Homework 4.1 - Auto dataset

Hamed

2/26/2020

# 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 in the ISLR library.

#Load libraries and dataset  
library(ISLR)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(mlr)

## Loading required package: ParamHelpers

## 'mlr' is in maintenance mode since July 2019. Future development  
## efforts will go into its successor 'mlr3' (<https://mlr3.mlr-org.com>).

library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

##   
## Attaching package: 'caret'

## The following object is masked from 'package:mlr':  
##   
## train

library(ROCR)

## Loading required package: gplots

##   
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':  
##   
## lowess

##   
## Attaching package: 'ROCR'

## The following object is masked from 'package:mlr':  
##   
## performance

library(kernlab)

##   
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':  
##   
## alpha

library(e1071)

##   
## Attaching package: 'e1071'

## The following object is masked from 'package:mlr':  
##   
## impute

library(foreach)  
library(doParallel)

## Loading required package: iterators

## Loading required package: parallel

Auto=as.data.frame(Auto)

## (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.

med=median(Auto$mpg)  
Auto$target=ifelse(Auto$mpg>med,'1','0')  
  
# convert target to factor and drop the string variable for classification purposes  
  
Auto$target=as.factor(as.character(Auto$target)) #convert to factor  
class(Auto$target)

## [1] "factor"

Auto=select(Auto,-starts\_with('name')) #drop name  
  
# Change to fatcor and perform Variable encoding on the origin variable because its a categorical variable  
Auto$origin=as.factor(as.character(Auto$origin))  
Auto=createDummyFeatures(Auto, cols = "origin")  
str(Auto)

## 'data.frame': 392 obs. of 11 variables:  
## $ mpg : num 18 15 18 16 17 15 14 14 14 15 ...  
## $ cylinders : num 8 8 8 8 8 8 8 8 8 8 ...  
## $ displacement: num 307 350 318 304 302 429 454 440 455 390 ...  
## $ horsepower : num 130 165 150 150 140 198 220 215 225 190 ...  
## $ weight : num 3504 3693 3436 3433 3449 ...  
## $ acceleration: num 12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...  
## $ year : num 70 70 70 70 70 70 70 70 70 70 ...  
## $ target : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ origin.1 : num 1 1 1 1 1 1 1 1 1 1 ...  
## $ origin.2 : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ origin.3 : num 0 0 0 0 0 0 0 0 0 0 ...

## (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-validated training dataset error associated with different values of this parameter. Comment on your results.

# split the dataset  
set.seed(123)  
smp\_size<- floor(0.85 \* nrow(Auto))  
train\_ind<-sample(seq\_len(nrow(Auto)), size = smp\_size)  
train.Auto<-Auto[train\_ind, ]  
test.Auto<-Auto[-train\_ind, ]  
  
# Fit the model  
SVM1=tune(svm,target~.,data = train.Auto,ranges=list(cost=c(.01,.02,.05,  
 .1,.2,.5,1,2,5,10)),kernel='linear',  
 trControl = trainControl("cv", number = 5))  
  
SVM1$best.model

##   
## Call:  
## best.tune(method = svm, train.x = target ~ ., data = train.Auto,   
## ranges = list(cost = c(0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2,   
## 5, 10)), kernel = "linear", trControl = trainControl("cv",   
## number = 5))  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: linear   
## cost: 10   
##   
## Number of Support Vectors: 22

summary(SVM1) #The error reduces as the model cross validates towards 10 folds

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 10-fold cross validation   
##   
## - best parameters:  
## cost  
## 10  
##   
## - best performance: 0.009090909   
##   
## - Detailed performance results:  
## cost error dispersion  
## 1 0.01 0.071746881 0.05095783  
## 2 0.02 0.065864528 0.05222247  
## 3 0.05 0.068894831 0.05275986  
## 4 0.10 0.059893048 0.05093626  
## 5 0.20 0.039215686 0.04287966  
## 6 0.50 0.033155080 0.03319105  
## 7 1.00 0.039304813 0.03516316  
## 8 2.00 0.030213904 0.04040502  
## 9 5.00 0.018092692 0.02113321  
## 10 10.00 0.009090909 0.01463775

## (c) Now repeat (b), this time using SVMs with radial and polynomial basis kernels, with different values of gamma (sigma) and degree and cost. Comment on your results.

# RADIAL KERNEL  
SVM2 = tune(svm ,target~.,data=train.Auto,ranges=list(  
 cost=c(.01,.02,.05,.1,.2,.5,1,2,5,10),gamma=c(.001,.002,.005,  
 .01,.02,.05,.1,.2,.5,1,2,5,10)),kernel='radial',  
 trControl = trainControl("cv", number = 10),  
 preProcess = c("center","scale"))  
  
SVM2$best.model

##   
## Call:  
## best.tune(method = svm, train.x = target ~ ., data = train.Auto,   
## ranges = list(cost = c(0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2,   
## 5, 10), gamma = c(0.001, 0.002, 0.005, 0.01, 0.02, 0.05,   
## 0.1, 0.2, 0.5, 1, 2, 5, 10)), kernel = "radial", trControl = trainControl("cv",   
## number = 10), preProcess = c("center", "scale"))  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: radial   
## cost: 10   
##   
## Number of Support Vectors: 55

summary(SVM2)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 10-fold cross validation   
##   
## - best parameters:  
## cost gamma  
## 10 0.02  
##   
## - best performance: 0.0271836   
##   
## - Detailed performance results:  
## cost gamma error dispersion  
## 1 0.01 1e-03 0.49759358 0.17098990  
## 2 0.02 1e-03 0.49759358 0.17098990  
## 3 0.05 1e-03 0.49759358 0.17098990  
## 4 0.10 1e-03 0.49759358 0.17098990  
## 5 0.20 1e-03 0.14714795 0.06585783  
## 6 0.50 1e-03 0.09313725 0.04146754  
## 7 1.00 1e-03 0.08707665 0.03879037  
## 8 2.00 1e-03 0.07807487 0.03825458  
## 9 5.00 1e-03 0.06916221 0.03792554  
## 10 10.00 1e-03 0.06916221 0.02874307  
## 11 0.01 2e-03 0.49759358 0.17098990  
## 12 0.02 2e-03 0.49759358 0.17098990  
## 13 0.05 2e-03 0.49759358 0.17098990  
## 14 0.10 2e-03 0.16221034 0.06843104  
## 15 0.20 2e-03 0.09304813 0.04587724  
## 16 0.50 2e-03 0.08707665 0.03879037  
## 17 1.00 2e-03 0.07807487 0.03825458  
## 18 2.00 2e-03 0.07219251 0.04094652  
## 19 5.00 2e-03 0.06916221 0.02874307  
## 20 10.00 2e-03 0.06622103 0.03130882  
## 21 0.01 5e-03 0.49759358 0.17098990  
## 22 0.02 5e-03 0.49759358 0.17098990  
## 23 0.05 5e-03 0.10819964 0.05203580  
## 24 0.10 5e-03 0.09010695 0.04506051  
## 25 0.20 5e-03 0.08404635 0.04212805  
## 26 0.50 5e-03 0.07807487 0.03825458  
## 27 1.00 5e-03 0.07219251 0.03561595  
## 28 2.00 5e-03 0.07219251 0.03561595  
## 29 5.00 5e-03 0.06622103 0.03130882  
## 30 10.00 5e-03 0.05721925 0.03314315  
## 31 0.01 1e-02 0.49759358 0.17098990  
## 32 0.02 1e-02 0.25499109 0.08217111  
## 33 0.05 1e-02 0.09010695 0.04506051  
## 34 0.10 1e-02 0.09010695 0.04506051  
## 35 0.20 1e-02 0.08110517 0.04051747  
## 36 0.50 1e-02 0.07513369 0.03851636  
## 37 1.00 1e-02 0.06916221 0.03792554  
## 38 2.00 1e-02 0.06622103 0.03130882  
## 39 5.00 1e-02 0.05730838 0.03020983  
## 40 10.00 1e-02 0.04830660 0.03268730  
## 41 0.01 2e-02 0.48244207 0.18328896  
## 42 0.02 2e-02 0.09001783 0.04914279  
## 43 0.05 2e-02 0.08707665 0.04817609  
## 44 0.10 2e-02 0.08110517 0.04051747  
## 45 0.20 2e-02 0.07807487 0.03825458  
## 46 0.50 2e-02 0.06916221 0.03792554  
## 47 1.00 2e-02 0.06916221 0.02874307  
## 48 2.00 2e-02 0.05721925 0.02672080  
## 49 5.00 2e-02 0.04224599 0.02565029  
## 50 10.00 2e-02 0.02718360 0.02234789  
## 51 0.01 5e-02 0.22495544 0.07340696  
## 52 0.02 5e-02 0.09010695 0.04506051  
## 53 0.05 5e-02 0.08413547 0.04698547  
## 54 0.10 5e-02 0.07807487 0.03825458  
## 55 0.20 5e-02 0.07816399 0.03568539  
## 56 0.50 5e-02 0.07219251 0.03280643  
## 57 1.00 5e-02 0.06622103 0.02416501  
## 58 2.00 5e-02 0.05436720 0.04001773  
## 59 5.00 5e-02 0.03930481 0.03213078  
## 60 10.00 5e-02 0.04536542 0.03575556  
## 61 0.01 1e-01 0.28778966 0.09380233  
## 62 0.02 1e-01 0.08716578 0.04401084  
## 63 0.05 1e-01 0.08716578 0.04401084  
## 64 0.10 1e-01 0.08110517 0.03512185  
## 65 0.20 1e-01 0.07816399 0.03568539  
## 66 0.50 1e-01 0.07522282 0.02976584  
## 67 1.00 1e-01 0.05418895 0.02798252  
## 68 2.00 1e-01 0.05133690 0.04063537  
## 69 5.00 1e-01 0.04233512 0.03560417  
## 70 10.00 1e-01 0.03930481 0.02878066  
## 71 0.01 2e-01 0.49759358 0.17098990  
## 72 0.02 2e-01 0.09304813 0.05001016  
## 73 0.05 2e-01 0.08716578 0.04401084  
## 74 0.10 2e-01 0.08716578 0.04401084  
## 75 0.20 2e-01 0.07816399 0.03568539  
## 76 0.50 2e-01 0.05713012 0.03019346  
## 77 1.00 2e-01 0.05427807 0.04007822  
## 78 2.00 2e-01 0.05427807 0.04909868  
## 79 5.00 2e-01 0.05124777 0.03515011  
## 80 10.00 2e-01 0.04224599 0.03563874  
## 81 0.01 5e-01 0.49759358 0.17098990  
## 82 0.02 5e-01 0.46755793 0.17332828  
## 83 0.05 5e-01 0.08716578 0.04387626  
## 84 0.10 5e-01 0.08413547 0.04462899  
## 85 0.20 5e-01 0.07219251 0.03837390  
## 86 0.50 5e-01 0.06024955 0.04750853  
## 87 1.00 5e-01 0.05124777 0.04529650  
## 88 2.00 5e-01 0.06024955 0.05162541  
## 89 5.00 5e-01 0.06033868 0.04756952  
## 90 10.00 5e-01 0.05418895 0.03999081  
## 91 0.01 1e+00 0.49759358 0.17098990  
## 92 0.02 1e+00 0.49759358 0.17098990  
## 93 0.05 1e+00 0.25757576 0.12242499  
## 94 0.10 1e+00 0.09607843 0.04197852  
## 95 0.20 1e+00 0.08101604 0.04499386  
## 96 0.50 1e+00 0.06916221 0.04310795  
## 97 1.00 1e+00 0.05124777 0.04749562  
## 98 2.00 1e+00 0.05436720 0.05504402  
## 99 5.00 1e+00 0.05427807 0.04697467  
## 100 10.00 1e+00 0.05721925 0.05237158  
## 101 0.01 2e+00 0.50668449 0.14572563  
## 102 0.02 2e+00 0.50668449 0.14572563  
## 103 0.05 2e+00 0.50668449 0.14572563  
## 104 0.10 2e+00 0.34438503 0.13034449  
## 105 0.20 2e+00 0.12629234 0.04002225  
## 106 0.50 2e+00 0.07210339 0.04332698  
## 107 1.00 2e+00 0.06319073 0.05049885  
## 108 2.00 2e+00 0.06613191 0.04915141  
## 109 5.00 2e+00 0.06907308 0.04954391  
## 110 10.00 2e+00 0.06907308 0.04954391  
## 111 0.01 5e+00 0.51274510 0.12969134  
## 112 0.02 5e+00 0.51274510 0.12969134  
## 113 0.05 5e+00 0.51274510 0.12969134  
## 114 0.10 5e+00 0.51274510 0.12969134  
## 115 0.20 5e+00 0.49456328 0.14079893  
## 116 0.50 5e+00 0.18021390 0.06432883  
## 117 1.00 5e+00 0.09010695 0.03765988  
## 118 2.00 5e+00 0.09304813 0.03605974  
## 119 5.00 5e+00 0.09304813 0.03605974  
## 120 10.00 5e+00 0.09304813 0.03605974  
## 121 0.01 1e+01 0.52486631 0.10095281  
## 122 0.02 1e+01 0.52486631 0.10095281  
## 123 0.05 1e+01 0.52486631 0.10095281  
## 124 0.10 1e+01 0.52486631 0.10095281  
## 125 0.20 1e+01 0.52486631 0.10095281  
## 126 0.50 1e+01 0.45846702 0.12867755  
## 127 1.00 1e+01 0.19812834 0.08396423  
## 128 2.00 1e+01 0.17397504 0.07305213  
## 129 5.00 1e+01 0.17397504 0.07305213  
## 130 10.00 1e+01 0.17397504 0.07305213

# POLYNOMIAL  
  
SVM3 <- train(target ~., data = train.Auto, method = "svmPoly",  
 trControl = trainControl("cv", number = 10),  
 tuneGrid = expand.grid(C=c(.01,.02,.05,.1,.2,.5,1,2,5,10)  
 ,degree=c(1:5),scale=c(0.01:1)),  
 preProcess = c("center","scale"),  
 tuneLength = 4)  
  
SVM3$finalModel #displays cost , error, degree and scale of the model

## Support Vector Machine object of class "ksvm"   
##   
## SV type: C-svc (classification)   
## parameter : cost C = 10   
##   
## Polynomial kernel function.   
## Hyperparameters : degree = 3 scale = 0.01 offset = 1   
##   
## Number of Support Vectors : 57   
##   
## Objective Function Value : -386.4731   
## Training error : 0.018018

SVM3$results #displays the accuracy of the model crossvalidated

## C degree scale Accuracy Kappa AccuracySD KappaSD  
## 1 0.01 1 0.01 0.5075758 0.0000000 0.00798555 0.00000000  
## 6 0.02 1 0.01 0.5075758 0.0000000 0.00798555 0.00000000  
## 11 0.05 1 0.01 0.8950813 0.7894894 0.03173341 0.06389572  
## 16 0.10 1 0.01 0.9070243 0.8135216 0.03242268 0.06510458  
## 21 0.20 1 0.01 0.9129066 0.8252863 0.03819970 0.07670940  
## 26 0.50 1 0.01 0.9220031 0.8435636 0.03978340 0.07979590  
## 31 1.00 1 0.01 0.9310049 0.8616456 0.03954578 0.07935661  
## 36 2.00 1 0.01 0.9310049 0.8616456 0.03954578 0.07935661  
## 41 5.00 1 0.01 0.9281529 0.8560259 0.03693919 0.07389721  
## 46 10.00 1 0.01 0.9370655 0.8739129 0.03218206 0.06435307  
## 2 0.01 2 0.01 0.5075758 0.0000000 0.00798555 0.00000000  
## 7 0.02 2 0.01 0.8680704 0.7351296 0.03687497 0.07408586  
## 12 0.05 2 0.01 0.9070243 0.8135216 0.03242268 0.06510458  
## 17 0.10 2 0.01 0.9129066 0.8252863 0.03819970 0.07670940  
## 22 0.20 2 0.01 0.9190619 0.8376813 0.03676513 0.07373887  
## 27 0.50 2 0.01 0.9310049 0.8616456 0.03954578 0.07935661  
## 32 1.00 2 0.01 0.9310049 0.8616456 0.03954578 0.07935661  
## 37 2.00 2 0.01 0.9310049 0.8616456 0.03954578 0.07935661  
## 42 5.00 2 0.01 0.9370655 0.8739129 0.03218206 0.06435307  
## 47 10.00 2 0.01 0.9552529 0.9104220 0.03971853 0.07945079  
## 3 0.01 3 0.01 0.7478220 0.4911067 0.03464861 0.07255900  
## 8 0.02 3 0.01 0.8981116 0.7955891 0.03454240 0.06954739  
## 13 0.05 3 0.01 0.9099655 0.8194039 0.03418653 0.06866912  
## 18 0.10 3 0.01 0.9190619 0.8376813 0.03676513 0.07373887  
## 23 0.20 3 0.01 0.9250334 0.8496409 0.04232796 0.08491717  
## 28 0.50 3 0.01 0.9310049 0.8616456 0.03954578 0.07935661  
## 33 1.00 3 0.01 0.9340352 0.8677906 0.03611186 0.07229359  
## 38 2.00 3 0.01 0.9281529 0.8560259 0.03693919 0.07389721  
## 43 5.00 3 0.01 0.9522226 0.9043222 0.03965000 0.07933507  
## 48 10.00 3 0.01 0.9702317 0.9404411 0.03396490 0.06793024  
## 4 0.01 4 0.01 0.8470365 0.6925715 0.04467668 0.08998290  
## 9 0.02 4 0.01 0.8981116 0.7955891 0.03454240 0.06954739  
## 14 0.05 4 0.01 0.9129066 0.8252863 0.03819970 0.07670940  
## 19 0.10 4 0.01 0.9220031 0.8435636 0.03978340 0.07979590  
## 24 0.20 4 0.01 0.9280637 0.8557633 0.04214412 0.08450884  
## 29 0.50 4 0.01 0.9310049 0.8616456 0.03954578 0.07935661  
## 34 1.00 4 0.01 0.9310940 0.8619083 0.03393616 0.06791975  
## 39 2.00 4 0.01 0.9431261 0.8861125 0.03507553 0.07014116  
## 44 5.00 4 0.01 0.9581941 0.9163491 0.03964926 0.07929543  
## 49 10.00 4 0.01 0.9701370 0.9402684 0.04384919 0.08769847  
## 5 0.01 5 0.01 0.8770778 0.7531444 0.03784889 0.07634027  
## 10 0.02 5 0.01 0.9011419 0.8017569 0.03083533 0.06183184  
## 15 0.05 5 0.01 0.9159369 0.8314313 0.03634954 0.07286790  
## 20 0.10 5 0.01 0.9250334 0.8496409 0.04232796 0.08491717  
## 25 0.20 5 0.01 0.9280637 0.8557633 0.04214412 0.08450884  
## 30 0.50 5 0.01 0.9310049 0.8616456 0.03954578 0.07935661  
## 35 1.00 5 0.01 0.9310940 0.8619083 0.03393616 0.06791975  
## 40 2.00 5 0.01 0.9520388 0.9039770 0.03722469 0.07444716  
## 45 5.00 5 0.01 0.9643494 0.9286764 0.04335254 0.08670204  
## 50 10.00 5 0.01 0.9612244 0.9224487 0.04158756 0.08317513