,9% of the data points from one distribution, even a poor method will classify all the points to the largest data set with low training and test error but the decision boundary we get from this method is far from the Bayes decision boundary.

Linear	Cost	r	k	Train. err	Test. err
1	0.01	2	0.5	0.29	0.30
2	1	2	0.5	0.27	0.3
3	0.01	2	0.8	0.14	0.15
4	1	2	0.8	0.14	0.15
5	0.01	4	0.5	0.22	0.23
6	1	4	0.5	0.12	0.12
7	0.01	4	0.8	0.11	0.12
8	1	4	0.8	0.058	0.065

Table 5

Polynomial	Cost	r	k	Train. err	Test. err
1	0.01	2	0.5	0.33	0.37
2	1	2	0.5	0.32	0.37
3	0.01	2	0.8	0.14	0.12
4	1	2	0.8	0.14	0.12
5	0.01	4	0.5	0.12	0.14
6	1	4	0.5	0.12	0.14
7	0.01	4	0.8	0.068	0.045
8	1	4	0.8	0.058	0.045

Table 6

Sigmoid	Cost	r	k	Train. err	Test. err
1	0.01	2	0.5	0.27	0.31
2	1	2	0.5	0.40	0.41
3	0.01	2	0.8	0.20	0.20
4	1	2	0.8	0.26	0.31
5	0.01	4	0.5	0.20	0.25
6	1	4	0.5	0.17	0.15
7	0.01	4	0.8	0.057	0.054
8	1	4	0.8	0.058	0.079

Table 7

Radial Basis Function	Cost	r	k	Train. err	Test. err
1	0.01	2	0.5	0.37	0.35
2	1	2	0.5	0.37	0.35
3	0.01	2	0.8	0.14	0.15
4	1	2	0.8	0.14	0.15
5	0.01	4	0.5	0.15	0.16
6	1	4	0.5	0.15	0.16
7	0.01	4	0.8	0.058	0.065
8	1	4	0.8	0.058	0.065

Table 8

2 Part B

2.1 Method and resources

This part of the project was carried out using the program R, version 3.5.2. The packages used were the following:

Package	Purpose
ISLR	Obtain data.
ggplot2	Plotting relevant plots.

In this assignment, the package *rpart* was used to grow the classification and regression trees. When doing classification trees, there are two key constraint variables to this method: the minimum split constraint and the complexity parameter. The minimum split is the number of observations that need to exist in a node before a split is allowed. The complexity parameter is the factor by which a split needs to improve the model to be attempted. In regression trees, this means that the split must increase the R-squared value by the assigned complexity parameter for each split. Essentially, the lower the complexity parameter, the more splits will be attempted. Obviously, setting relatively low values for both the minimum split parameter and the complexity parameter will yield a large tree. These two parameters will be the ones varied in order to fit different trees in both assignments and they will be called *minsplit* and *cp* (Complexity Parameter).

2.2 Assignment a

The first part of this assignment was carried out in the same manner as in Part A of Project One, as requested. The variable *mpg_bin* was computed in the same way, to be used as the classification target. However, there was no variable selection conducted in this part of the project. The reason behind this is that classification and regression tree methods (CART methods) only use the most important variables

when creating a split in the modelled tree. Thus, if a variable is not significant to the model developed, it will not be included either way. The model fitted was thus:

$$mpg\_bin \sim cylinders + displacement + horsepower + weight + acceleration + origin + year$$

Since the target variable, *mpg_bin* was a classification variable, the a classification tree was fitted to this model. A training data set and a test data set were sampled from the original data, where the size of the training data was 75% of the original data set and the test data 25% of the original data set. Below, four models were developed: two unpruned trees and two pruned trees which were derived from each of the unpruned trees.

Firstly, *minsplit* was set to 1 and *cp* to a miniscule value of $0.00001 = 10^{-5}$. This configuration was set so that the largest tree possible would be developed. As expected, an overfitted model was obtained, and it was so overfitted that it correctly classified all the training data. This model will be denoted Model 1. This overgrown tree was then pruned with $cp = 0.01$, which yielded Model 2.

Subsequently, an unpruned tree was fitted but this time with $minsplit = 5$ and $cp = 0.0001 = 10^{-4}$ in order to obtain a less overfitted unpruned tree. Let this model be called Model 3. This tree was pruned with $cp = 0.02$ which yielded Model 4.

The models were all evaluated according to their training error and test error and the results are presented below.

Model	Pruned?	Minsplit	cp	# splits	Train. err.	Test err.
1	Unpruned	1	10^-5	26	0%	7.14%
2	Pruned	N/A	0.01	10	3.06%	6.12%
3	Unpruned	5	10^-4	13	2.38%	6.12%
4	Pruned	N/A	0.02	4	6.46%	7.14%

Table 9

As expected, model 1 gave an overfitted model which resulted in a training error of 0%. The other models with fewer splits also expectedly showed a higher training error. However, models 2 and 3 show a lower test error than model 1, probably displaying the negative effect of overfitting on the predictive ability of a model. Model 4 is perhaps pruned to be too small, since this model gives the highest training error and a test error equal to model 1.

For the task of comparing the errors for varying set sizes of the training and test data sets, the procedure employed in Project 1 was used once again. Four

different splits between the data sets were used and this was done for two different seeds. The numbers shown below are averages of the numbers obtained from each seed.

Type of error	**Prop of data**	**Model 1**	**Model 2**	**Model 3**	**Model 4**
Test	0.8	10.03%	10.03	12.42%	12.42%
Test	0.6	9.11%	8.05%	9.32%	9.75%
Test	0.4	9.24%	8.92%	6.37%	8.60%
Test	0.2	11.39%	12.03%	10.76%	9.49%
Test, average	N/A	**9.94%**	**9.76%**	**9.72%**	**10.06%**
Training	0.8	0.00%	4.31%	2.56%	7.19%
Training	0.6	0.00%	3.83%	2.77%	4.89
Training	0.4	0.00%	2.24%	3.21%	5.45%
Training	0.2	0.00%	0.00%	1.92%	1.92%
Training, average	N/A	**0.00%**	**2.60%**	**2.61%**	**4.86%**

Table 10

This shows no apparent dependence of the test size set and the error rate for either the training error or the test error for any of the models. However, it does give some more reliable error figures for the different models. One can see that model 2 and 3 perform slightly better than model 1 and 4, but the difference is perhaps not as great as the figures in the first table suggest.

2.3 Assignment b

This assignment of Part A was carried out in similar fashion to assignment a, but instead of using a classification tree, a regression tree was fit using *mpg* as the target variable. When predicting values from observations, the output is thus a numerical value; more specifically it is the empirical average of the training data observations of the observations within the region the observation is put in.

Of course, it is easier to answer the question "Does a car have a high mpg or not?" for the classification tree, since the classification tree specifically indicated with a 0 or a 1 if the car did in fact have a high mpg or not respectively. At the expense of its intepretability and simplicity, the classification tree does however give up information, as the regression tree would be far if one were interested in predicting an actual value for mpg for a car.

The regression tree is not incapable of making predictions in the same way was the classification trees, however. For each predicted *mpg* value, one would simply check if that value is lower or higher than the median of the *mpg* in order to obtain a classification from the regression tree. This will be done in order to obtain training and test error rates and moreover compare the classification trees to the regression trees.

Four models are summarized below. Note that for model 1, *minsplit* was set to 2 since setting it to 1 resulted in only a root node.

Model	minsplit	cp	# splits	Train. err.	Test err.
1	2	0.00001	210	0.00%	8.16%
2	N/A	0.001	38	4.08%	8.16%
3	5	0.0001	88	3.06%	9.18%
4	N/A	0.005	11	6.80%	10.20%

Table 11

The results are similar to those obtained in a). Worth noting is that the test error is lowest for model 1 and 2. Perhaps varying the test sizes and the seeds will prove this an anomaly.

Type of error	Prop. of data	Model 1	Model 2	Model 3	Model 4
Test	0.8	9.55%	9.55%	11.78%	10.67%
Test	0.6	12.29%	11.02%	14.19%	12.92%
Test	0.4	15.61%	12.42%	13.06%	14.33%
Test	0.2	15.19%	12.03%	17.09%	15.19%
Test, average	N/A	13.16%	11.25%	14.03%	13.28%
Training	0.8	0.00%	3.35%	1.92%	6.23%
Training	0.6	0.00%	2.77%	2.77%	5.74%
Training	0.4	0.00%	2.24%	2.88%	5.77%
Training	0.2	0.00%	0.64%	3.21%	4.49%
Training, average	N/A	0.00%	2.25%	2.69%	5.56%

Table 12

Model 2 obtains the lowest test error, which would be expected since it is less overfitted than model 1. Comparing these values in *Table 12* to the ones in *Table 10*, one can see that the error rates are generally higher for the regression tree. This is not unexpected since the classification tree was specifically aimed to classify the observations according to *mpg_bin*. While the regression tree offers more information in its predictions, they are thus not better when the task is actually a classification task.

References

[1] Gareth James, Daniela Witten, Trevor Hastie Robert Tibshirani. *An Introduction to Statistical Learning*. Springer Science, New York 2013