**Analytics for Business Intelligence**

Last name **Subraveti Ranganatha** First name **Ajay Simha** Section number (31, 40 or 60) **31**

**Final project**  Total marks are 25 which provide 25% to the total assessment. Students must implement the homework using R language, cut-and-paste of outputs from text book are not permitted (and easily detectable).

The Steps of this homework should be implemented using RStudio and printouts inserted in this document after each step.

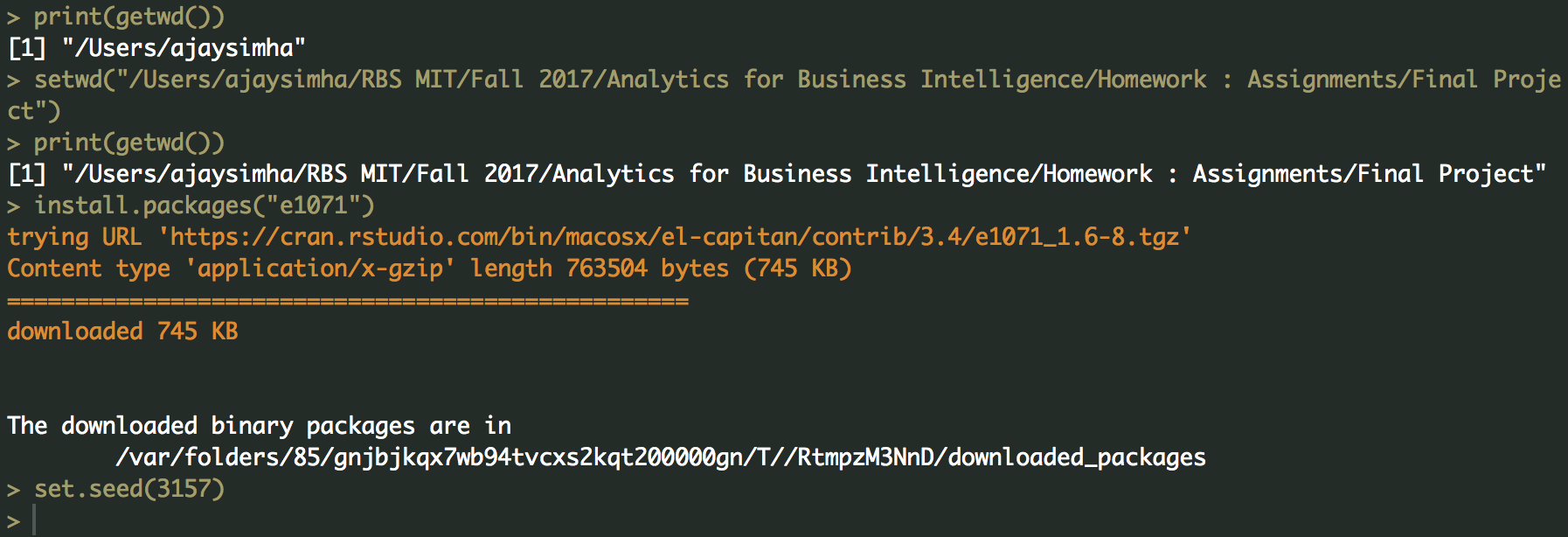
Advanced Classification: Classifiers and Support Vector Machine

Support vector classifier (Refer Section 9.6 from the text book) 13 marks

The e1071 library contains implementations for a number of statistical learning methods. In particular, the svm() function can be used to fit a support vector classifier when the argument kernel="linear" is used. A cost argument allows us to specify the cost of a violation to the margin. When the cost argument is small, then the margins will be wide and many support vectors will be on the margin or will violate the margin. When the cost argument is large, then the margins will be narrow and there will be few support vectors on the margin or violating the margin.

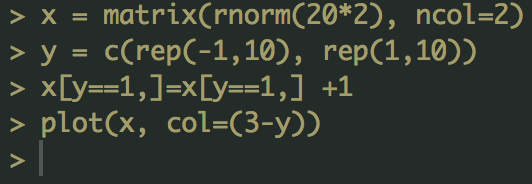
In this project we assume that students will implement independently all necessary steps like setting the working directory and connecting the library, which were explained before in HW1 and HW2.

**Step 1** Set variable k equal to the last 4 digits of your student number. Then initialize the random number generator as set.seed(k). This is an important requirement which makes all project results different for all students with very high level of probability. Do not re-set this value for other steps of this work.

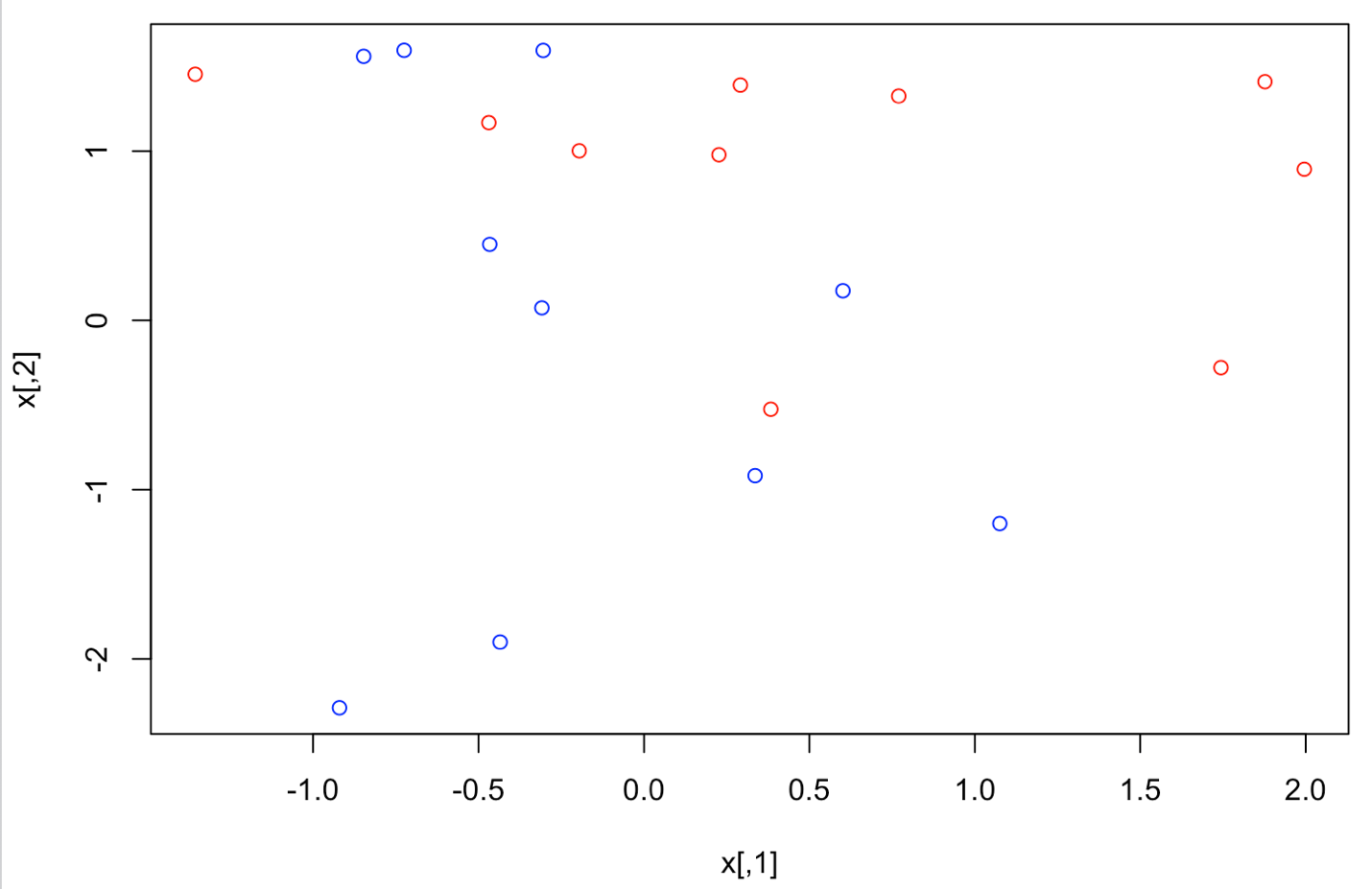


**Step 2** (1 mark) We begin by generating the observations, which belong to two classes, and checking whether the classes are linearly separable. Use commands matrix to generate two sets of data.

Plot these data using command plot. Demonstrate this plot and answer to the questions if these two sets are separable.



|  |
| --- |
|  |



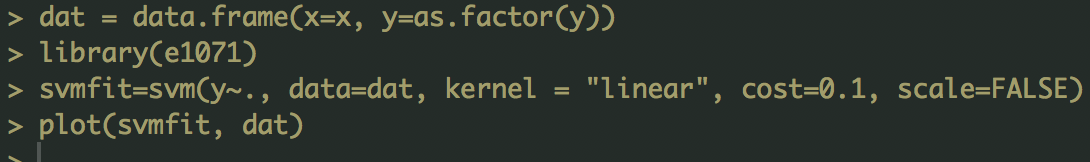
The above two sets are not separable as we cannot justify them individually in a specific area.

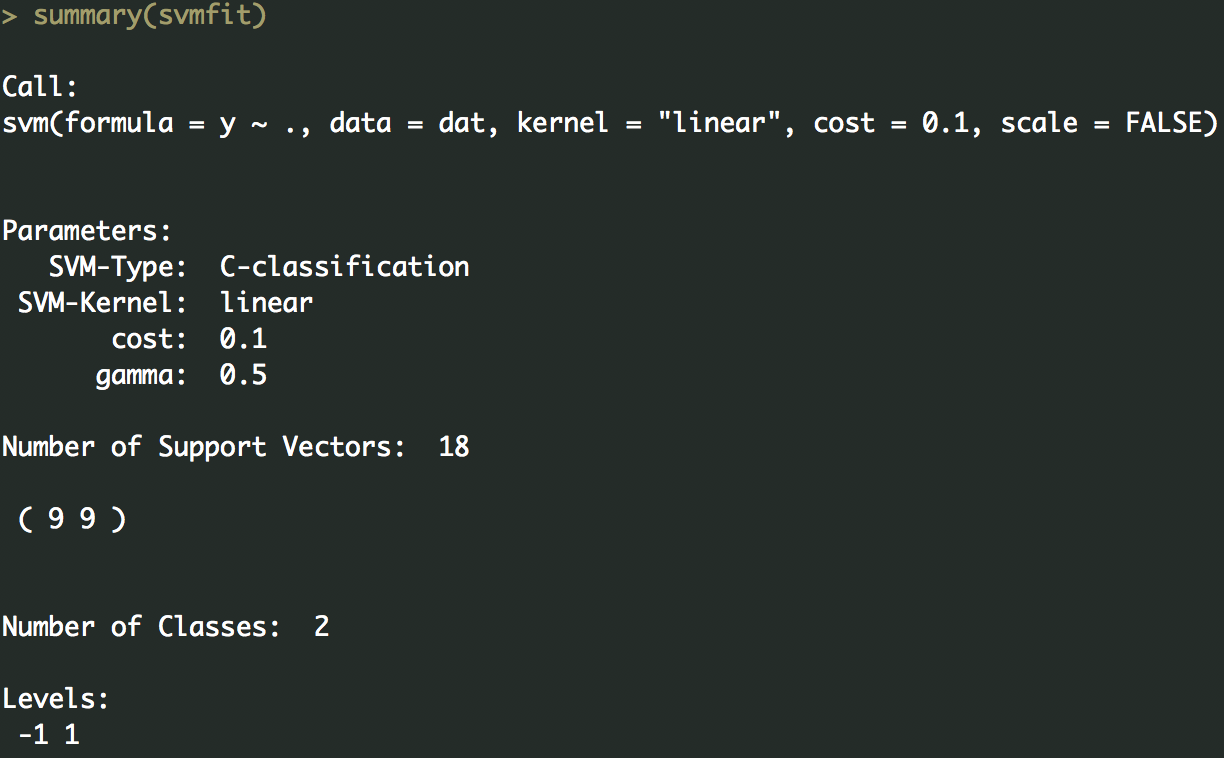
**Step 3** (1 mark) Fit the support vector classifier for cost function value 0.1. Note that

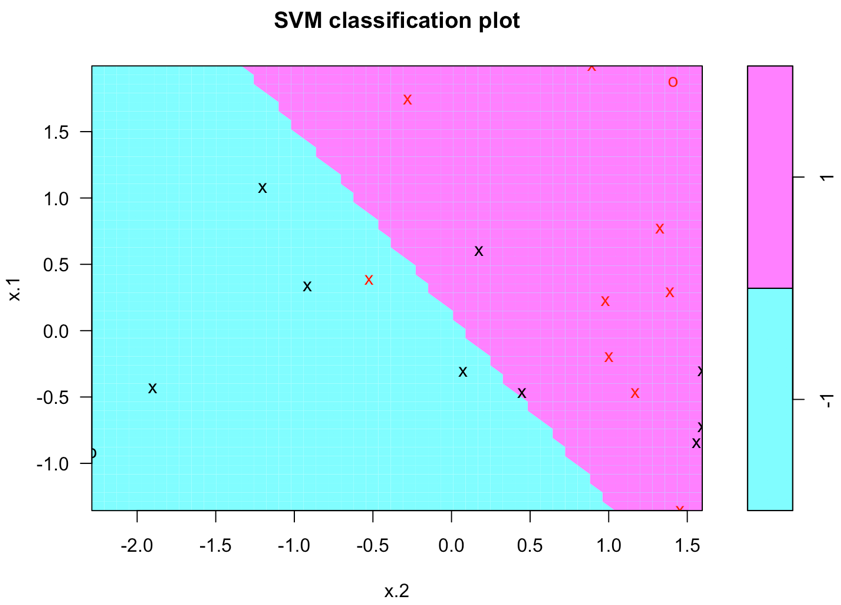
in order for the svm() function to perform classification (as opposed to SVM-based regression), we must encode the response as a factor variable. Provide summary of the svmfit. Plot the support vector classifier obtained.

The important point is that before following the instructions from the text book, or use the R

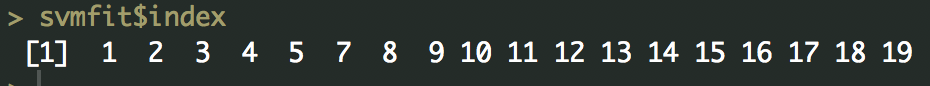
commands from the website, you have to install package e1071.







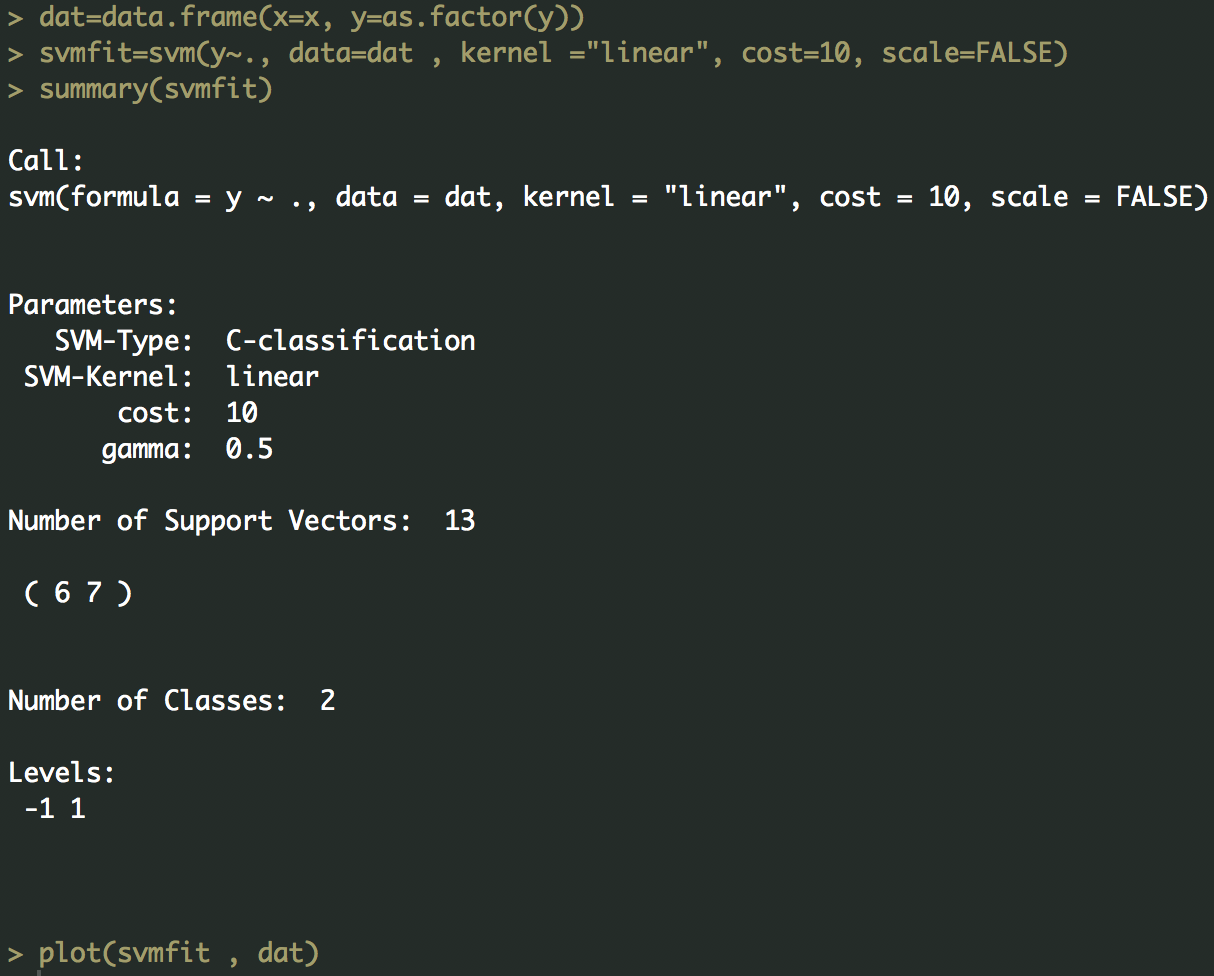
**Step 4** (1 mark) Determine their identities of the support vectors.



No. of support of vectors: 18

**Step 5** (1 mark) Increase number of cost parameter to 10. Check and identify the support

vectors, wrote how they number changed.





The number of support vectors changed when a different cost value was used. In this case, a cost value of 0.1 yielded 18 (9 9) support vectors, while a cost value of 10 yielded 13

(6 7) support vectors.

**Step 6** (1 mark) Compare SVMs with a linear kernel, using a range of values of the cost parameter. Print and interpret summary.

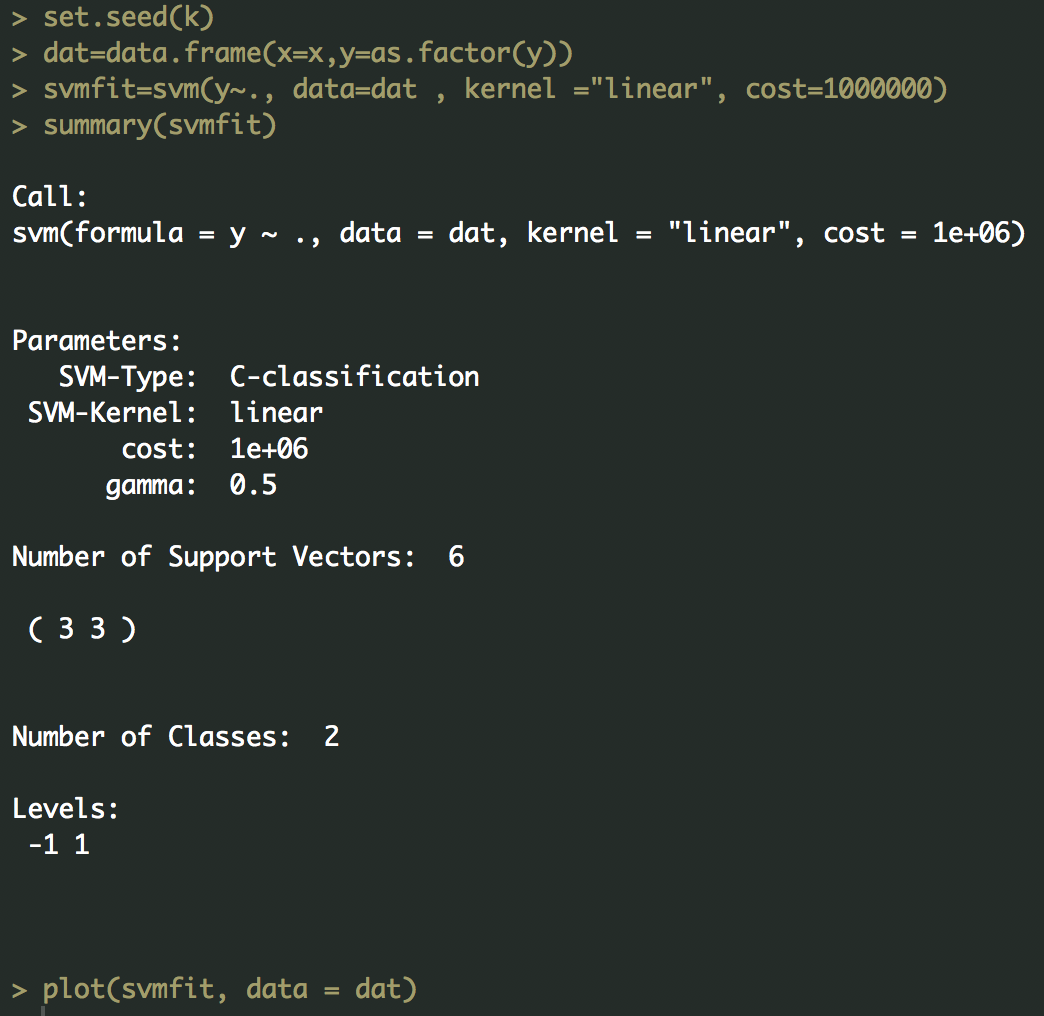
|  |  |
| --- | --- |
| |  | | --- | | Number of support vectors varies with cost. Using the tune() function we can optimize the model’s parameters.  **Step 7** (1 mark) The tune() function stores the best model obtained; accessed it using the command. Print summary.      **Step 8** (2 marks) Generate the test data set and predict the class labels of these test observations. | |

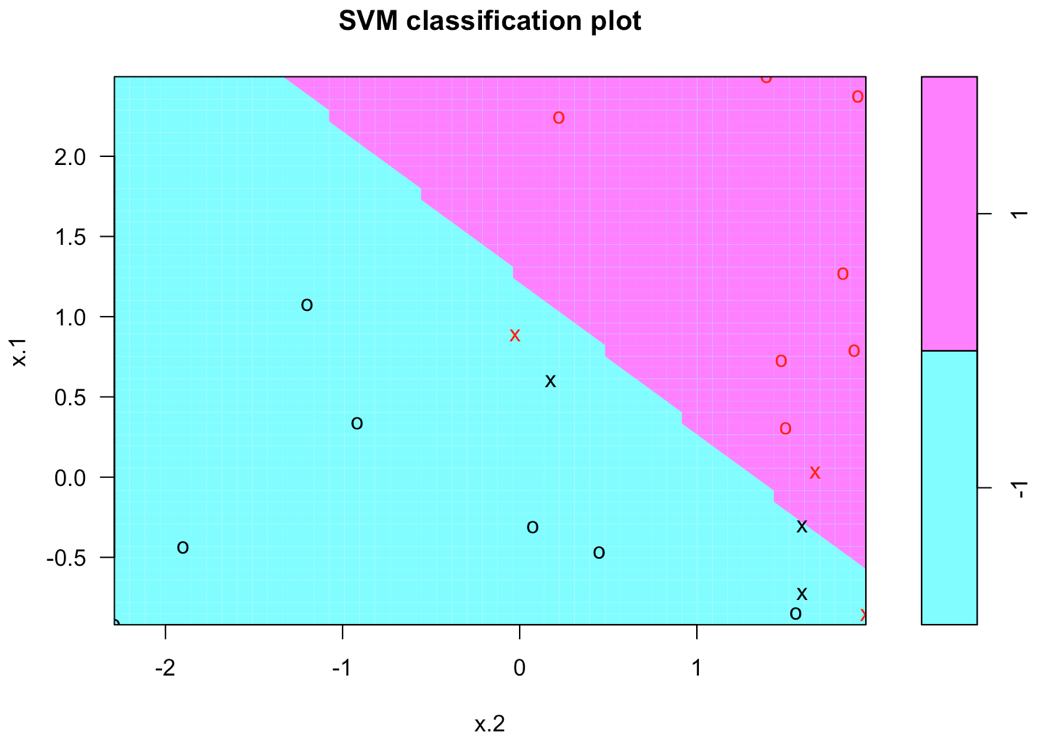
12 of the values are correctly classified

**Step 9** (2 marks) Now consider a situation in which the two classes are linearly separable. Then find a separating hyperplane using the svm() function. Separate the two classes in our simulated data so that they are linearly separable.

|  |
| --- |
|  |
|  |
|  |

**Step 10** (2 marks) Fit the support vector classifier and plot the resulting hyperplane, using a very large value of cost so that no observations are misclassified.





**Step 11** (1 marks) Answer the multiple choice question:

1. Are the support vectors outside of the margin? - **NO**
2. Are the support vectors on the boarder of the margin? - **YES**
3. Are the support vectors within the margin? - **NO**

Support vector machine (Refer Section 9.6 from the text book) 5 marks

In order to fit an SVM using a non-linear kernel, use the svm() function. Use a different value of the parameter kernel. To fit an SVM with a polynomial kernel use kernel="polynomial", and to fit an SVM with a radial kernel use kernel="radial". In the former case we also use the degree argument to specify a degree for the polynomial kernel (this is *d* in (9.22)), and in the latter case we use gamma to specify a value of *γ* for the radial basis kernel (9.24).

**Step 1** (1 marks) Generate some data with a non-linear class boundary and plot them.

|  |
| --- |
| **Step 2** (1 marks) Fit the training data using the svm() function with a radial kernel and γ = 1.      **Step 3** (1 marks) Print summary. What can you tell about of the error? Re-fit the SVM  classification with higher cost. Print summary and plot results. What are your major concern about  these results?    There is a fair amount of training error with this svm fit.      The major concern about these results is the risk of overfitting the data.  **Step 4** (1 marks) Perform cross-validation using tune() to select the best choice of *γ* and cost for an SVM with a radial kernel.        **Step 5** (1 marks) Interpret results: what si the optimal values of cost and *γ* and what is the lowastt percent of misclassified objects?  Optimal value of cost = 1, optimal value of *γ* is 0.5. Lowest percent of misclassified objects is 10%. |
|  |
|  |

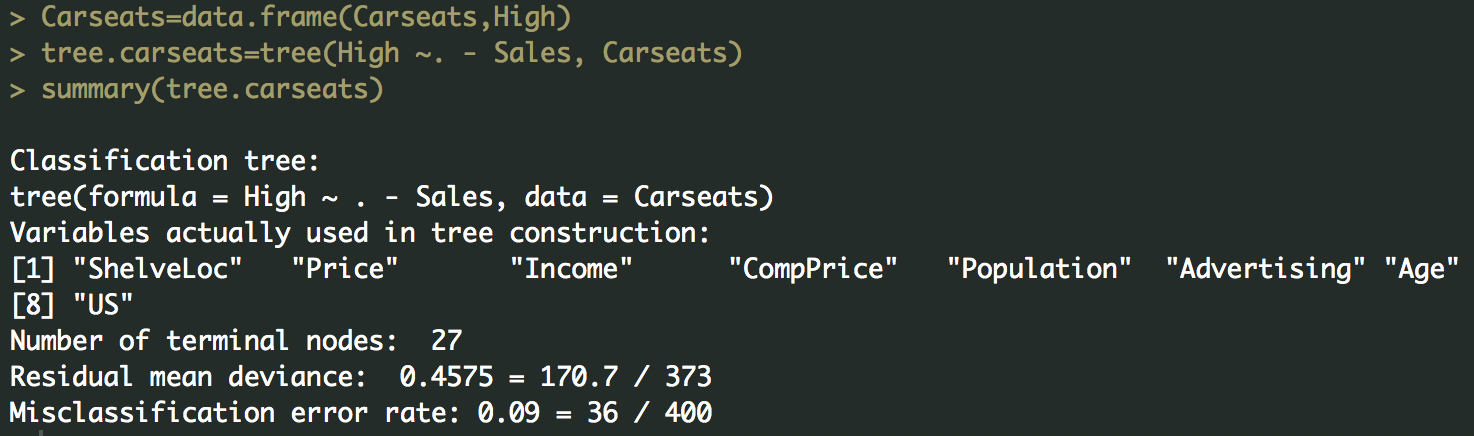
Decision trees for classification (Refer Section 8.3 from the text book)

7 marks

**Step 1** The ISLR and tree libraries are used to construct classification and regression trees. First use classification trees to analyze the Carseats data set. In these data, Sales is a continuous variable, and so we begin by recoding it as a binary variable. Use the ifelse() function to create a variable, called High, which takes on a value of Yes if the Sales variable exceeds 8, and takes on a value of No otherwise. Do not forget to install relevant packages. The description of ISLR package including Carseats (which contains Sales) data set is available on the course website (R language page).

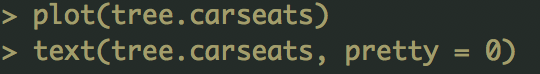


**Step 2** (1 marks) Use the data.frame() function to merge High with the rest of the Carseats data. Use the tree() function to fit a classification tree in order to predict High using all variables but Sales. The syntax of the tree() function is quite similar to that of the lm() function. Use summary() function lists the variables that are used as internal nodes in the tree, the number of terminal nodes, and the (training) error rate. What is the training error rate?



The training error rate is 9%

**Step 3** (1 marks) Plot and text the car seat tree. Provide in your answer the tree **without** texts.

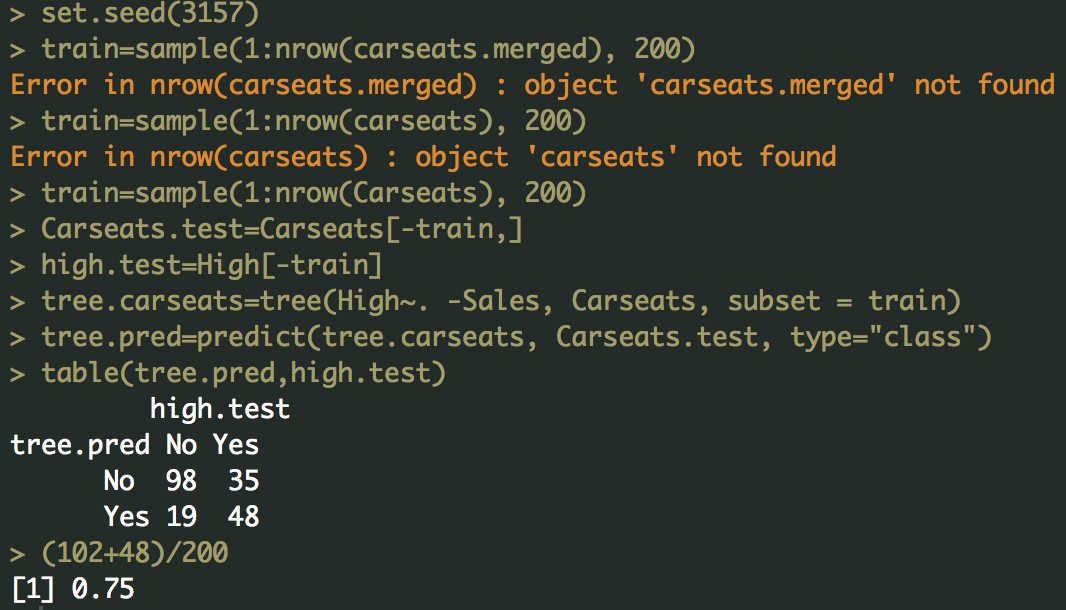




**Step 4** (1 marks) Type the name of the tree object, and analyze the R prints output corresponding to each branch of the tree. R displays the split criterion (e.g. Price<92.5), the number of observations in that branch, the deviance, the overall prediction for the branch (Yes or No), and the fraction of observations in that branch that take on values of Yes and No. Branches that lead to terminal nodes are indicated using asterisks.

|  |
| --- |
|  |
|  |

**Step 5** (1 marks) Evaluate the performance of a classification tree on these data and the training error. Split the observations into a training set (200 records) and a test set, build the tree using the training set, and evaluate its performance on the test data. The predict() function can be used for this purpose. In the case of a classification tree, the argument type="class" instructs R to return the actual class prediction.



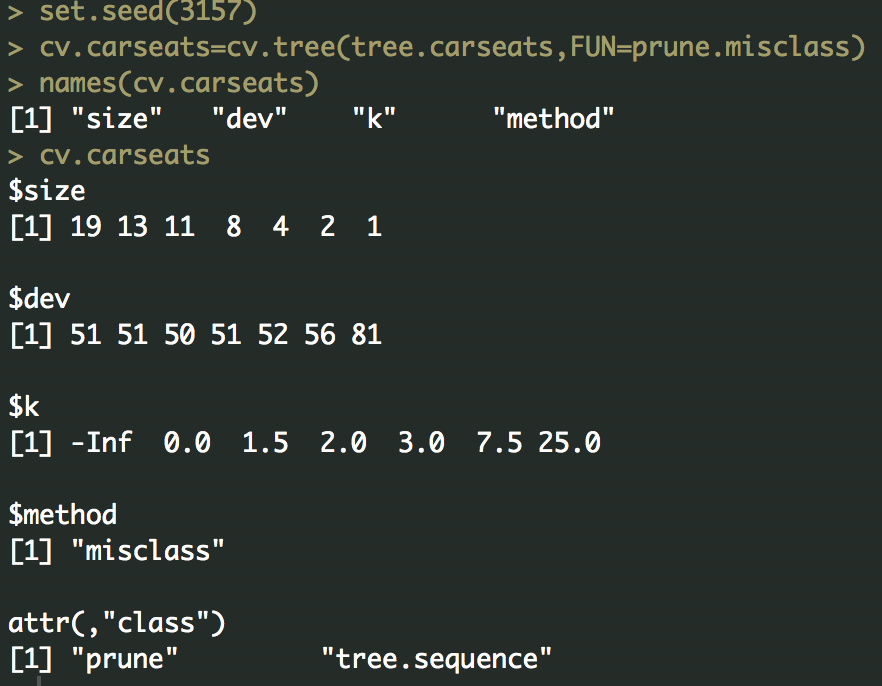
With a random number generator of 3157, the performance of the test data is

75% accurate.

**Step 6** (1 marks) Consider whether pruning the tree might lead to improved results. The function cv.tree() performs cross-validation in order to cv.tree() determine the optimal level of tree complexity; cost complexity pruning is used in order to select a sequence of trees for consideration. Use the argument FUN=prune.misclass in order to indicate that we want the

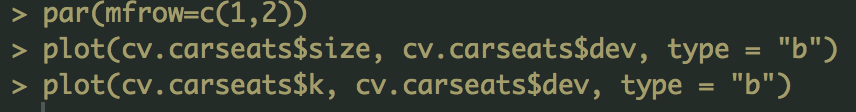
classification error rate to guide the cross-validation and pruning process, rather than the default for the cv.tree() function, which is deviance. The cv.tree() function reports the number of terminal nodes of each tree considered (size) as well as the corresponding error rate and the value of the cost-complexity parameter used (k, which corresponds to *α* in (8.4)).

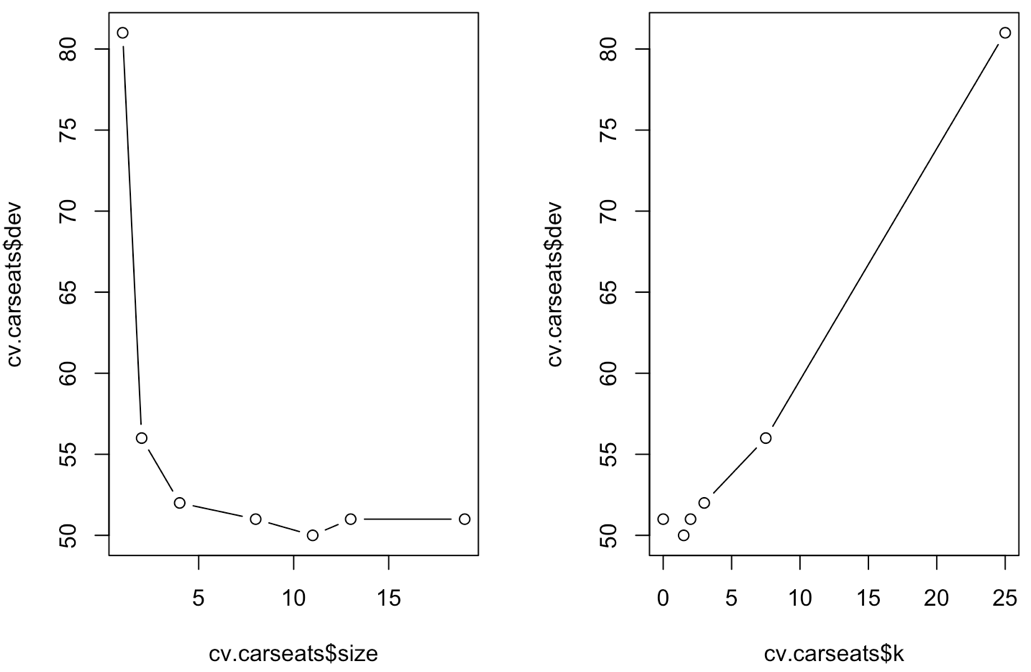
What is the optimal pruning (optimal number of leaves)?



In this case, it appears that pruning is optimized with 25 nodes because this corresponds to the minimum deviation (51).

**Step 7** (1 marks) Plot the error rate as a function of both size and k.





**Step 8** (1 marks) Apply the prune.misclass() function in order to prune the tree to prune.

Obtain the nine-node tree. Plot it **with** text (do not care about overlapping!).

