Q37

Rahul Atre

2023-11-21

Q37 [Classification]

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.

1. 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.

First, lets load in the data for Auto.csv:

```
auto_df = read.csv("auto.csv")
mpg01 = ifelse(auto_df$mpg > median(auto_df$mpg), 1, 0)
auto_df$mpg01 = as.factor(mpg01)
```

2. 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.

```
set.seed(144)

support_vector_model = tune(svm, mpg01 ~ cylinders + displacement + horsepower + weight + acceleration +
summary(support_vector_model)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##    10
##
## - best performance: 0.08668016
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-03 0.12001687 0.04807884
## 2 5e-02 0.10483131 0.05499372
## 3 1e+00 0.09168016 0.03580886
## 4 5e+00 0.08924426 0.02954293
```

```
## 5 1e+01 0.08668016 0.03176062
## 6 5e+01 0.08668016 0.03176062
## 7 1e+02 0.08668016 0.03176062
```

From the above function call, we can see that the best cost is 0.001 and has an error of 0.1128195.

3. Now repeat 2., this time using SVMs with radial and polynomial basis kernels, with different values of gamma and degree and cost. Comment on your results.

```
set.seed(144)

poly_svm_model = tune(svm, mpg01 ~ cylinders + displacement + horsepower + weight + acceleration + year
summary(poly_svm_model)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
##    10      3
##
## - best performance: 0.071417
##
## - Detailed performance results:
##     cost degree      error dispersion
## 1  1e-03      2 0.47730769 0.06772068
## 2  5e-02      2 0.35966937 0.12057354
## 3  5e+00      2 0.18117409 0.02828383
## 4  1e+01      2 0.18117409 0.03064089
## 5  1e+02      2 0.19380567 0.03569396
## 6  1e-03      3 0.45948718 0.08917494
## 7  5e-02      3 0.23695682 0.08020970
## 8  5e+00      3 0.08417679 0.03628926
## 9  1e+01      3 0.07141700 0.03146198
## 10 1e+02      3 0.08154184 0.04481199
## 11 1e-03      4 0.47730769 0.06772068
## 12 5e-02      4 0.31106613 0.05661811
## 13 5e+00      4 0.18848516 0.05061486
## 14 1e+01      4 0.17052632 0.06244988
## 15 1e+02      4 0.12469636 0.06747540
```

From the above function call, the best cost is 10 with degree 3, and has an error of 0.1143509.

Let us now use the radial kernel:

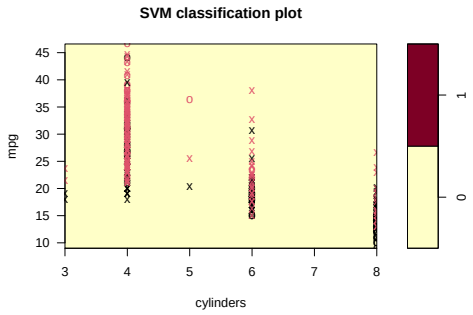
```
set.seed(144)

radial_model = tune(svm, mpg01 ~ cylinders + displacement + horsepower + weight + acceleration + year +
summary(radial_model)
```

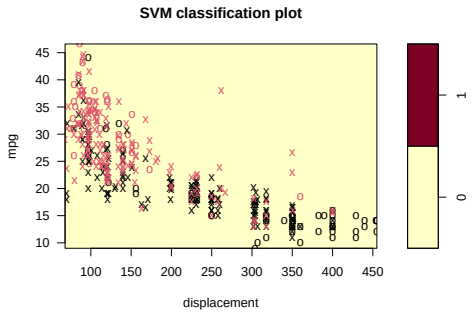
```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     1     1
##
## - best performance: 0.06859312
##
## - Detailed performance results:
##     cost gamma      error dispersion
## 1   0.01  0.01 0.47730769 0.06772068
## 2   0.50  0.01 0.10213225 0.04955525
## 3   1.00  0.01 0.10476383 0.05193436
## 4   4.00  0.01 0.09970310 0.05881525
## 5   5.00  0.01 0.09970310 0.06358966
## 6  20.00  0.01 0.08668016 0.04317073
## 7  50.00  0.01 0.07904858 0.04175808
## 8   0.01  0.50 0.47730769 0.06772068
## 9   0.50  0.50 0.08154858 0.04425724
## 10  1.00  0.50 0.07628543 0.04760394
## 11  4.00  0.50 0.07615722 0.05547883
## 12  5.00  0.50 0.08122132 0.05259236
## 13 20.00  0.50 0.09397773 0.04091827
## 14 50.00  0.50 0.10186910 0.04097864
## 15  0.01  1.00 0.47730769 0.06772068
## 16  0.50  1.00 0.07628543 0.04911454
## 17  1.00  1.00 0.06859312 0.04600475
## 18  4.00  1.00 0.08128543 0.04574275
## 19  5.00  1.00 0.07878543 0.04679550
## 20 20.00  1.00 0.09154521 0.04758236
## 21 50.00  1.00 0.09680837 0.04404380
```

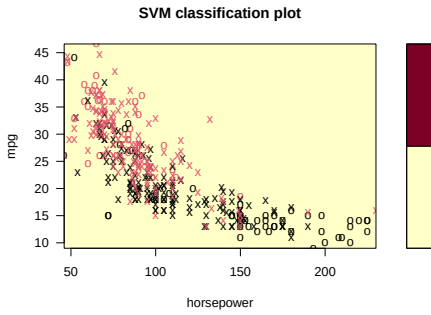
For the radial kernel, we obtain that the best cost is 1 with gamma = 0.5. It has an error of 0.08907398.

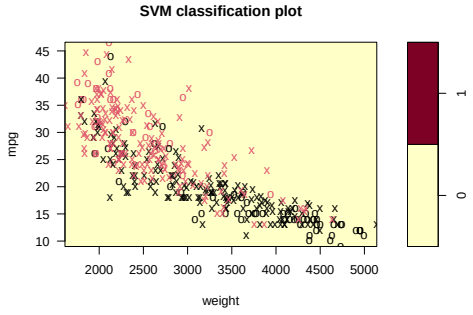
4. Make some plots to back up your assertions in 2. and 3.

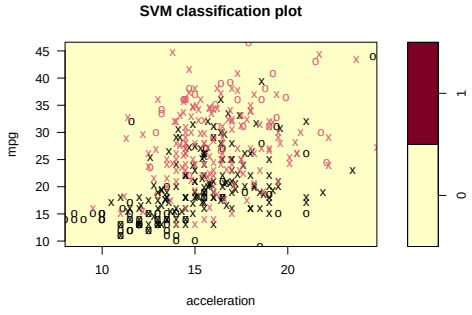
```
linear_svm = svm(mpg01 ~ cylinders + displacement + horsepower + weight + acceleration + year + origin,

poly_svm = svm(mpg01 ~ cylinders + displacement + horsepower + weight + acceleration + year + origin, da

radial_svm = svm(mpg01 ~ cylinders + displacement + horsepower + weight + acceleration + year + origin,

plotpairs = function(model) {
    for(name in names(auto_df)[!(names(auto_df) %in% c("mpg", "name", "mpg01"))]) {
        plot(model, auto_df, as.formula(paste("mpg~", name, sep = "")))
    }
}
plotpairs(linear_svm)
```

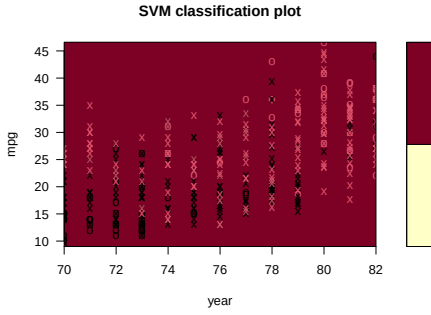


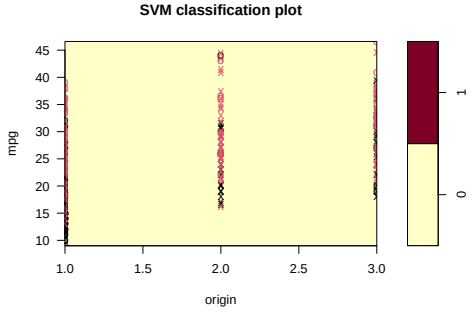




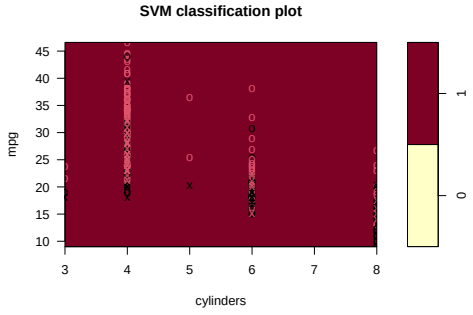


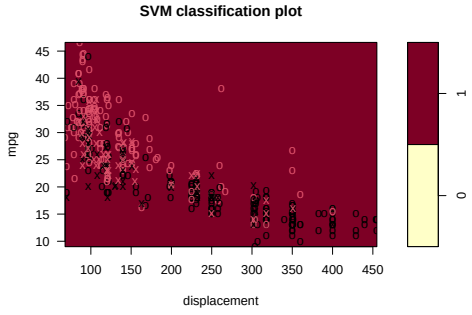


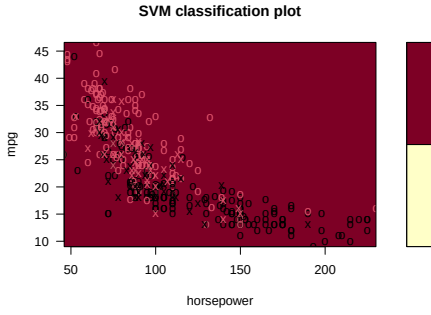


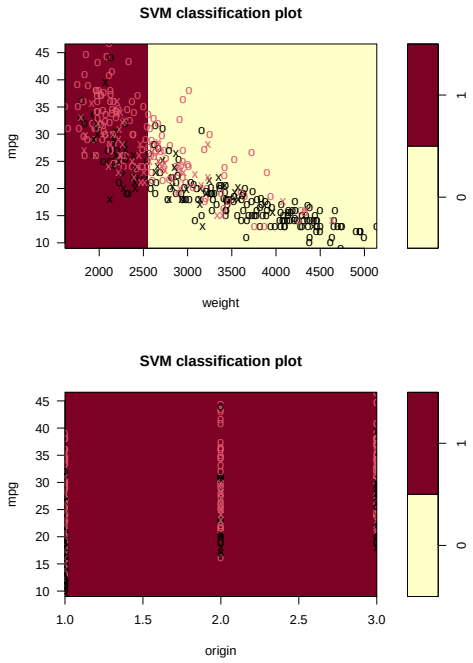


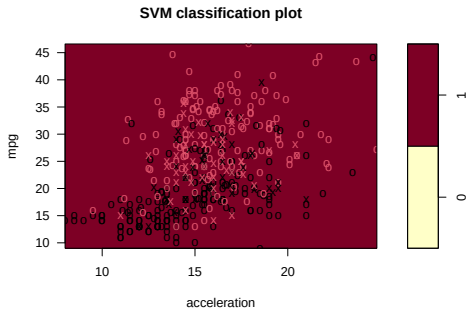

```
plotpairs(poly_svm)
```

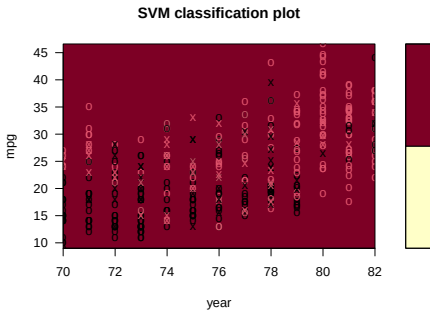




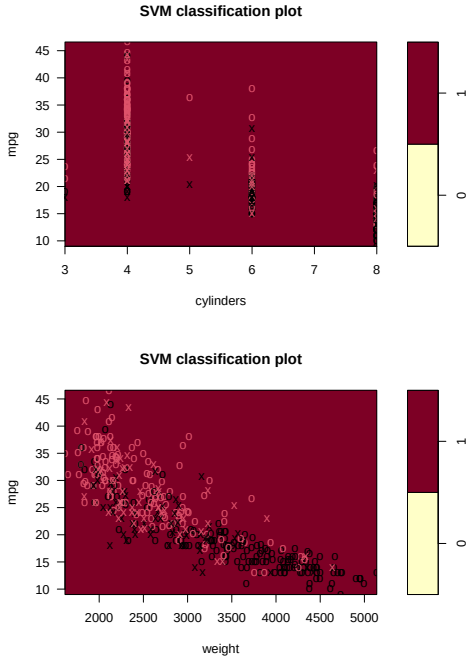


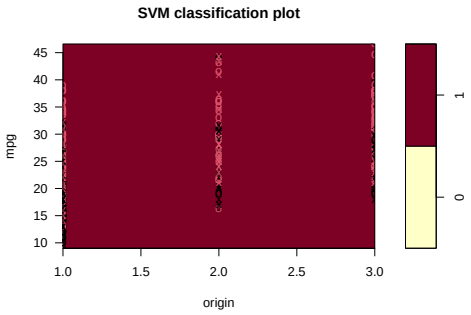







```
plotpairs(radial_svm)
```




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