P8106_HW5

Naomi Simon-Kumar ns3782

23/11/2025

Contents

Loading libraries 1

Question 1. Support Vector Machines **1**

Partition into training and testing set 1

a) Fit support vector classifier 2

b) Fit support vector machine with radial kernel 6

References **6**

Loading libraries

```
# Load libraries
library(tidyverse)
library(caret)
library(ggplot2)
library(tidymodels)
library(e1071)
```

Question 1. Support Vector Machines

Partition into training and testing set

```
# Read in dataset
auto <- read.csv("auto.csv")

# Remove NAs
auto <- na.omit(auto)

# Make sure factor variables are correctly coded
auto$cylinders <- factor(auto$cylinders)
auto$origin <- factor(auto$origin)
```

```r
auto$mpg_cat <- factor(auto$mpg_cat, levels = c("low", "high"))

# Check variable types
str(auto)
```

```
## 'data.frame':    392 obs. of  8 variables:
##  $ cylinders   : Factor w/ 5 levels "3","4","5","6",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : int  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : int  3504 3693 3436 3433 3449 4341 4354 4312 4425 3850 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : int  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
##  $ mpg_cat     : Factor w/ 2 levels "low","high": 1 1 1 1 1 1 1 1 1 1 ...
```

```r
levels(auto$mpg_cat)
```

```
## [1] "low"  "high"
```

```r
# Set seed for reproducibility
set.seed(299)

# Split data into training and testing data
data_split_auto <- initial_split(auto, prop = 0.7)


# Extract the training and test data
training_data_auto <- training(data_split_auto)
testing_data_auto <- testing(data_split_auto)

# Check variable types
# str(training_data_auto)
# str(testing_data_auto)
```

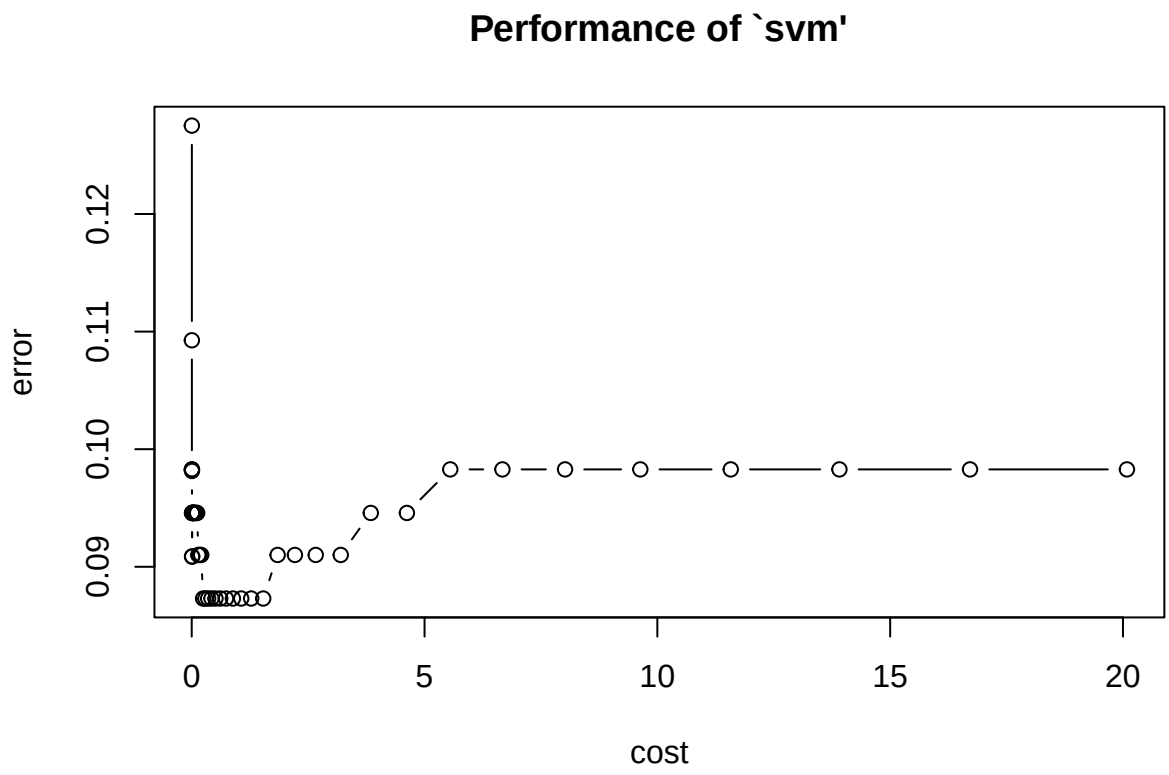
I made sure to recode the variables origin and cylinders to factor variable type. Although cylinders was originally represented as an integer, it is a multi-valued discrete variable as its values represent categorical groupings of engine types (i.e., 4, 6, 8 cylinder),

a) Fit support vector classifier

```r
# Set seed for reproducibility
set.seed(299)

# Fit model
linear.tune <- tune.svm(mpg_cat ~ . ,
                        data = training_data_auto,
                        kernel = "linear",
                        cost = exp(seq(-6,3, len = 50)),
                        scale = TRUE)
```

```r
# Tuning curve
plot(linear.tune)
```

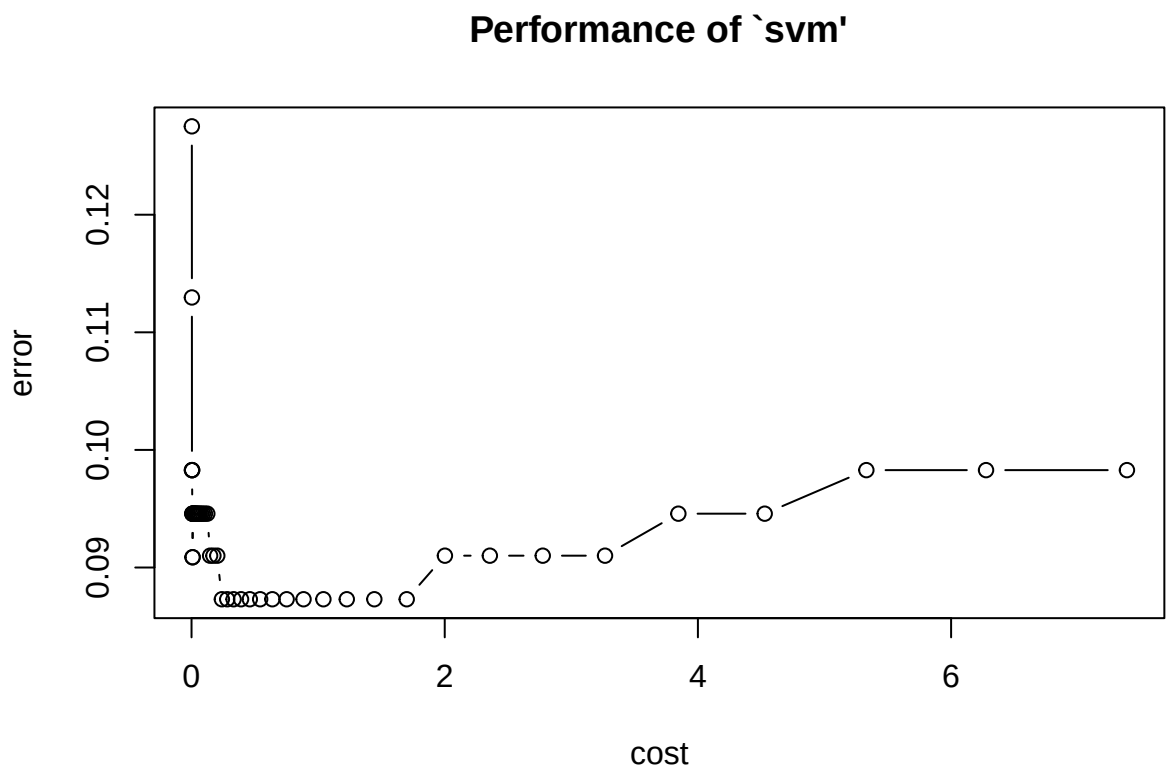


I initially proceeded with exploring a wide grid for the cost tuning parameter, from exp(-6) to exp(3). However, the plot shows that accuracy (i.e., 1-Misclassification Error) stabilises quite early, around cost = 1, and increasing cost beyond that does not notably improve performance. Therefore, I decided on reducing the size of the cost tuning parameter grid.

```r
# Set seed for reproducibility
set.seed(299)

# Fit model
linear.tune.2 <- tune.svm(mpg_cat ~ . ,
                          data = training_data_auto,
                          kernel = "linear",
                          cost = exp(seq(-6,2, len = 50)),
                          scale = TRUE)

# Tuning curve
plot(linear.tune.2)
```




```
# Optimal parameters
linear.tune.2$best.parameters
```

```
##        cost
## 29 0.239651
```

I refined the tuning parameter grid to cover values in a range between exp(-6) and exp(2), which appears to be appropriate. The **best cost tuning parameter is 0.239651** which is within this range and not at the edge of the grid boundaries. This is also confirmed by the plotted tuning curve, where the minimum cross validation accuracy (1-Misclassification error) is around 0.08.

Next, finding the training error:

```
# Set seed for reproducibility
set.seed(299)

# get the best model
best.linear <- linear.tune.2$best.model

# Training Error
pred_train <- predict(best.linear, newdata = training_data_auto)
confusionMatrix(data = pred_train,
                reference = training_data_auto$mpg_cat)
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction low high
##       low  120   10
##       high  13  131
##
##                Accuracy : 0.9161
##                  95% CI : (0.8767, 0.946)
##     No Information Rate : 0.5146
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8319
##
##  Mcnemar's Test P-Value : 0.6767
##
##             Sensitivity : 0.9023
##             Specificity : 0.9291
##          Pos Pred Value : 0.9231
##          Neg Pred Value : 0.9097
##              Prevalence : 0.4854
##          Detection Rate : 0.4380
##    Detection Prevalence : 0.4745
##       Balanced Accuracy : 0.9157
##
##        'Positive' Class : low
##
```

```r
# Calculating training misclassification error

1-0.9161 # 0.0839
```

```
## [1] 0.0839
```

```r
# Test Error
pred_test <- predict(best.linear, newdata = testing_data_auto)
confusionMatrix(data = pred_test,
                reference = testing_data_auto$mpg_cat)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##       low   55    5
##       high   8   50
##
##                Accuracy : 0.8898
##                  95% CI : (0.819, 0.94)
##     No Information Rate : 0.5339
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7794
##
##  Mcnemar's Test P-Value : 0.5791
```

```
##
##             Sensitivity : 0.8730
##             Specificity : 0.9091
##          Pos Pred Value : 0.9167
##          Neg Pred Value : 0.8621
##              Prevalence : 0.5339
##          Detection Rate : 0.4661
##    Detection Prevalence : 0.5085
##       Balanced Accuracy : 0.8911
##
##        'Positive' Class : low
##
```

```
# Calculating test misclassification error

1-0.8898 # 0.1102
```

```
## [1] 0.1102
```

Based on the model, the training misclassification error is **0.0839 (8.39%)** with **accuracy = 0.9161**, which shows that there is strong model performance (i.e., the model is a good fit to the training data). **The Kappa statistic is 0.8319**, which indicates an excellent level of agreement between predicted, and observed mpg categories beyond chance (McHugh, 2012).

The test misclassification error is **0.1102 (11.02%)** with **accuracy = 0.8898**, which indicates that our model generalises well to unseen data. **The Kappa statistic is 0.7794**, indicating substantial agreement between predicted and observed mpg categories, beyond what would be expected due to chance (McHugh, 2012).

b) Fit support vector machine with radial kernel

References

McHugh, M. L. (2012). Interrater reliability: the kappa statistic. Biochemia medica, 22(3), 276-282.