Assignment 1: Support Vector Machines and Splines

BUAD 5082 Section 1 – Spring 2018

1. Objectives

The purpose of this assignment is to exercise recently developed skills in Support Vector Machines.

2. What You Will Need

- Access to a Windows computer with a recent version of R installed.

3. What You Will Hand In

Submit your **single** script as Assignment1.R via Blackboard – Assignment1.

4. Due Date

Monday February 12th, 2018 just before midnight.

5. Note on Collaboration

This is a Category C individual assignment.

6. Preliminaries:

To get set up for the lab, follow these steps:

1. As the first statement in your script file, enter rm(list=ls())
2. Your response to each lettered question in the assignment should begin with the following three comment lines, where *n* is the Section number and m is the Part number:

   ```
   ######################
   #### Part m
   ######################
   ```

3. I should be able to run your script on my computer without errors or interruptions. For this to happen, you must:
   a. Avoid entering file path information…my files will be located in a different location that yours, and so your code will fail on my machine. Instead, always refer only to files in your working directory.
   b. Do not use functions like file.choose(), fix(),edit(), or q()
   c. Do not include install.packages() functions (or comment them out)
4. Do not create console output other than what is asked of you explicitly. For example, in your final script, remove any statements that you used to verify the contents or structure of data that were not requested.

7. Tasks:

<u>Section 1: Support Vector Machines (34%)</u>

This Section involves the OJ data set which is part of the ISLR package.

a) Use the following code to create a set of indices containing a random sample of 800 integers representing the training subset of set OJ, and a set of test indices representing the remaining observations:

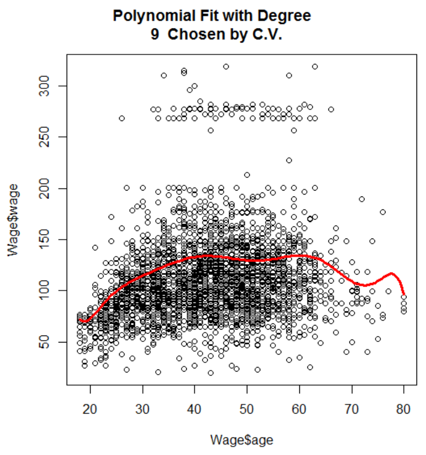
```
set.seed(5082)

n = dim(OJ)[1]

train_inds = sample(1:n,800)

test_inds = (1:n)[-train_inds]
```

b) Fit a support vector classifier to the training data using cost=0.01, with Purchase as the response and the other variables as predictors. Use the summary() function to produce summary statistics, and describe the results obtained.
c) Compute and display the training and test error rates?
d) Use the tune() function to select an optimal cost. Consider values in the range 0.01 to 10.

e) Compute and display the training and test error rates using this new value for cost.
f) Repeat parts (b) through (e) using a support vector machine with a radial kernel. Use the default value for gamma.
g) Repeat parts (b) through (e) using a support vector machine with a polynomial kernel. Set degree=2.
h) Overall, which approach seems to give the best results on this data?

<u>Section 2: Polynomial and Step Function Regression (33%</u>

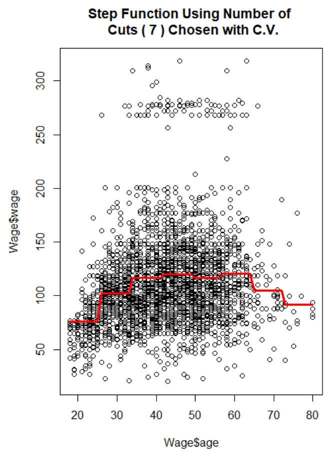
a) Work through the lab in ISLR Section 7.8.1 (beginning on pp. 288) – **<u>do not include this code in your submission</u>**.
b) Set the seed to 5082. Using the Wage dataset from the ISLR package, perform polynomial regression to predict wage using age with polynomial degrees from 1 to 10, and use 10-fold cross-validation to select the optimal degree *d* from these 10 choices (recall cv.glm() function (bottom of page 192).
c) Plot the errors from the cross validation. What degree was chosen?
d) Plot the original data and the polynomial fit using the optimal value of *d*.
    i. To plot the fit, break the range of age into 100 partitions, use the model to predict these points and plot the points against the predictions. My plot looks like this:



e) Fit a step function to predict wage using age (recall the cut() function used in the last step of the lab on Splines). To do this, set the seed to 5082, then investigate step functions using steps from 1 to 12. Use 10-fold cross validation to choose the optimal number of steps.

    Note that the cut() function wants the number of intervals to cut the data into, not the number of steps inside the range. So cut(age,1) produces an error and cut(age,2) has one step…we want from 1 to 12 steps, not intervals

f) Plot the C.V. errors as a function of the number of cuts. What was the optimum number of cuts?
g) Create a model using this optimal number of cuts and plot this model's fitted values as a function of the Boston$age data. I get 8 cuts (7 intervals) as the optimal number and my plot looks like this (so should yours):



Section 3: Regression Splines and Smoothing Splines (33%)

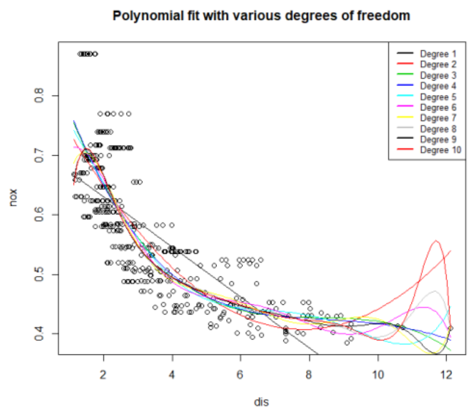
a) Work through the lab in ISLR Section 7.8.2 (beginning on pp. 293) – **do not include this code in your submission.**.

The questions in this section use the Boston data. Specifically, we are interested in Boston$nox ~ Boston$dis. Before starting, create a sequence of 'test' x values composed of 100 equally-spaced points between the min and the max of Boston$dis. One way to do this is the following:

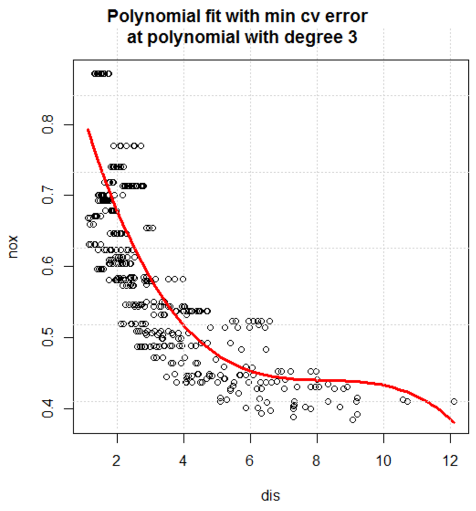
```
> disrange <- range( Boston$dis )
> dissamples <- seq(from=disrange[1], to=disrange[2], length.out=100)
```

All plots in this section with dis on the x-axis should use these values as predictors.

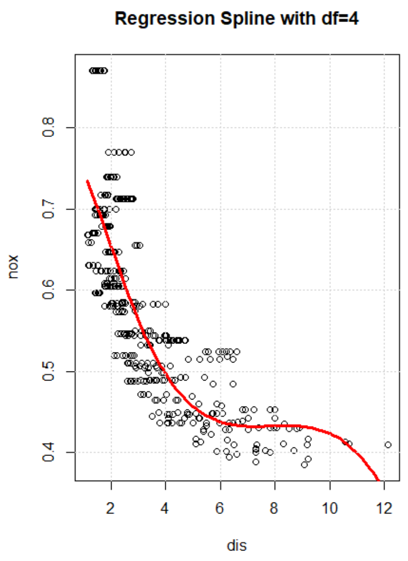
b) Plot the polynomial fits for polynomial degrees from 1 to 10, and report the associated residual sums of squares for the training error in a table. My plot looks like this (include a legend):



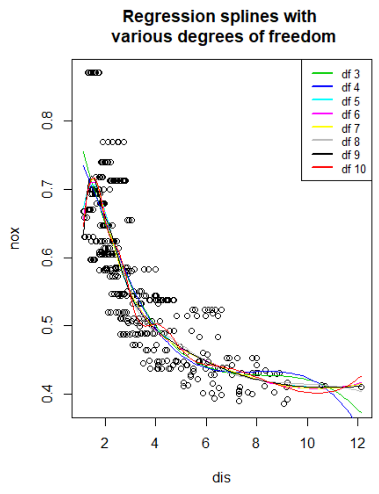
c) Set the seed to 5082, then perform cross-validation (recall cv.glm) to select the optimal degree (from 1 to 10) for the polynomial, determine the degree with the minimum c.v. error and plot the original data points and the fit resulting from the optimal degree. My plot looks like this:



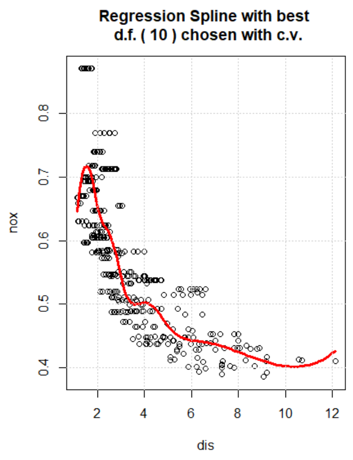
d) Use the bs() function to fit a regression spline to predict nox using dis – request 4 degrees of freedom. Report the output for the fit using summary(). Answer the following questions:
   i. How were the knots chosen?
   ii. Where was the knot placed?
      i. Recall that the attr(bs(…),'knots') function returns the position of the knots
   iii. Plot the resulting fit. Mine looks like this:



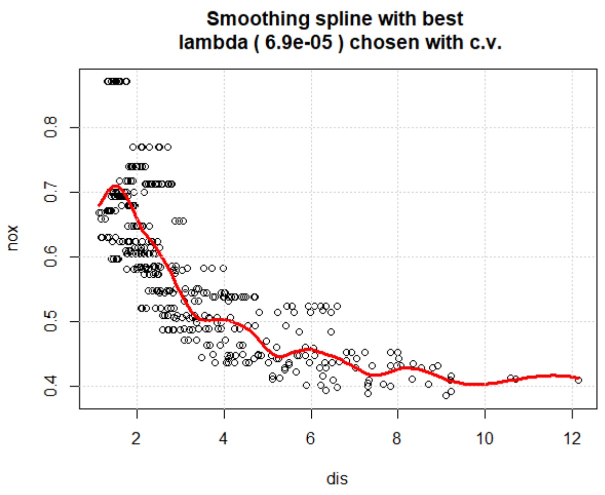
e) Now fit a regression spline for a range of degrees of freedom from 3 to 10, plot the resulting fits and report the associated residual sum of squares in a table. My plot looks like this:



f) Set the seed to 5082, then perform 10-fold cross-validation in order to select the best degrees of freedom (from 3 to 10) for a regression spline on this data. Plot your results, including your best degrees of freedom in the chart title. Mine looks like this:



g) Set the seed to 5082, then perform 10-fold cross-validation in order to select the best λ (from 3 to 10) for a smoothing spline on this data. You are still interested in nox as a function of dis. Plot your results, including your best λ in the chart title.
   a. Some notes:
      i. I had to use the jitter function on the dis variable to avoid warning messages from smooth.spline().
      ii. Recall that setting cv=T as an argument to the smooth.spline() function causes cross validation to be performed automatically. In this case, only the 'best' model (i.e. lowest training cv error) is produced.
      iii. The 'best' lambda is a named element of the smooth fit object.
      iv. The predict() function for smooth splines produces a list containing two named data frames – the first, named x, is a data frame containing the original x data and the second, named y, is a data frame with one variable – the predictions.
      v. Mine plot looks like this:


Smoothing spline with best
lambda (6.9e-05) chosen with c.v.
nox
0.4
0.5
0.6
0.7
0.8
dis
2
4
6
8
10
12