Homework #11

Justin Robinette

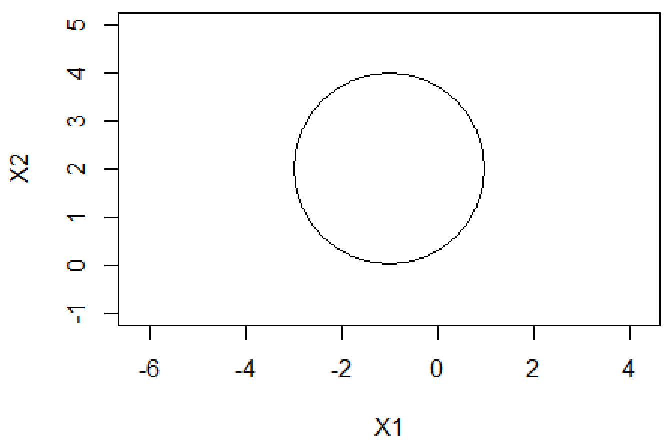
April 9, 2019

No collaborators for any problem

Question 9.7.2, pg 368: We have seen that in p=2 dimensions, a linear decision boundary takes the form $\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$. We now investigate a non-linear decision boundary.

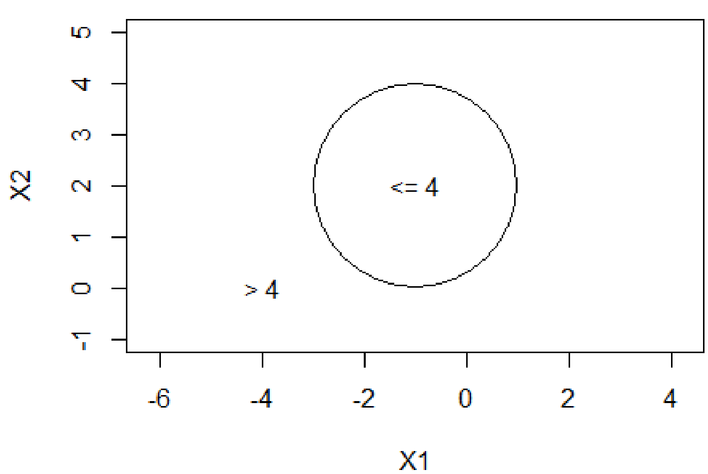
Part A: Sketch the curve $(1 + X_2)^2 + (2 - X_2)^2 = 4$

Results: First I plotted the curve given above.



Part B: On your sketch, indicate the set of points for which $(1 + X_2)^2 + (2 - X_2)^2 > 4$ as well as the set of points for which $(1 + X_2)^2 + (2 - X_2)^2 <= 4$.

Results: I took the plot from above and added text indicating the values that fall inside and outside of the boundary.



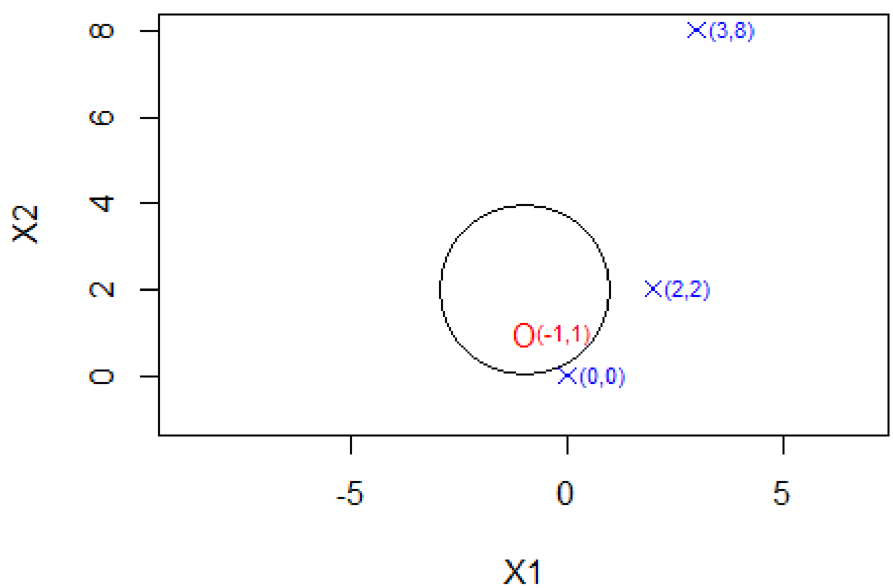
Part C: Suppose that a classifier assigns an observation to the blue class if $(1+X_1)^2+(2-X_2)^2>4$ and to the red class otherwise. To what class are the following observations classified?
- (0,0)
- (-1,1)
- (2,2)
- (3,8)

Results: First, I calculated the values of each of the points given the formula provided. As we can see, 3 of the 4 observations should fall outside of our curve based on the table shown. Only values that are less than or equal to 4 will show up in the curve.

Last, I plotted our curve and added the points. As we can see, the blue points (x's) are shown to fall outside the curve with the red point falling inside the circle. The coordinates of the points are shown.

Values of Supplied Observations

(0,0)	(-1,1)	(2,2)	(3,8)
5	1	9	52



Part D: Argue that while the decision boundary in (c) is not linear in terms of X_1 and X_2 , it is linear in terms of X_1, X_1^2, X_2, X_2^2.

Results: Using algebra, we can expand the equation so that it is linear in terms of X_1, X_1^2, X_2, X_2^2.

$$(1+X_1)^2+(2-X_2)^2>4$$

$$1+2X_1+X_1^2+4-4X_2+X_2^2>4$$

$$X_1^2+X_2^2+2X_1-4X_2+5>4$$

$$X_1^2+X_2^2+2X_1-4X_2+1>0$$

Question 9.7.7, pg 371: In this problem, you will use support vector approaches in order to predict whether a given car gets high or low gas mileage based on the **auto** data set.

Part A: Create a binary variable that takes on a 1 for cars with gas mileage above the median, and a 0 for cars with gas mileage below the median.

Results: I loaded the data set and created a factor variable (**mileage**) that is '1' if the vehicle's mpg is above the median. Otherwise, the factor variable is '0'. I printed the first 3 rows to show the new binary variable.

```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
##                        name mileage
## 1 chevrolet chevelle malibu       0
## 2         buick skylark 320       0
## 3        plymouth satellite       0
```

Part B: Fit a support vector classifier to the data with various values of *cost*, in order to predict whether a car gets high or low gas mileage. Report the cross-validation errors associated with different values of this parameter. Comment on your results.

Results: I fit a support vector classifier using various values of cost. As we can see from the table below, the error is lowest (**0.0126282**) when cost = 1. Error is highest (**0.0764103**) when cost = 0.01.

CV Error by Cost

Cost	CV Error	Dispersion
1e-02	0.0764103	0.0563826
1e-01	0.0534615	0.0472335
1e+00	0.0126282	0.0177802
5e+00	0.0202564	0.0197951
1e+01	0.0202564	0.0197951
1e+02	0.0355769	0.0173202

Part C: Now repeat (b), this time using SVMs with radial and polynomial basis kernels, with different values of *gamma* and *degree* and *cost*. Comment on your results.

Results: Here, I repeatd the steps from the previous exercise, but using radial and polynomial for my kernel with varying gamma and degree values, respectively. I also included the same range of cost values from the prior exercise for comparison.

The first table below shows the error rates for the radial kernel. As we can see, the error is lowest (**0.06115385**) when cost = 1, gamma = 1 and degree = 2.

The second table below shows the error rates for the polynomial kernel. As we can see, the error rate is lowest (**0.0405128**) when cost = 1, gamma = 1 and degree = 3.

The third table summarizes these error rates.

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma degree
##     1     1      2
##
## - best performance: 0.06115385


##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma degree
##     1     1      3
##
## - best performance: 0.04051282
```

CV Error by Parameters with Radial Kernel

Cost	Gamma	Degree	CV Error	Dispersion
1e-02	1	2	0.5740385	0.0307948
1e-01	1	2	0.5740385	0.0307948
1e+00	1	2	0.0611538	0.0528458
5e+00	1	2	0.0612179	0.0500978
1e+01	1	2	0.0612179	0.0500978
1e+02	1	2	0.0612179	0.0500978
1e-02	2	2	0.5740385	0.0307948
1e-01	2	2	0.5740385	0.0307948
1e+00	2	2	0.1124359	0.0596642
5e+00	2	2	0.1047436	0.0561378
1e+01	2	2	0.1047436	0.0561378
1e+02	2	2	0.1047436	0.0561378
1e-02	3	2	0.5740385	0.0307948
1e-01	3	2	0.5740385	0.0307948
1e+00	3	2	0.4825000	0.0879647
5e+00	3	2	0.4596795	0.1044295
1e+01	3	2	0.4596795	0.1044295
1e+02	3	2	0.4596795	0.1044295
1e-02	4	2	0.5740385	0.0307948
1e-01	4	2	0.5740385	0.0307948
1e+00	4	2	0.5128846	0.0528452
5e+00	4	2	0.5026923	0.0564958
1e+01	4	2	0.5026923	0.0564958
1e+02	4	2	0.5026923	0.0564958
1e-02	1	3	0.5740385	0.0307948
1e-01	1	3	0.5740385	0.0307948
1e+00	1	3	0.0611538	0.0528458
5e+00	1	3	0.0612179	0.0500978
1e+01	1	3	0.0612179	0.0500978
1e+02	1	3	0.0612179	0.0500978
1e-02	2	3	0.5740385	0.0307948
1e-01	2	3	0.5740385	0.0307948
1e+00	2	3	0.1124359	0.0596642
5e+00	2	3	0.1047436	0.0561378
1e+01	2	3	0.1047436	0.0561378
1e+02	2	3	0.1047436	0.0561378
1e-02	3	3	0.5740385	0.0307948
1e-01	3	3	0.5740385	0.0307948
1e+00	3	3	0.4825000	0.0879647
5e+00	3	3	0.4596795	0.1044295
1e+01	3	3	0.4596795	0.1044295
1e+02	3	3	0.4596795	0.1044295
1e-02	4	3	0.5740385	0.0307948
1e-01	4	3	0.5740385	0.0307948
1e+00	4	3	0.5128846	0.0528452
5e+00	4	3	0.5026923	0.0564958
1e+01	4	3	0.5026923	0.0564958
1e+02	4	3	0.5026923	0.0564958
1e-02	1	4	0.5740385	0.0307948
1e-01	1	4	0.5740385	0.0307948
1e+00	1	4	0.0611538	0.0528458
5e+00	1	4	0.0612179	0.0500978
1e+01	1	4	0.0612179	0.0500978
1e+02	1	4	0.0612179	0.0500978
1e-02	2	4	0.5740385	0.0307948
1e-01	2	4	0.5740385	0.0307948
1e+00	2	4	0.1124359	0.0596642
5e+00	2	4	0.1047436	0.0561378
1e+01	2	4	0.1047436	0.0561378
1e+02	2	4	0.1047436	0.0561378
1e-02	3	4	0.5740385	0.0307948
1e-01	3	4	0.5740385	0.0307948
1e+00	3	4	0.4825000	0.0879647
5e+00	3	4	0.4596795	0.1044295
1e+01	3	4	0.4596795	0.1044295
1e+02	3	4	0.4596795	0.1044295
1e-02	4	4	0.5740385	0.0307948
1e-01	4	4	0.5740385	0.0307948
1e+00	4	4	0.5128846	0.0528452
5e+00	4	4	0.5026923	0.0564958
1e+01	4	4	0.5026923	0.0564958
1e+02	4	4	0.5026923	0.0564958

CV Error by Parameters with Polynomial Kernel

Cost	Gamma	Degree	CV Error	Dispersion
1e-02	1	2	0.2525641	0.0735297
1e-01	1	2	0.1403205	0.0274412
1e+00	1	2	0.1454487	0.0437617
5e+00	1	2	0.1607692	0.0716203
1e+01	1	2	0.1760897	0.0731144
1e+02	1	2	0.1760897	0.0731144
1e-02	2	2	0.1558333	0.0432580
1e-01	2	2	0.1402564	0.0297765
1e+00	2	2	0.1633333	0.0665226
5e+00	2	2	0.1760897	0.0731144
1e+01	2	2	0.1760897	0.0731144
1e+02	2	2	0.1760897	0.0731144
1e-02	3	2	0.1454487	0.0342073
1e-01	3	2	0.1505769	0.0332407
1e+00	3	2	0.1760897	0.0731144
5e+00	3	2	0.1760897	0.0731144
1e+01	3	2	0.1760897	0.0731144
1e+02	3	2	0.1760897	0.0731144
1e-02	4	2	0.1402564	0.0211739
1e-01	4	2	0.1531410	0.0593882
1e+00	4	2	0.1760897	0.0731144
5e+00	4	2	0.1760897	0.0731144
1e+01	4	2	0.1760897	0.0731144
1e+02	4	2	0.1760897	0.0731144
1e-02	1	3	0.0560256	0.0446333
1e-01	1	3	0.0482051	0.0449682
1e+00	1	3	0.0405128	0.0427871
5e+00	1	3	0.0405128	0.0427871
1e+01	1	3	0.0405128	0.0427871
1e+02	1	3	0.0405128	0.0427871
1e-02	2	3	0.0457051	0.0389757
1e-01	2	3	0.0405128	0.0427871
1e+00	2	3	0.0405128	0.0427871
5e+00	2	3	0.0405128	0.0427871
1e+01	2	3	0.0405128	0.0427871
1e+02	2	3	0.0405128	0.0427871
1e-02	3	3	0.0430769	0.0425646
1e-01	3	3	0.0405128	0.0427871
1e+00	3	3	0.0405128	0.0427871
5e+00	3	3	0.0405128	0.0427871
1e+01	3	3	0.0405128	0.0427871
1e+02	3	3	0.0405128	0.0427871
1e-02	4	3	0.0405128	0.0427871
1e-01	4	3	0.0405128	0.0427871
1e+00	4	3	0.0405128	0.0427871
5e+00	4	3	0.0405128	0.0427871
1e+01	4	3	0.0405128	0.0427871
1e+02	4	3	0.0405128	0.0427871
1e-02	1	4	0.1735256	0.0477563
1e-01	1	4	0.1784615	0.0532863
1e+00	1	4	0.1732051	0.0549314
5e+00	1	4	0.1732051	0.0549314
1e+01	1	4	0.1732051	0.0549314
1e+02	1	4	0.1732051	0.0549314
1e-02	2	4	0.1783333	0.0650281
1e-01	2	4	0.1732051	0.0549314
1e+00	2	4	0.1732051	0.0549314
5e+00	2	4	0.1732051	0.0549314
1e+01	2	4	0.1732051	0.0549314
1e+02	2	4	0.1732051	0.0549314
1e-02	3	4	0.1732051	0.0549314
1e-01	3	4	0.1732051	0.0549314
1e+00	3	4	0.1732051	0.0549314
5e+00	3	4	0.1732051	0.0549314
1e+01	3	4	0.1732051	0.0549314
1e+02	3	4	0.1732051	0.0549314
1e-02	4	4	0.1732051	0.0549314
1e-01	4	4	0.1732051	0.0549314
1e+00	4	4	0.1732051	0.0549314
5e+00	4	4	0.1732051	0.0549314
1e+01	4	4	0.1732051	0.0549314
1e+02	4	4	0.1732051	0.0549314

Best Error Rate by Kernel

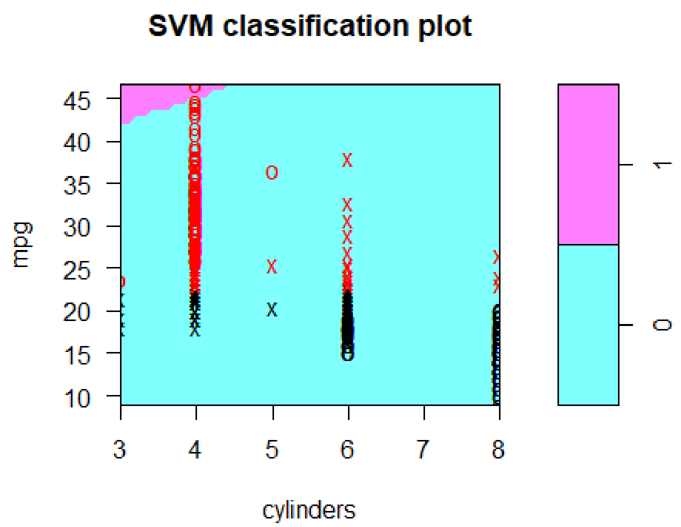
Radial	Polynomial
0.0611538	0.0405128

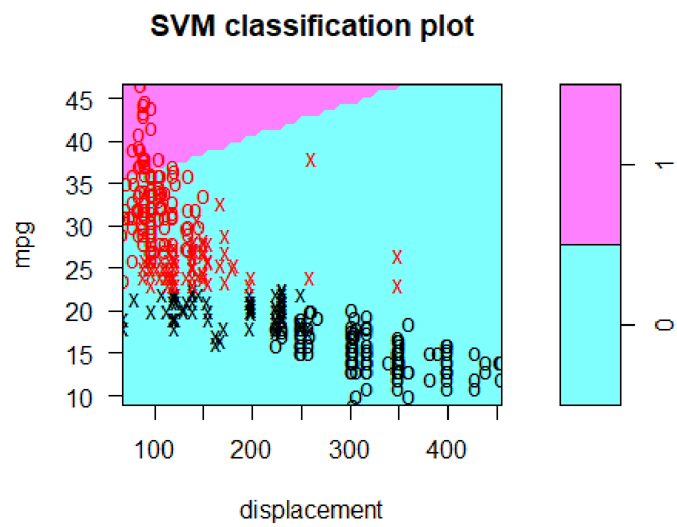
Part D: Make some plots to back up your assertions in (b) and (c).

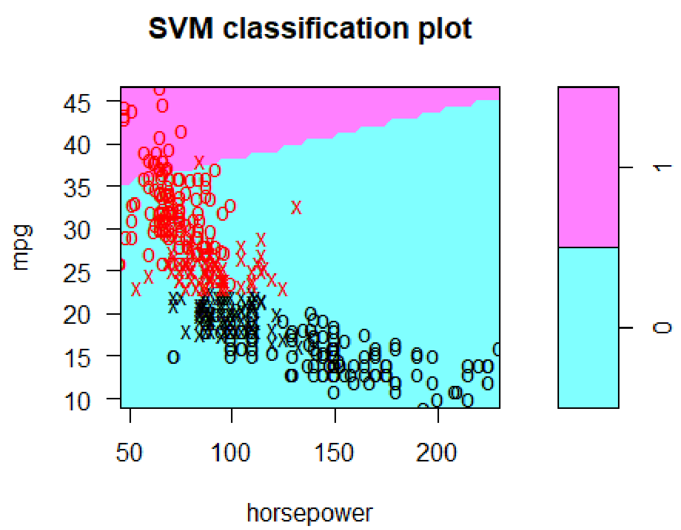
Results: I've plotted each of the three models with **mpg** versus the predictors in **Auto** with the classification from the models. To do so, I created a function with a for loop to improve efficiency.

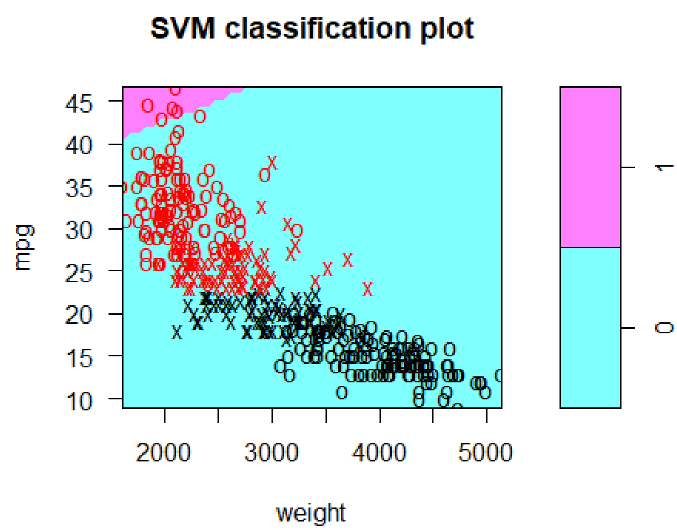
From the plots below, we can see that the Linear and Polynomial plots of **mpg~cylinders** are quite similar. The same can be said for the displacement, horsepower and weight plots with the Linear and Polynomial methods. The radial plots, for the most part, are quite different than the other two methods. This makes some sense because the Linear and Polynomial methods produced superior performing models, by error rate, versus the Radial SVM.

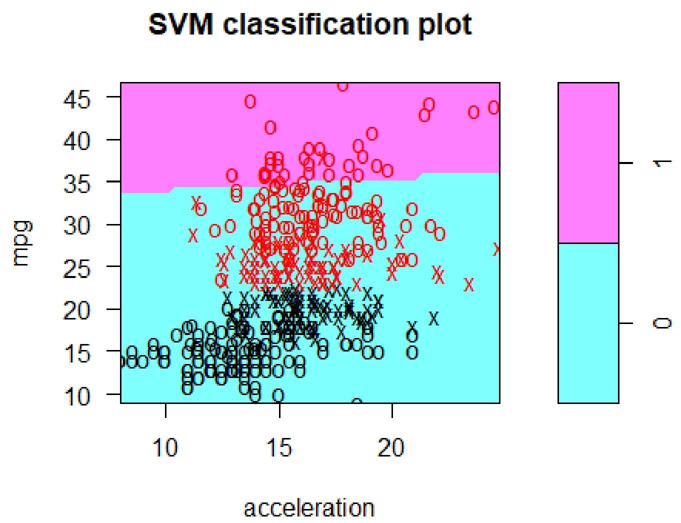
```
## [1] "Linear SVM Classification Plots"
```

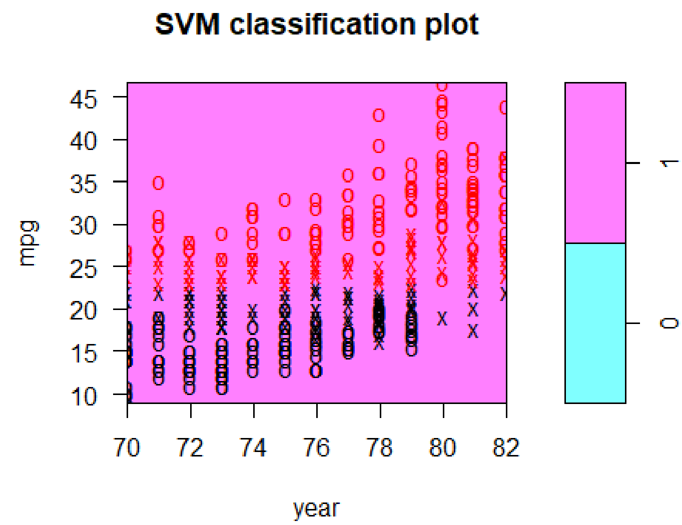


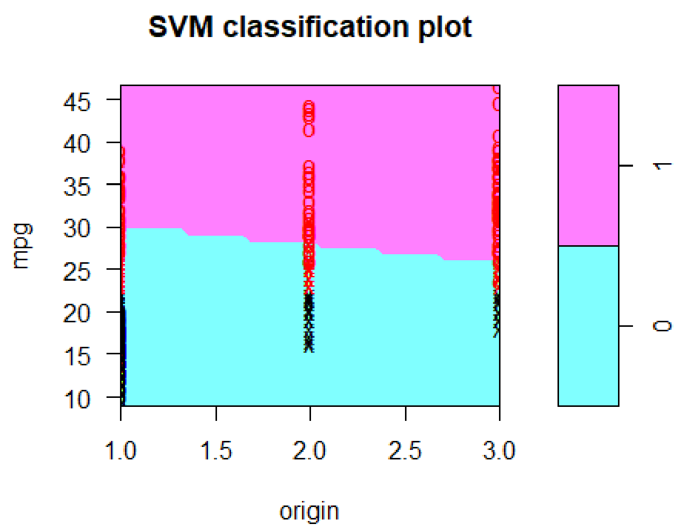




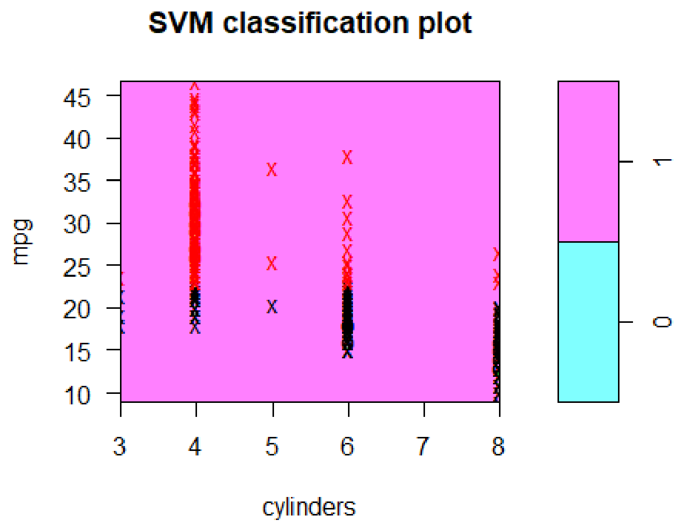


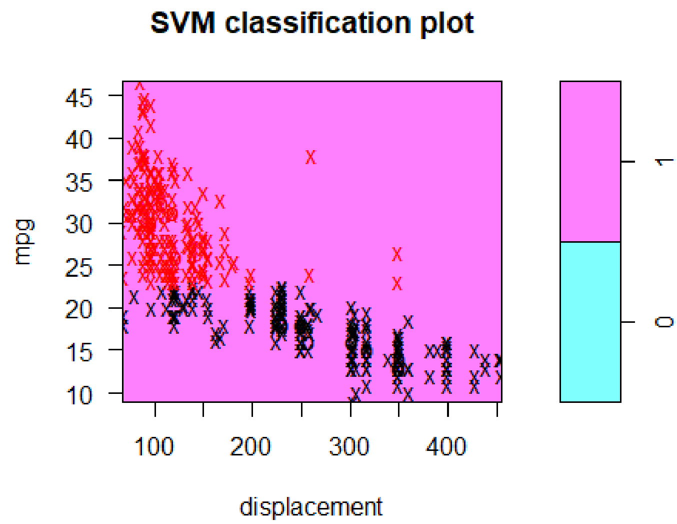

SVM classification plot
mpg
acceleration
1
0

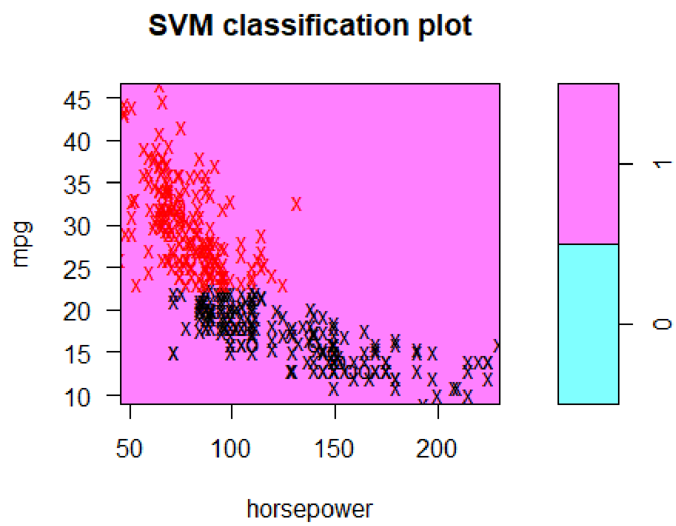

SVM classification plot
mpg
year
1
0

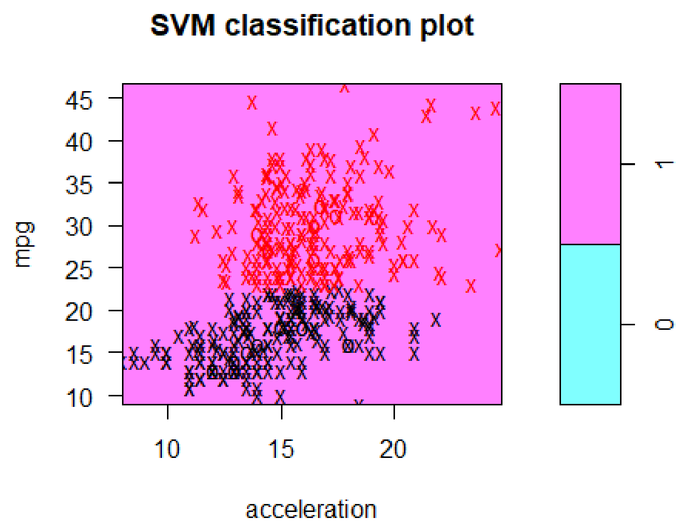

SVM classification plot
mpg
origin
1
0

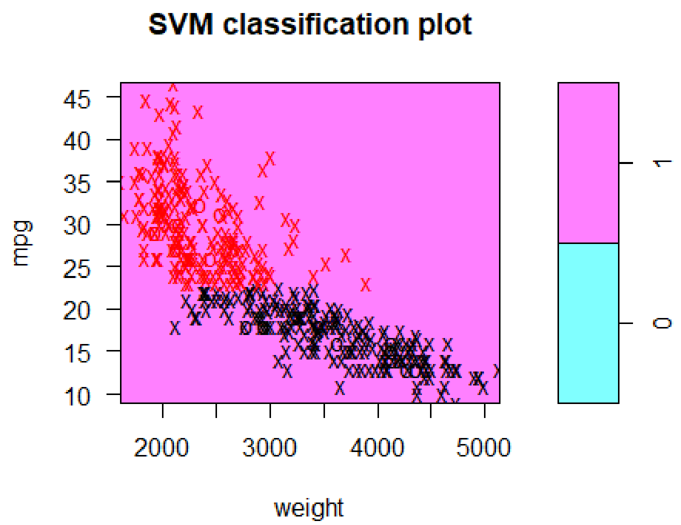
```
## [1] "Radial SVM Classification Plots"
```

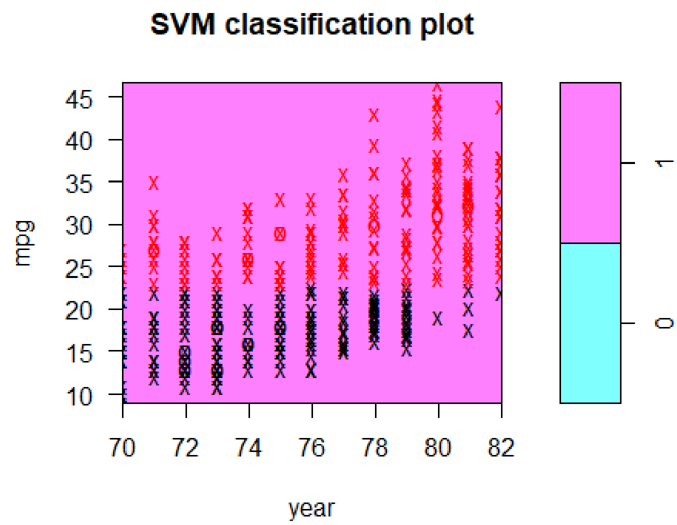


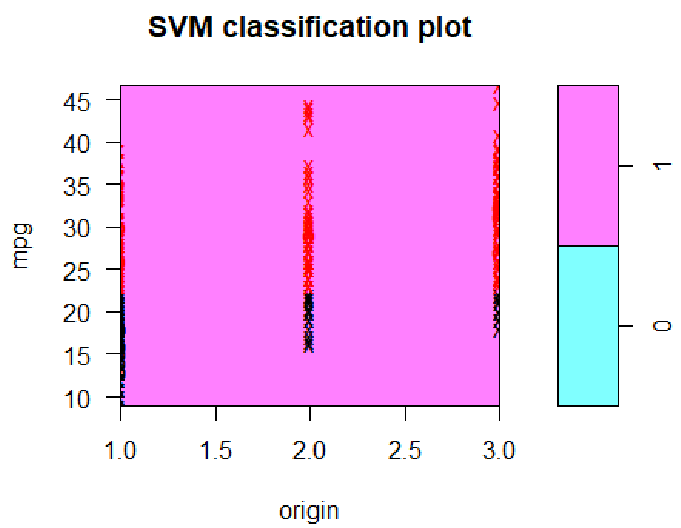




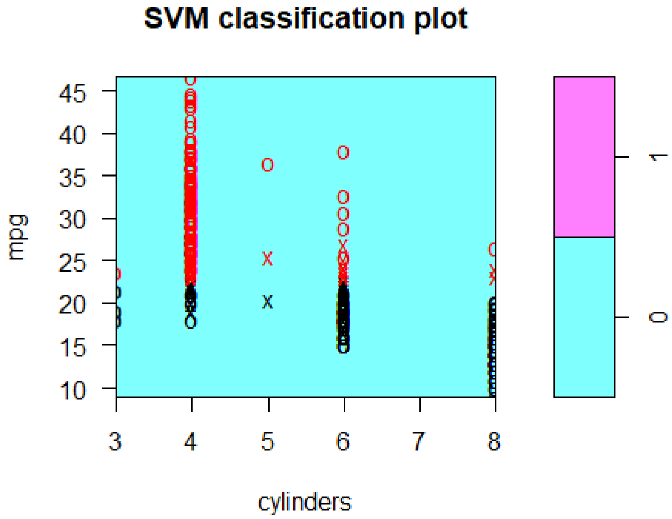


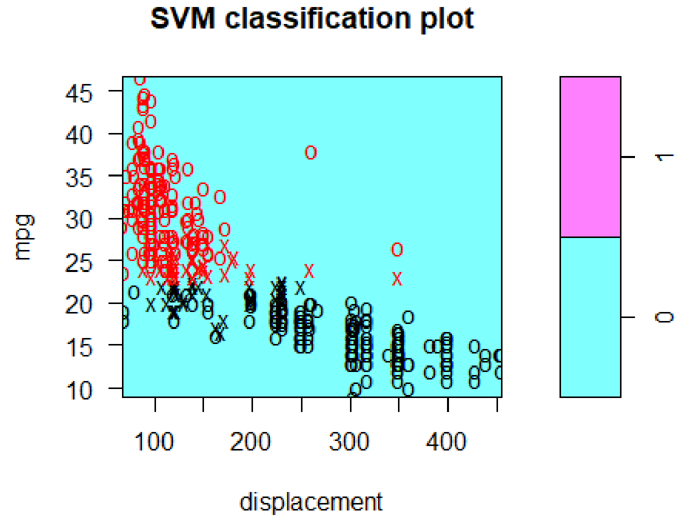

SVM classification plot
mpg
weight
2000
3000
4000
5000
10
15
20
25
30
35
40
45
0
1

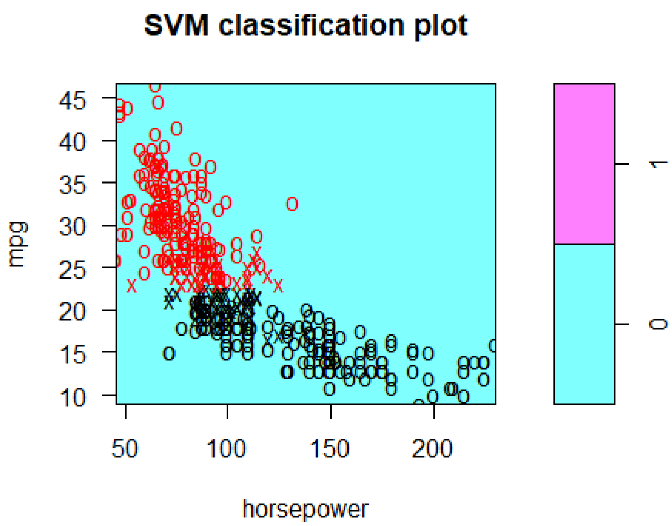

SVM classification plot
mpg
year
70
72
74
76
78
80
82
10
15
20
25
30
35
40
45
0
1

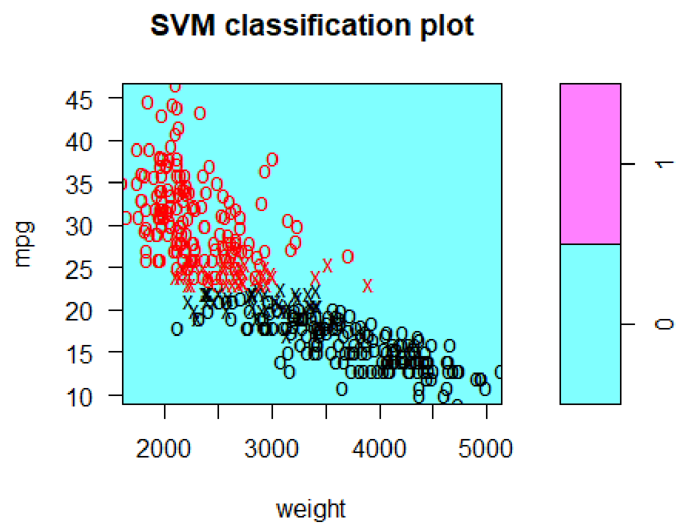

SVM classification plot
mpg
origin
1.0
1.5
2.0
2.5
3.0
10
15
20
25
30
35
40
45
0
1

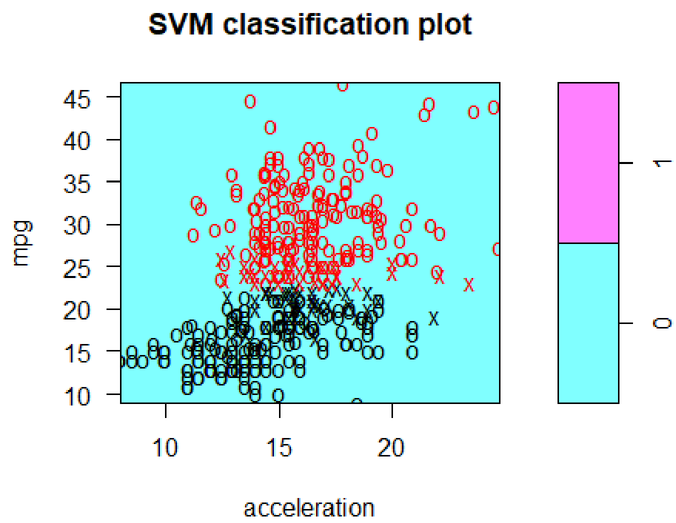
```
## [1] "Polynomial SVM Classification Plots"
```

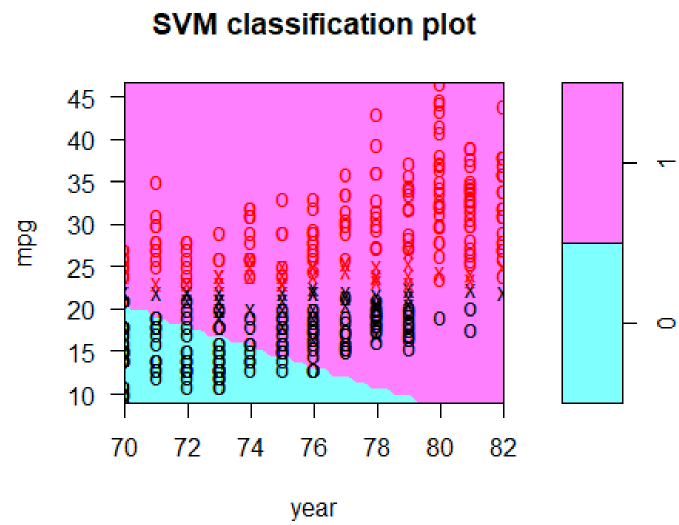


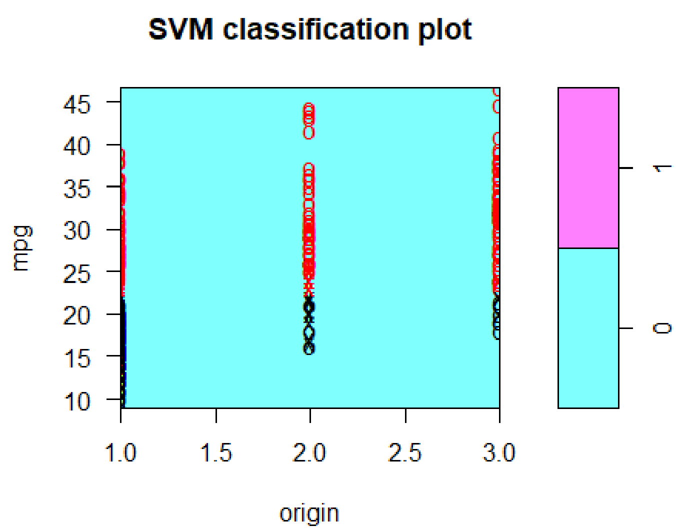












Question 9.7.8, pg 371: This problem involves the **OJ** data set.

Part A: Create a training set containing a random sample of 800 observations, and a test set containing the remaining observations.

Results: Here I've split the sets per the instructions and added a table showing the breakdown.

# of Obs

OJ	Training	Test
1070	800	270

Part 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 **summary()** to produce summary statistics and describe the results obtained.

Results: I fit a classifier based on the instructions. Looking at the summary, we see that, from the 800 training obs, the classifier created 446 Support Vectors. From these Support Vectors, 224 belong to the Level **CH** and 222 belong to the Level **MM**.

```
##
## Call:
## svm(formula = Purchase ~ ., data = oj.train, kernel = "linear",
##     cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##       gamma:  0.05555556
##
## Number of Support Vectors:  446
##
##  ( 224 222 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

Part C: What are the training and test error rates?

Results: The training error rate is **0.1688** and the test error rate is **0.1556**.

```
##         Predicted
## Observed  CH  MM
##       CH 428  56
##       MM  79 237

##         Predicted
## Observed  CH  MM
##       CH 146  23
##       MM  19  82
```

OJ Train and Test Error Rates

Train Error	Test Error
0.1688	0.1556

Part D: Use the **tune()** function to select an optimal *cost*. Consider values ranging from 0.01 to 10.

Results: Using the tune function we see that the optimal *cost* parameter is 1.

```
## [1] "The optimal cost value is: 1"
```

Part E: Compute the training and test error rates using the new value for *cost*.

Results: Fitting a new model using this cost parameter, we find that both the training error and test error rates decreased when changing the cost parameter from 0.01 to 1. The error rates from both training and test as well as both cost values are represented in the table below.

OJ Error Rates - Linear Kernel

Training Error	Test Error	Cost Value
0.1688	0.1556	0.01
0.1638	0.1519	1.00

Part F: Repeat parts (b) through (e) using a support vector machine with a radial kernel. Use the default value for *gamma*.

Results: First, from the summary of the radial kernel SVM with the cost parameter from part B, we see that the classifier created 634 Support Vectors. From these Support Vectors, 318 belong to the Level **CH** and 316 belong to the Level **MM**.

Next, I used tune to select the optimal cost value. As we can see from the output, the optimal cost parameter value is again 1. Using this parameter value, we can fit a new radial kernel SVM and make updated predictions on the training and test set.

Lastly, we compare the training and test error rates of the un-tuned SVM and the tuned SVM. As we can see, we the cost value is arbitrarily set at 0.01, the training error and test error are much higher than we saw in part c with the linear approach. After tuning, however, we see that the training and test errors actually improve from the linear approach when using radial for the kernel parameter and cost = 1.

```
##
## Call:
## svm(formula = Purchase ~ ., data = oj.train, kernel = "radial",
##     cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.01
##       gamma:  0.05555556
##
## Number of Support Vectors:  634
##
##  ( 318 316 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM

## [1] "The optimal cost value is: 1"
```

OJ Error Rates - Radial Kernel

Training Error	Test Error	Cost Value
0.3950	0.3741	0.01
0.1525	0.1370	1.00

Part G: Repeat parts (b) through (e) using a support vector machine with a polynomial kernel. Set *degree = 2*.

Results: First, from the summary of the polynomial kernel SVM with the cost parameter from part B and degree = 2, we see that the classifier created 635 Support Vectors. From these Support Vectors, 319 belong to the Level **CH** and 316 belong to the Level **MM**.

Next, I used tune to select the optimal cost value. As we can see from the output, the optimal cost parameter value is 10. Using this parameter value, we can fit a new polynomial kernel SVM and make updated predictions on the training and test set.

Lastly, we compare the training and test error rates of the un-tuned SVM and the tuned SVM. As we can see, we the cost value is arbitrarily set at 0.01, the training error and test error are much higher than we saw in part c with the linear approach. After tuning, however, we see that the training error improves and the test error gets worse when going from the linear approach to polynomial with cost = 10.

```
##
## Call:
## svm(formula = Purchase ~ ., data = oj.train, kernel = "polynomial",
##     cost = 0.01, degree = 2)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  0.01
##      degree:  2
##       gamma:  0.05555556
##      coef.0:  0
##
## Number of Support Vectors:  635
##
##  ( 319 316 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM

## [1] "The optimal cost value is: 10"
```

OJ Error Rates - Poylnomial Kernel

Training Error	Test Error	Cost Value
0.3938	0.3741	0.01
0.1588	0.1667	10.00

Part H: Overall, which approach seems to give the best results on this data?

Results: Based on the three summary tables below, we can see that once the cost parameter is optimized, the radial basis kernel is producing the best misclassification rate in both the training and test sets. The polynomial basis kernel is performing the worst on the test set and the linear approach is performing the worse on the training set.

OJ Error Rates - Linear Kernel

Training Error	Test Error	Cost Value
0.1688	0.1556	0.01
0.1638	0.1519	1.00

OJ Error Rates - Radial Kernel

Training Error	Test Error	Cost Value
0.3950	0.3741	0.01
0.1525	0.1370	1.00

OJ Error Rates - Poylnomial Kernel

Training Error	Test Error	Cost Value
0.3938	0.3741	0.01
0.1588	0.1667	10.00

Question 4: In the past couple of homework assignments you have used different classification methods to analyze the dataset you chose. For this homework, use a support vector machine to model your data. Find the test error using any/all methods. Compare the results you obtained with the result from previous homework. Did the results improve? Use the table with the previous results to compare.

Results: Here I performed SVM using linear, radial, and polynomial kernels. I optimized some cost, gamma, and degree parameters for the various methods.

Using these results, we can see that the Linear SVM performs the best using the VSA with a misclassification rate of **12.0192%** - which ties for the best performing method on this data set.

In the LOOCV approach, the Linear SVM again performed the best, and again tied the best performing method on this data set with a misclassification rate on the test set of **14.4928%**. The same is true for the 5-Fold CV approach.

Across the board, each of these SVMs performed very well in relation to the other models that I have tried this semester on this data set.

Test Error by Validation Approach (%)

Method	VSA	LOOCV	5-Fold CV
Logistic Reg	**12.0192**	**14.4928**	**14.4928**
KNN	16.3462	18.6957	18.5507
LDA	**12.0192**	**14.4928**	**14.4928**
QDA	16.8269	17.3913	17.5362
MclustDA	16.8269	20	17.5362
MclustDA (EDDA)	16.8269	17.3913	17.5362
Neural Network	**12.0192**	14.6377	**14.4928**
Tree	**12.0192**	**14.4928**	**14.4928**
Bagging	17.7885	20.4348	20.2899
Random Forest	13.9423	14.7826	14.7826
Boosting	16.8269	18.8406	**14.4928**
Linear SVM	**12.0192**	**14.4928**	**14.4928**
Radial SVM	13.4615	14.9275	14.9275
Polynomial SVM	12.5	14.6377	14.6377