-1960 output: pdf_document —

FE590. Assignment #4.

Enter Your Name Here, or "Anonymous" if you want to remain anonymous. Name: YANG YUE

2017-12-12

Instructions

When you have completed the assignment, knit the document into a PDF file, and upload *both* the .pdf and .Rmd files to Canvas.

Note that you must have LaTeX installed in order to knit the equations below. If you do not have it installed, simply delete the questions below.

Question 1:

In this assignment, you will be required to find a set of data to run regression on. This data set should be financial in nature, and of a type that will work with the models we have discussed this semester (hint: we didn't look at time series) You may not use any of the data sets in the ISLR package that we have been looking at all semester. Your data set that you choose should have both qualitative and quantitative variables. (or has variables that you can transform)

Provide a description of the data below, where you obtained it, what the variable names are and what it is describing.

The data set given the GDP change of the United States from 1960 to 2015.

There are 56 observations and 8 variable. [1] AWI: The average Wage index from 1960 to 2015 [2] GDP(US dollar): American GDP from 1960 to 2015. [3] Population, total: The total population of the United States from 1960 to 2015. [4] Exports of goods and services (US dollar): American exports of goods and service from 1960 to 2015. [5] Electric power consumption (kWh per capita): American electric power consuption per person from 1960 to 2015. [6] Household final consumption expenditure (US dollar): The total expense of American family from 1960 to 2015. [7] Unemployment rate: American unemployment rate from 1960 to 2015.

Source World Bank Open Data https://data.worldbank.org/

National Average Wage Index https://www.ssa.gov/oact/cola/AWI.html

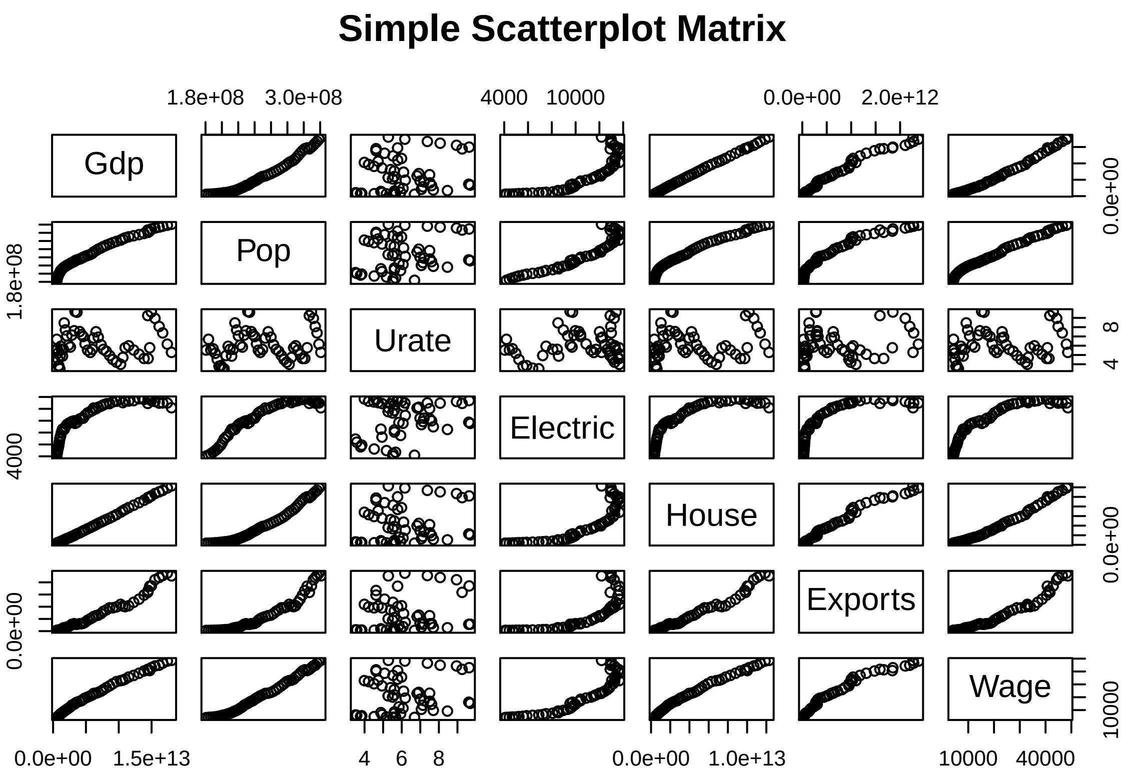
Question 2:

Pick a quantitative variable and fit at least four different models in order to predict that variable using the other predictors. Determine which of the models is the best fit. You will need to provide strong reason why the particular model you chose is the best one. You will need to confirm the model you have selected provides the best fit and that you have obtained the best version of that particular model (i.e. subset selection or validation for example). You need to convince the grader that you have chosen the best model.

```r
# Here we reanme the data, so we will have formatted data matrix.
library(readxl)
```

```
## Warning: package 'readxl' was built under R version 3.4.2
```

```r
USA_GDP <- read_excel("USA GDP.xlsx")
colnames(USA_GDP)=(c("Gdp","Pop","Exports","Electric","House","Urate","Wage"))
pairs(~Gdp + Pop+ Urate+ Electric+ House + Exports + Wage, data = USA_GDP,main="Simple Scatterplot Matri
```




```r
names(USA_GDP)
```

```
## [1] "Gdp"      "Pop"      "Exports"  "Electric" "House"    "Urate"
## [7] "Wage"
```

```r
attach(USA_GDP)
```

```r
# create training and testing data set
set.seed(1)
train=sample(1:nrow(USA_GDP), nrow(USA_GDP)/2)
Gdp.train=USA_GDP[train,]
Gdp.test=USA_GDP[-train,]


# Multiple linear Regression

fit1=glm(Gdp~., data=Gdp.train)
pre.fit1=predict(fit1,Gdp.test)
linear.fit=mean(pre.fit1-Gdp.test$Gdp)^2
```

```r
linear.fit
```

```
## [1] 1.064944e+19
```

```r
# Ridge Regression
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.4.3
```

```
## Loading required package: Matrix
```

```
## Warning: package 'Matrix' was built under R version 3.4.3
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-13
```

```r
x=model.matrix(Gdp ~., data =Gdp.train)
x.test=model.matrix(Gdp~., data=Gdp.test)
y=Gdp.train$Gdp
cv.out =cv.glmnet (x, y, alpha =0)
```

```
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations
## per fold
```

```r
bestlam =cv.out$lambda.min
bestlam
```

```
## [1] 675612359776
```

```r
ridge.mod=glmnet(x, y, alpha = 0)
ridge.pred=predict (ridge.mod ,s=bestlam ,newx=x.test)
ridge.fit=mean(( ridge.pred-Gdp.test$Gdp)^2)
ridge.fit
```

```
## [1] 1.293276e+23
```

```r
# Generalized additive model and using linear regression and smooth splines as basis function
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 3.4.3
```

```
##
## Attaching package: 'ISLR'
```

```
## The following object is masked from 'USA_GDP':
##
##     Wage
```

```r
library(splines)
library(boot)
library(gam)
```

```
## Warning: package 'gam' was built under R version 3.4.3
```

```
## Loaded gam 1.14-4
```

```r
# Find the best degree of freedom for Unemployment rateand Electirc that should be use in this model.
# we use 4 for Electric right here.
fit2=smooth.spline(Urate, Gdp,cv=TRUE)
```

```
## Warning in smooth.spline(Urate, Gdp, cv = TRUE): cross-validation with non-
## unique 'x' values seems doubtful
```

```
fit2$df
```

```
## [1] 2.002545
```

```
fit3=smooth.spline(Electric, Gdp,cv=TRUE)
fit3$df
```

```
## [1] 3.841074
```

```
gam1=gam(Gdp~s(Urate, 2)+ Pop + s(Electric,4)+House + Exports + Wage,data=Gdp.train)
pre.gam=predict (gam1, newdata =Gdp.test)
gam.fit=mean(pre.gam-Gdp.test$Gdp)^2
gam.fit
```

```
## [1] 1.945878e+18
```

```
# Bagging
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.2
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
bag.G =randomForest(Gdp~.,data=Gdp.train,mytry=7,importance=TRUE)
yhat.bag = predict (bag.G ,newdata =Gdp.test)
bag.fit=mean(( yhat.bag - Gdp.test)^2)
bag.fit
```

```
## [1] 3.936846e+25
```

```
# Tree regression
library(tree)
```

```
## Warning: package 'tree' was built under R version 3.4.2
```

```
tree.G=tree(Gdp ~ ., data=USA_GDP,subset = train)
yhat=predict(tree.G, newdata = Gdp.test)
mean((yhat - USA_GDP$Gdp)^2)
```

```
## [1] 2.938855e+25
```

```
cv.G =cv.tree(tree.G )
cv.G
```

```
## $size
## [1] 4 3 2 1
##
## $dev
## [1] 1.098330e+26 2.099373e+26 2.099373e+26 1.132060e+27
##
## $k
## [1]          -Inf 6.654622e+25 6.885291e+25 8.889541e+26
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
#Prune the tree for best MSE
prune.G=prune.tree(tree.G,best = 2)
yhat.prune=predict(prune.G, newdata = Gdp.test)
tree.fit=mean((yhat.prune - Gdp.test$Gdp)^2)
tree.fit
```

```
## [1] 7.329178e+24
```

```
data.frame(Model=c("tree.fit","bag.fit","gam.fit","ridge.fit","linear.fit"),MSE=c(tree.fit,bag.fit,gam.
```

```
##        Model          MSE
## 1   tree.fit 7.329178e+24
## 2    bag.fit 3.936846e+25
## 3    gam.fit 1.945878e+18
## 4  ridge.fit 1.293276e+23
## 5 linear.fit 1.064944e+19
```

In thsi question, I picked Gdp as dependent variable and other variable as my predictors.I have used Multiply linear regression, Ridge regression, Generalized additive model, Tree regression and Bagging for predicting GDP.From the scatterplot, we can see that the relationship between Gdp and other predictors is mostly linear. Therefore, I assume that multiple linear regression and Generalized additive will work the best in this question. And the proformance of GAM should work better than multiple linear regression's since I select Regression splines, linear regression as my basis function in GAM. Compare the MSE we got above, the result accords with my assumption.

Question 3:

Do the same approach as in question 2, but this time for a qualitative variable.

```
# Again,here we reanme the data, so we will have formatted data matrix.
GDP_Q <- read_excel("GDP Q.xlsx")
names(GDP_Q)
```

```
## [1] "GDPQ"
## [2] "PopulationQ"
## [3] "Exports of goods and servicesQ"
## [4] "Electric power consumptionQ"
## [5] "Household final consumption expenditureQ"
## [6] "Unemployment rateQ"
## [7] "AWIQ"
```

```
colnames(GDP_Q)=(c("Gdpq","Popq","Exportsq","Electricq","Houseq","Urateq","Wageq"))
names(GDP_Q)
```

```
## [1] "Gdpq"      "Popq"      "Exportsq"  "Electricq" "Houseq"    "Urateq"
## [7] "Wageq"
```

```
attach(GDP_Q)

# Use cut function to change numerical variable "Employment rate" to catagoriacal variable
# Divide Employment rate to two interval[3.50,5.50),[5.50,9.69), which indicates "good" and "bad".
# Set up the train and testing data set.
set.seed(1)
UnemQ=cut(GDP_Q$Urateq,br=c(3.50,5.5,9.69),labels=c("good","bad"))
trainQ=sample(1:nrow(GDP_Q), nrow(GDP_Q)/2)
Gdpq.train=GDP_Q[trainQ,]
```

```r
Gdpq.test=GDP_Q[-trainQ,]
unem.train=UnemQ[trainQ]
unem.test=UnemQ[-trainQ]
contrasts(UnemQ)
```

```
##      bad
## good   0
## bad    1
```

```r
# Logistic Regression
glm.fit=glm(UnemQ~Popq+Exportsq+Houseq+Electricq+Gdpq+Wageq,data=GDP_Q,family = binomial,subset = train(
glm.probs=predict(glm.fit,Gdpq.test,type="response")
glm.pred=rep("good",28)
glm.pred[glm.probs>0.5]="bad"
log.fit=mean(glm.pred==unem.test)
log.fit
```

```
## [1] 0.6785714
```

```r
# Lda
library(MASS)
library(ISLR)

lda.fit=lda(UnemQ~Popq+Exportsq+Houseq+Electricq+Gdpq+Wageq,data=GDP_Q ,subset =trainQ)
lda.fit
```

```
## Call:
## lda(UnemQ ~ Popq + Exportsq + Houseq + Electricq + Gdpq + Wageq,
##     data = GDP_Q, subset = trainQ)
##
## Prior probabilities of groups:
##      good       bad
## 0.3571429 0.6428571
##
## Group means:
##           Popq     Exportsq       Houseq Electricq         Gdpq    Wageq
## good 247157442 742081000000 4.576654e+12  9737.034 6.894359e+12 22006.08
## bad  256597137 874726850000 5.184828e+12 10685.089 7.764617e+12 24794.03
##
## Coefficients of linear discriminants:
##                     LD1
## Popq      -3.181854e-07
## Exportsq   7.163671e-12
## Houseq     3.648947e-11
## Electricq  1.851220e-03
## Gdpq      -3.025892e-11
## Wageq      2.292981e-03
```

```r
lda.pred=predict(lda.fit,Gdpq.test)
lda.class=lda.pred$class
Lda.fit=mean(lda.class==unem.test)
Lda.fit
```

```
## [1] 0.6428571
```

```r
# QDA
qda.fit=qda(UnemQ~ Popq+Exportsq+Houseq+Electricq+Gdpq+Wageq,data=GDP_Q,subset=trainQ)
```

```r
qda.fit
```

```
## Call:
## qda(UnemQ ~ Popq + Exportsq + Houseq + Electricq + Gdpq + Wageq,
##     data = GDP_Q, subset = trainQ)
##
## Prior probabilities of groups:
##      good       bad
## 0.3571429 0.6428571
##
## Group means:
##           Popq     Exportsq       Houseq Electricq         Gdpq    Wageq
## good 247157442 742081000000 4.576654e+12  9737.034 6.894359e+12 22006.08
## bad  256597137 874726850000 5.184828e+12 10685.089 7.764617e+12 24794.03
```

```r
qda.class=predict(qda.fit,Gdpq.test)$class
table(qda.class,unem.test)
```

```
##          unem.test
## qda.class good bad
##      good    5   3
##      bad     4  16
```

```r
Qda.fit=mean(qda.class==unem.test)
Qda.fit
```

```
## [1] 0.75
```

```r
# Knn cross-validation(K=1,2,3)
library(class)
train01=cbind(Popq,Exportsq,Houseq,Electricq,Gdpq,Wageq)[trainQ,]
test01=cbind(Popq,Exportsq,Houseq,Electricq,Gdpq,Wageq)[-trainQ,]

knn.pred=knn(train01,test01,unem.train,k=1)
table(knn.pred,unem.test)
```

```
##         unem.test
## knn.pred good bad
##     good    6   0
##     bad     3  19
```

```r
K1.fit=mean(knn.pred==unem.test)
K1.fit
```

```
## [1] 0.8928571
```

```r
knn.pred=knn(train01,test01,unem.train,k=2)
table(knn.pred,unem.test)
```

```
##         unem.test
## knn.pred good bad
##     good    6   1
##     bad     3  18
```

```r
K2.fit=mean(knn.pred==unem.test)
K2.fit
```

```
## [1] 0.8571429
```

```
knn.pred=knn(train01,test01,unem.train,k=3)
table(knn.pred,unem.test)
```

```
##          unem.test
## knn.pred good bad
##     good    7   1
##     bad     2  18
```

```
K3.fit=mean(knn.pred==unem.test)
K3.fit
```

```
## [1] 0.8928571
```

```
data.frame(Model=c("Log.fit","Lda.fit","Qda.fit","K1.fit","K2.fit","K3.fit"),MSE=c(log.fit,Lda.fit,Qda.
```

```
##     Model       MSE
## 1 Log.fit 0.6785714
## 2 Lda.fit 0.6428571
## 3 Qda.fit 0.7500000
## 4  K1.fit 0.8928571
## 5  K2.fit 0.8571429
## 6  K3.fit 0.8928571
```

In thsi question, I picked "Unemployment rate" as dependent variable, and other variable as my predictors.I have used Logistic regression, LDA,QDA,KNN for predicting Empolyment rate. Compare the MSE we got above, KNN with K=1 and k=3 are the best model for the given data. #Question 4:

(Based on ISLR Chapter 9 #7) 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.

(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.

(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.

The best parameter of cost of is 1, which performs the best.

(c)

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

For a polynomia and radiall kernel, the lowest cross-validation error is obtained for a degree of 2 and a cost of 100.

(d)

Make some plots to back up your assertions in (b) and (c). Hint: In the lab, we used the plot() function for svm objects only in cases with p=2 When p>2,you can use the plot() function to create plots displaying pairs of variables at a time. Essentially, instead of typing plot(svmfit , dat) where svmfit contains your fitted model and dat is a data frame containing your data, you can type plot(svmfit , dat, x1???x4) in order to plot just the first and fourth variables. However, you must replace x1 and x4 with the correct variable names. To find out more, type ?plot.svm.

```
#(a)
library(ISLR)
bi.var=ifelse(Auto$mpg>median(Auto$mpg),1,0)
Auto$mpglevel=as.factor(bi.var)

#(b)
set.seed(1)
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.4.2
```

```
tune.out=tune(svm, mpglevel ~ ., data = Auto, kernel = "linear", ranges = list(cost = c(0.01, 0.1, 1, 5
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.01275641
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-02 0.07403846 0.05471525
## 2 1e-01 0.03826923 0.05148114
## 3 1e+00 0.01275641 0.01344780
## 4 5e+00 0.01782051 0.01229997
## 5 1e+01 0.02038462 0.01074682
## 6 1e+02 0.03820513 0.01773427
## 7 1e+03 0.03820513 0.01773427
```

```
#(c)
set.seed(1)
tune.poly=tune(svm, mpglevel ~ ., data = Auto, kernel = "polynomial", ranges = list(cost = c(0.01, 0.1,
summary(tune.poly)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
```

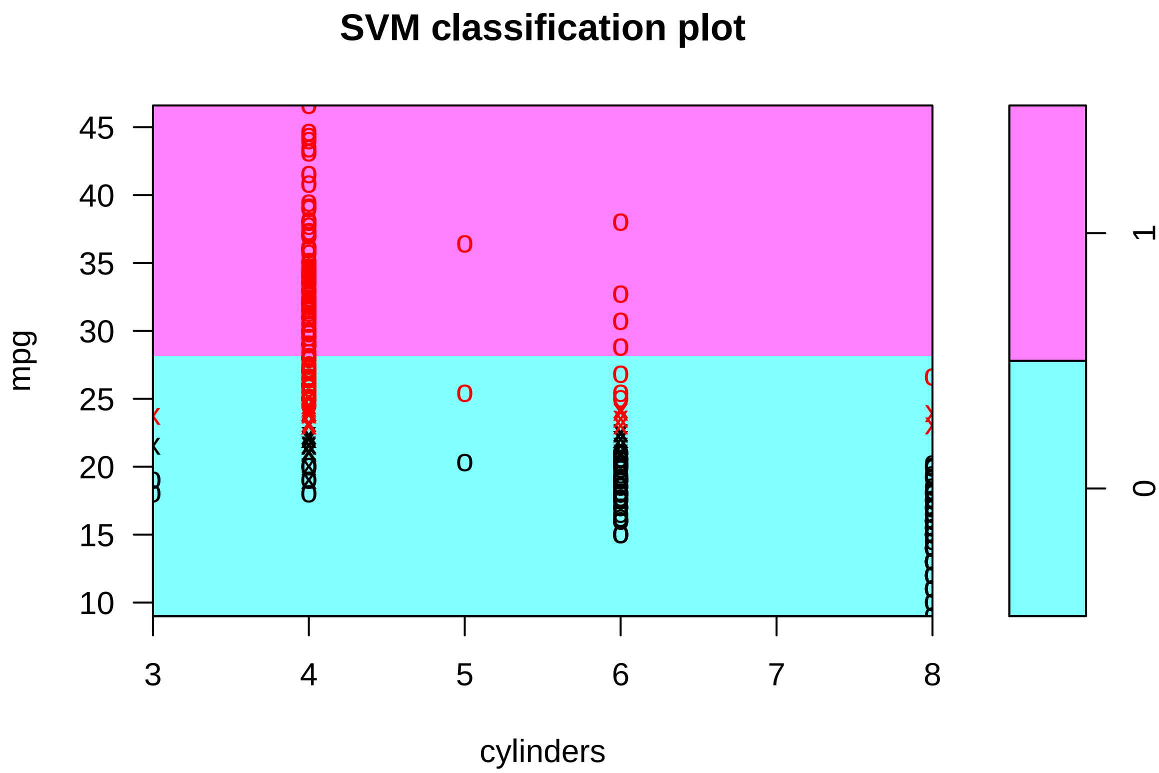
```
##   100      2
##
## - best performance: 0.3013462
##
## - Detailed performance results:
##     cost degree     error dispersion
## 1  1e-02      2 0.5611538 0.04344202
## 2  1e-01      2 0.5611538 0.04344202
## 3  1e+00      2 0.5611538 0.04344202
## 4  5e+00      2 0.5611538 0.04344202
## 5  1e+01      2 0.5382051 0.05829238
## 6  1e+02      2 0.3013462 0.09040277
## 7  1e-02      3 0.5611538 0.04344202
## 8  1e-01      3 0.5611538 0.04344202
## 9  1e+00      3 0.5611538 0.04344202
## 10 5e+00      3 0.5611538 0.04344202
## 11 1e+01      3 0.5611538 0.04344202
## 12 1e+02      3 0.3322436 0.11140578
## 13 1e-02      4 0.5611538 0.04344202
## 14 1e-01      4 0.5611538 0.04344202
## 15 1e+00      4 0.5611538 0.04344202
## 16 5e+00      4 0.5611538 0.04344202
## 17 1e+01      4 0.5611538 0.04344202
## 18 1e+02      4 0.5611538 0.04344202
## 19 1e-02      5 0.5611538 0.04344202
## 20 1e-01      5 0.5611538 0.04344202
## 21 1e+00      5 0.5611538 0.04344202
## 22 5e+00      5 0.5611538 0.04344202
## 23 1e+01      5 0.5611538 0.04344202
## 24 1e+02      5 0.5611538 0.04344202
```

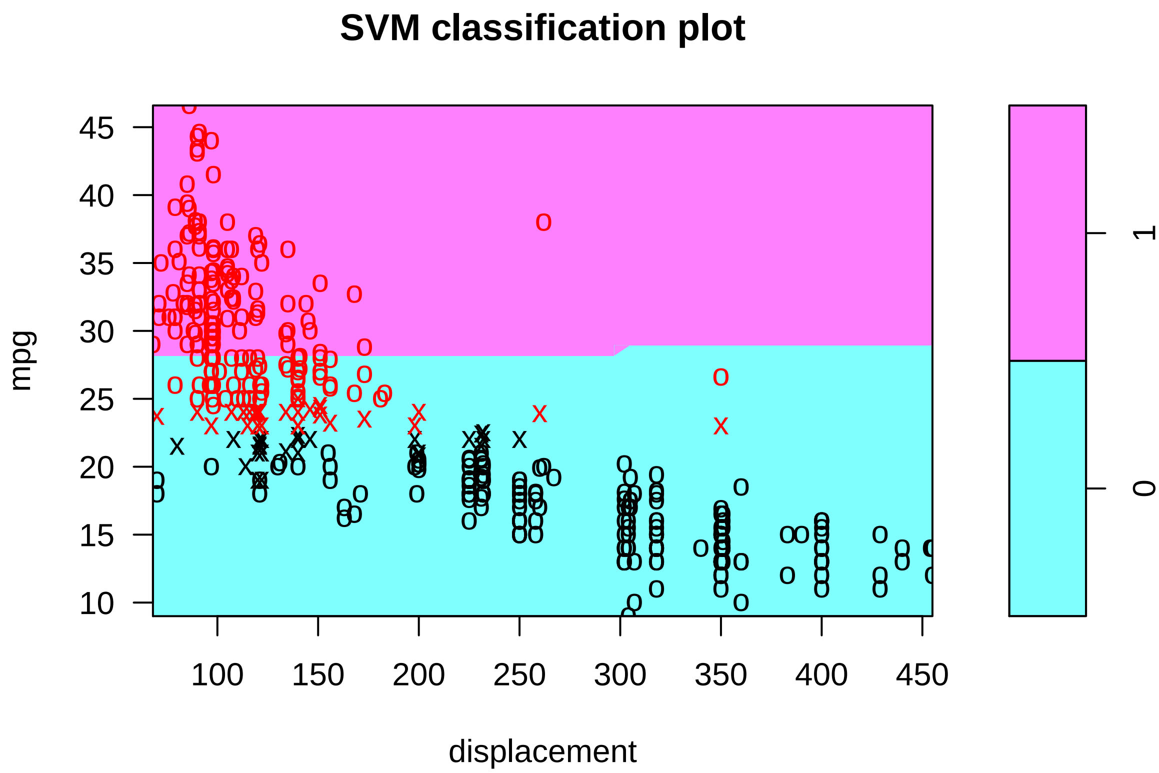
```r
tune.rad= tune(svm, mpglevel ~ ., data = Auto, kernel = "radial", ranges = list(cost = c(0.01, 0.1, 1, 5
summary(tune.rad)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     1   0.5
##
## - best performance: 0.04596154
##
## - Detailed performance results:
##     cost gamma      error dispersion
## 1  1e-02   0.5 0.55871795 0.05068311
## 2  1e-01   0.5 0.07916667 0.04084188
## 3  1e+00   0.5 0.04596154 0.02026662
## 4  5e+00   0.5 0.04839744 0.01859159
## 5  1e+01   0.5 0.04839744 0.01859159
## 6  1e+02   0.5 0.04839744 0.01859159
## 7  1e-02   1.0 0.55871795 0.05068311
## 8  1e-01   1.0 0.55871795 0.05068311
```

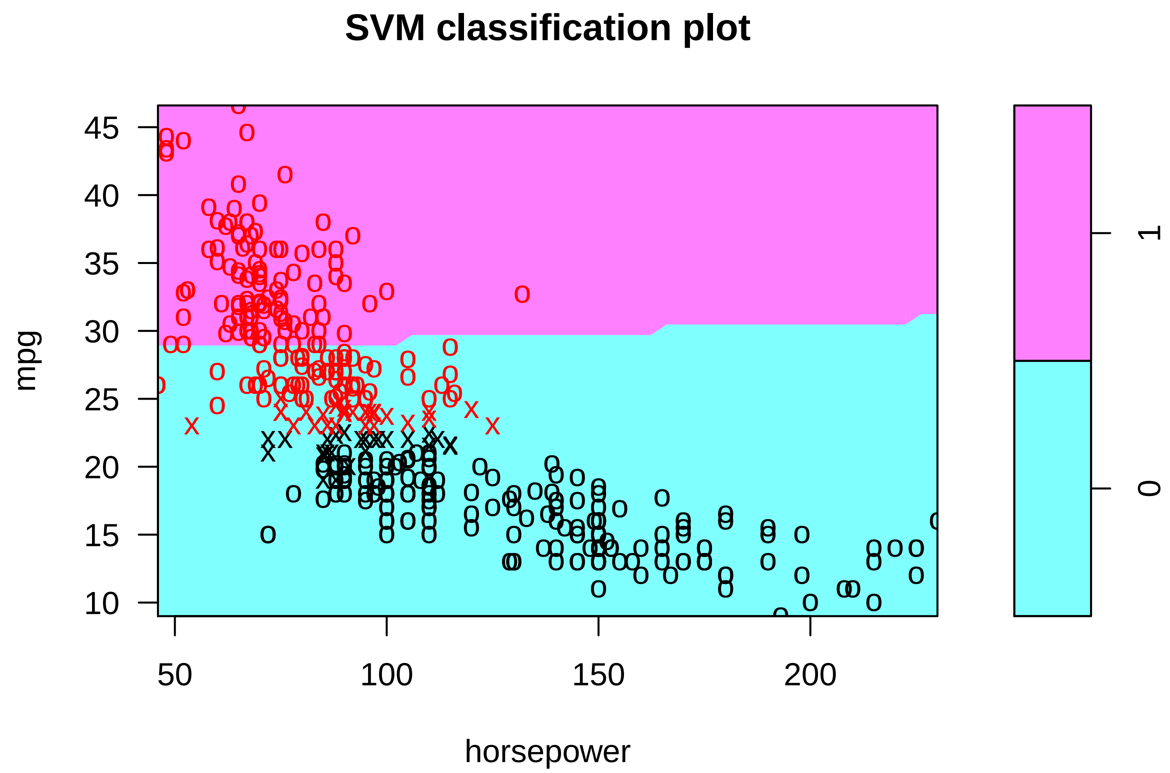
```
## 9  1e+00   1.0 0.06634615 0.03842264
## 10 5e+00   1.0 0.06628205 0.03999233
## 11 1e+01   1.0 0.06628205 0.03999233
## 12 1e+02   1.0 0.06628205 0.03999233
## 13 1e-02   2.0 0.55871795 0.05068311
## 14 1e-01   2.0 0.55871795 0.05068311
## 15 1e+00   2.0 0.11493590 0.09230772
## 16 5e+00   2.0 0.11493590 0.09230772
## 17 1e+01   2.0 0.11493590 0.09230772
## 18 1e+02   2.0 0.11493590 0.09230772
## 19 1e-02   3.0 0.55871795 0.05068311
## 20 1e-01   3.0 0.55871795 0.05068311
## 21 1e+00   3.0 0.37435897 0.18793801
## 22 5e+00   3.0 0.35403846 0.18582768
## 23 1e+01   3.0 0.35403846 0.18582768
## 24 1e+02   3.0 0.35403846 0.18582768
## 25 1e-02   4.0 0.55871795 0.05068311
## 26 1e-01   4.0 0.55871795 0.05068311
## 27 1e+00   4.0 0.48205128 0.08140100
## 28 5e+00   4.0 0.47948718 0.07702036
## 29 1e+01   4.0 0.47948718 0.07702036
## 30 1e+02   4.0 0.47948718 0.07702036
```

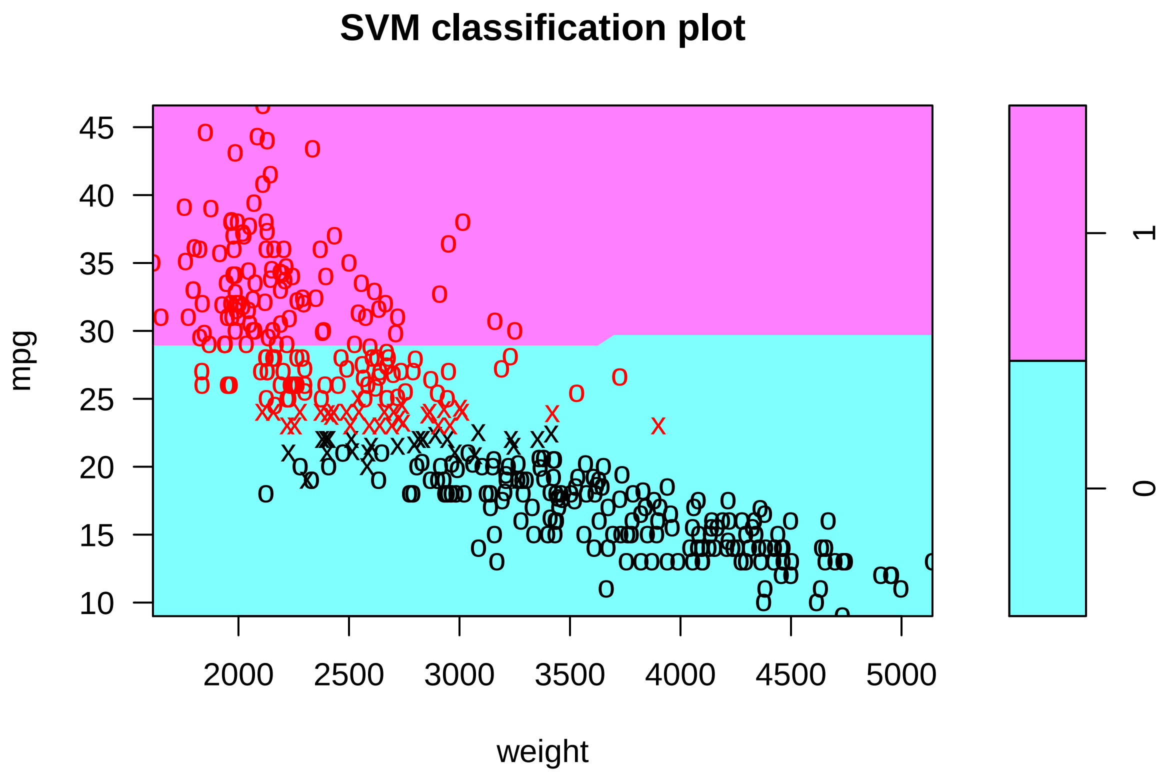
```r
#(d)
svm.fit= svm(mpglevel ~ ., data = Auto, kernel = "linear", cost = 1)
svm.poly=svm(mpglevel ~ ., data = Auto, kernel = "polynomial", cost = 100, degree = 2)
svm.radial=svm(mpglevel ~ ., data = Auto, kernel = "radial", cost = 100, gamma = 0.01)
plotpairs = function(fit) {
    for (name in names(Auto)[!(names(Auto) %in% c("mpg", "mpglevel", "name"))]) {
        plot(fit, Auto, as.formula(paste("mpg~", name, sep = "")))
    }
}
plotpairs(svm.fit)
```

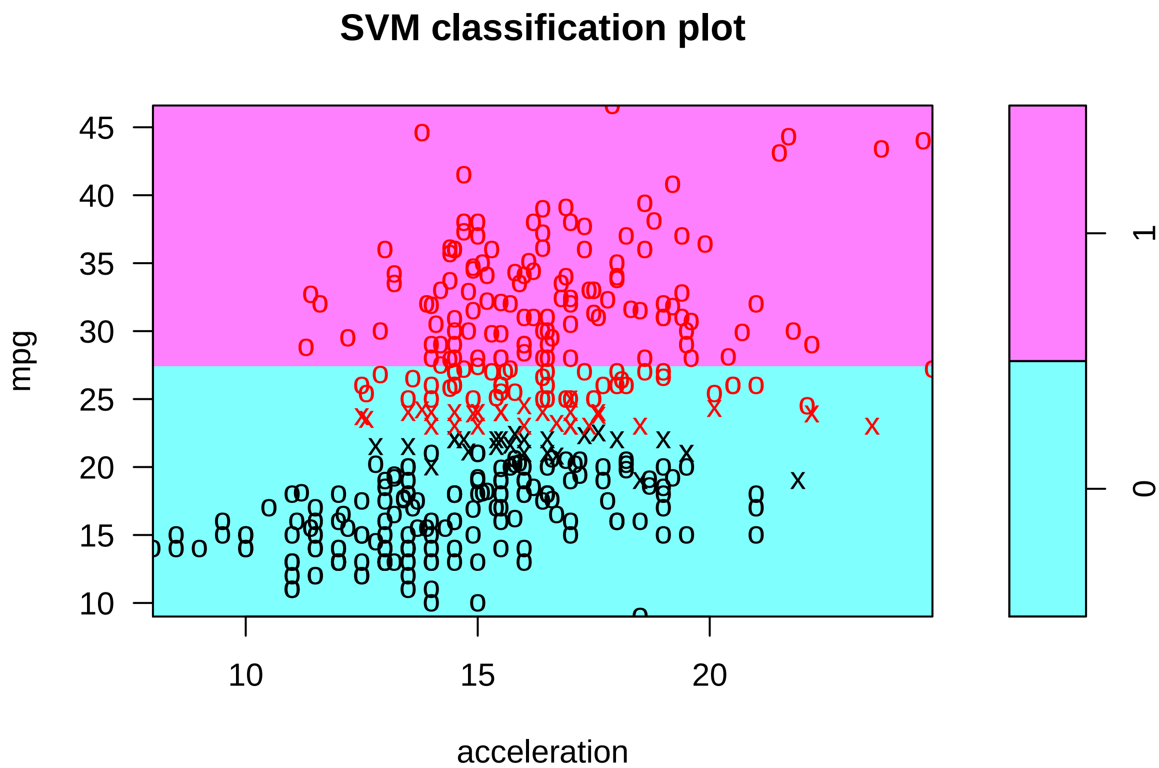

SVM classification plot
mpg
45
40
35
30
25
20
15
10
3
4
5
6
7
8
cylinders
1
0



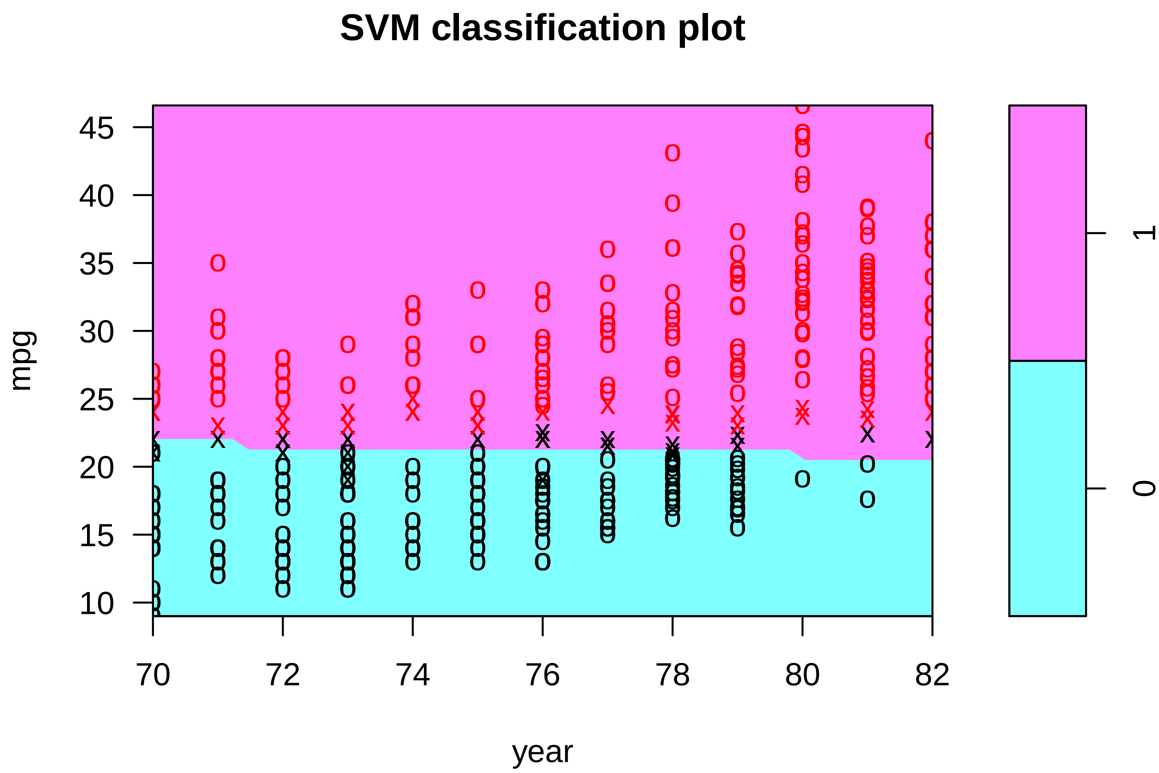
SVM classification plot
mpg
displacement
45
40
35
30
25
20
15
10
100
150
200
250
300
350
400
450
1
0

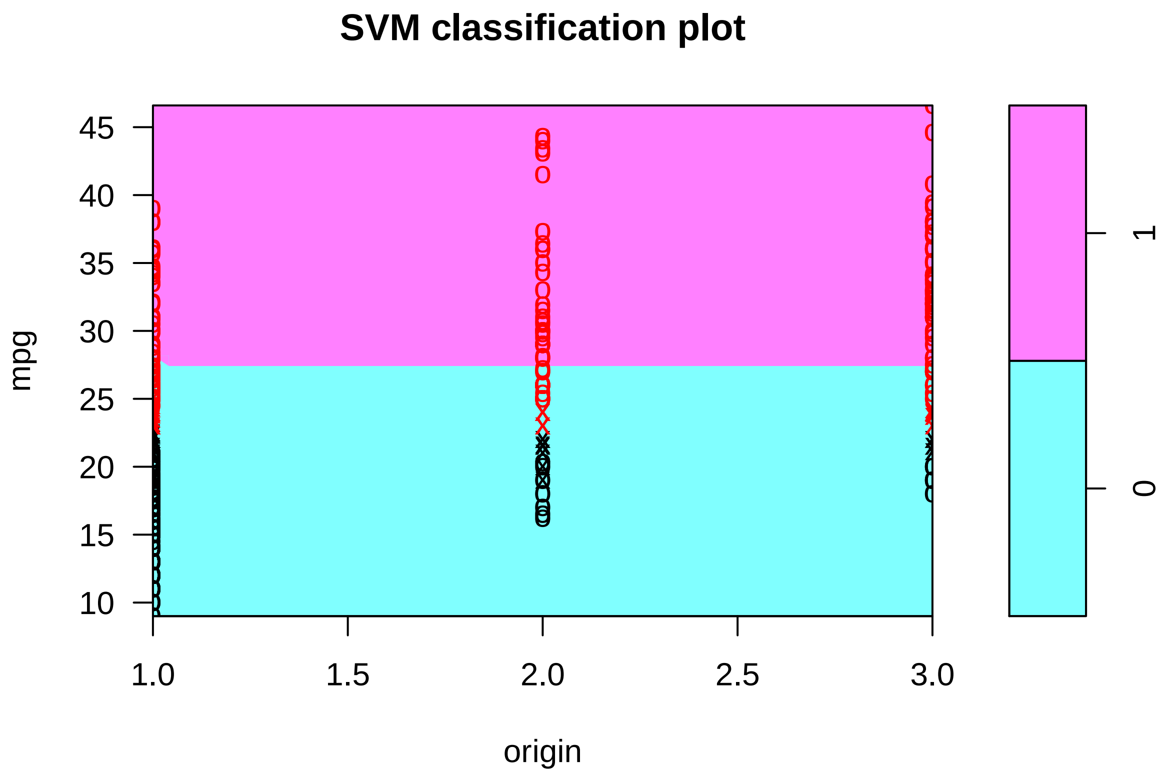

SVM classification plot
mpg
45
40
35
30
25
20
15
10
50
100
150
200
horsepower
1
0


SVM classification plot
mpg
45
40
35
30
25
20
15
10
2000
2500
3000
3500
4000
4500
5000
weight
1
0



SVM classification plot
45
40
35
30
25
20
15
10
mpg
10
15
20
acceleration
1
0


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



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