Homework 3

Gabriel Fei

11/18/2021

Predicting carseats sales using regularized regression methods

```
set.seed(123)

dat <- model.matrix(Sales~., Carseats)
train =  sample(nrow(dat), 30)
x.train = dat[train, ]
y.train = Carseats[train, ]$Sales

# The rest as test data
x.test = dat[-train, ]
y.test = Carseats[-train, ]$Sales
```

a)

```
lambda.list.ridge = 1000 * exp(seq(0, log(1e-5), length = 100))
ridge_mod = glmnet(x.train, y.train, alpha = 0, lambda = lambda.list.ridge)
cv.out.ridge = cv.glmnet(x.train, y.train, alpha = 0, folds = 5)
bestlam = cv.out.ridge$lambda.min
print(bestlam)
```

```
## [1] 0.1265465
```

```
coef(ridge_mod, bestlam)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                            1
## (Intercept)      6.5719453188
## (Intercept)      .
## CompPrice        0.0938756226
## Income           0.0075664204
## Advertising      0.0701179483
## Population      -0.0001226877
## Price           -0.0845397465
## ShelveLocGood    4.0442354708
## ShelveLocMedium  1.6201142294
## Age             -0.0527445709
## Education       -0.0939115995
## UrbanYes         0.5533244072
## USYes           -0.0173954221
```

The optimal value of the tuning paramter λ selected from the list of λ values using a 5-fold CV is 0.1265465. The ridge coefficient estimates corresponding to this value are shown above.

b)

```
ridge.pred.train = predict(ridge_mod, s = bestlam, newx = x.train)
ridge.pred.test = predict(ridge_mod, s = bestlam, newx = x.test)
print(mean((ridge.pred.train - y.train)^2))
```

```
## [1] 0.6190248
```

```
print(mean((ridge.pred.test - y.test)^2))
```

```
## [1] 1.460145
```

The training MSE for the model corresponding to the optimal value of λ selected by the CV is 0.6190248. The test MSE for the same model is 1.460145. Here we can see that when picking the optimal value of the tuning parameter, which is rather small, we get a small MSE, and this value is also optimal in the sense that not only do we get a small training MSE but we also get a small test MSE.

c)

```
lambda.list.lasso = 2 * exp(seq(0, log(1e-4), length = 100))
lasso_mod = glmnet(x.train, y.train, alpha = 1, lambda = lambda.list.lasso)
cv.out.lasso = cv.glmnet(x.train, y.train, alpha = 1, folds = 10)
bestlam2 = cv.out.lasso$lambda.min
print(bestlam2)
```

```
## [1] 0.006912323
```

```
coef(lasso_mod, bestlam2)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                             1
## (Intercept)      5.2066395170
## (Intercept)      .
## CompPrice        0.1170782474
## Income           0.0100554552
## Advertising      0.0639999131
## Population       0.0006586805
## Price           -0.1016845084
## ShelveLocGood    4.5198050021
## ShelveLocMedium  1.9399183644
## Age             -0.0651681218
## Education       -0.0806808234
## UrbanYes         0.6846286322
## USYes            0.4031090741
```

The optimal value of the tuning paramter λ selected from the list of λ values using a 10-fold CV is 0.006912323. The lasso coefficient estimates corresponding to this value are shown above. There are no coefficients set to zero in this model.

d)

```
lasso.pred.train = predict(lasso_mod, s = bestlam2, newx = x.train)
lasso.pred.test = predict(lasso_mod, s = bestlam2, newx = x.test)
print(mean((lasso.pred.train - y.train)^2))
```

```
## [1] 0.5004762
```

```r
print(mean((lasso.pred.test - y.test)^2))
```

```
## [1] 1.506844
```

The training MSE for the model corresponding to the optimal value of λ selected by the CV is 0.5004762. The test MSE for the same model is 1.506844. Here we can see that when picking the optimal value of the tuning parameter, which is rather small, we get a small MSE, and this value is also optimal in the sense that not only do we get a small training MSE but we also get a small test MSE.

e) The majority of the lasso estimates are larger than the ridge estimates by a small margin, meaning that to some extent ridge regression has some advantage over the lasso, in the sense that the parameters of the ridge regression don't hold as much influence in the model. However the difference in the estimates is so insubtantial that I believe both models give similar results. We can also see this from comparing the MSEs of the models as they are very similar, differing by a small margin in both training and test MSE.

Analyzing Drug Use

```r
drug <- read_csv('drug.csv',
col_names=c('ID','Age','Gender','Education','Country',
            'Ethnicity','Nscore',
            'Escore','Oscore','Ascore','Cscore',
            'Impulsive','SS','Alcohol','Amphet','Amyl','Benzos',
            'Caff','Cannabis', 'Choc','Coke','Crack','Ecstasy',
            'Heroin','Ketamine','Legalh','LSD','Meth',
            'Mushrooms','Nicotine','Semer','VSA'))
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   ID = col_double(),
##   Age = col_double(),
##   Gender = col_double(),
##   Education = col_double(),
##   Country = col_double(),
##   Ethnicity = col_double(),
##   Nscore = col_double(),
##   Escore = col_double(),
##   Oscore = col_double(),
##   Ascore = col_double(),
##   Cscore = col_double(),
##   Impulsive = col_double(),
##   SS = col_double()
## )
```

```
## See spec(...) for full column specifications.
```

a)

```r
drug = drug %>%
  mutate(recent_cannabis_use = factor(ifelse(Cannabis >= "CL3", "Yes", "No"), levels = c("No", "Yes")))
drug
```

```
## # A tibble: 1,885 x 33
##       ID     Age Gender Education Country Ethnicity Nscore   Escore  Oscore
##    <dbl>   <dbl>  <dbl>     <dbl>   <dbl>     <dbl>  <dbl>    <dbl>   <dbl>
##  1     1  0.498   0.482   -0.0592   0.961     0.126  0.313 -0.575   -0.583
##  2     2 -0.0785 -0.482    1.98     0.961    -0.317 -0.678  1.94     1.44
##  3     3  0.498  -0.482   -0.0592   0.961    -0.317 -0.467  0.805   -0.847
##  4     4 -0.952   0.482    1.16     0.961    -0.317 -0.149 -0.806   -0.0193
##  5     5  0.498   0.482    1.98     0.961    -0.317  0.735 -1.63    -0.452
##  6     6  2.59    0.482   -1.23     0.249    -0.317 -0.678 -0.300   -1.56
##  7     7  1.09   -0.482    1.16    -0.570    -0.317 -0.467 -1.09    -0.452
##  8     8  0.498  -0.482   -1.74     0.961    -0.317 -1.33   1.94    -0.847
##  9     9  0.498   0.482   -0.0592   0.249    -0.317  0.630  2.57    -0.976
## 10    10  1.82   -0.482    1.16     0.961    -0.317 -0.246  0.00332 -1.42
## # ... with 1,875 more rows, and 24 more variables: Ascore <dbl>, Cscore <dbl>,
## #   Impulsive <dbl>, SS <dbl>, Alcohol <chr>, Amphet <chr>, Amyl <chr>,
## #   Benzos <chr>, Caff <chr>, Cannabis <chr>, Choc <chr>, Coke <chr>,
## #   Crack <chr>, Ecstasy <chr>, Heroin <chr>, Ketamine <chr>, Legalh <chr>,
## #   LSD <chr>, Meth <chr>, Mushrooms <chr>, Nicotine <chr>, Semer <chr>,
## #   VSA <chr>, recent_cannabis_use <fct>
```

b)

```
drug_subset = drug %>%
  select(Age:SS, recent_cannabis_use)
drug_subset
```

```
## # A tibble: 1,885 x 13
##        Age Gender Education Country Ethnicity Nscore   Escore  Oscore Ascore
##      <dbl>  <dbl>     <dbl>   <dbl>     <dbl>  <dbl>    <dbl>   <dbl>  <dbl>
##  1  0.498   0.482   -0.0592   0.961     0.126  0.313 -0.575   -0.583  -0.917
##  2 -0.0785 -0.482    1.98     0.961    -0.317 -0.678  1.94     1.44    0.761
##  3  0.498  -0.482   -0.0592   0.961    -0.317 -0.467  0.805   -0.847  -1.62
##  4 -0.952   0.482    1.16     0.961    -0.317 -0.149 -0.806   -0.0193  0.590
##  5  0.498   0.482    1.98     0.961    -0.317  0.735 -1.63    -0.452  -0.302
##  6  2.59    0.482   -1.23     0.249    -0.317 -0.678 -0.300   -1.56    2.04
##  7  1.09   -0.482    1.16    -0.570    -0.317 -0.467 -1.09    -0.452  -0.302
##  8  0.498  -0.482   -1.74     0.961    -0.317 -1.33   1.94    -0.847  -0.302
##  9  0.498   0.482   -0.0592   0.249    -0.317  0.630  2.57    -0.976   0.761
## 10  1.82   -0.482    1.16     0.961    -0.317 -0.246  0.00332 -1.42    0.590
## # ... with 1,875 more rows, and 4 more variables: Cscore <dbl>,
## #   Impulsive <dbl>, SS <dbl>, recent_cannabis_use <fct>
```

c)

```
train2 = sample(nrow(drug_subset), 1100)
drug.train = drug_subset[train2, ]
drug.test = drug_subset[-train2, ]
dim(drug.train)
```

```
## [1] 1100   13
```

```r
dim(drug.test)
```

```
## [1] 785  13
```

d)

```r
drug.fit = glm(recent_cannabis_use~. , data = drug.train, family = binomial)
summary(drug.fit)
```

```
##
## Call:
## glm(formula = recent_cannabis_use ~ ., family = binomial, data = drug.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9064  -0.6138   0.1691   0.5673   2.5389
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.20658    0.24030   5.021 5.14e-07 ***
## Age         -0.82988    0.10806  -7.680 1.59e-14 ***
## Gender      -0.57332    0.18811  -3.048  0.00231 **
## Education   -0.41514    0.09435  -4.400 1.08e-05 ***
## Country     -1.12124    0.13694  -8.188 2.66e-16 ***
## Ethnicity    1.47419    0.68119   2.164  0.03045 *
## Nscore      -0.16738    0.10330  -1.620  0.10517
## Escore      -0.17053    0.11099  -1.536  0.12444
## Oscore       0.68940    0.10785   6.392 1.64e-10 ***
## Ascore       0.01965    0.09536   0.206  0.83672
## Cscore      -0.38334    0.10343  -3.706  0.00021 ***
## Impulsive   -0.15675    0.12157  -1.289  0.19725
## SS           0.67156    0.13056   5.144 2.69e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1519.10  on 1099  degrees of freedom
## Residual deviance:  884.16  on 1087  degrees of freedom
## AIC: 910.16
##
## Number of Fisher Scoring iterations: 5
```

e)

```r
drug.tree = tree(recent_cannabis_use~., data = drug.train)
summary(drug.tree)
```

```
##
## Classification tree:
## tree(formula = recent_cannabis_use ~ ., data = drug.train)
## Variables actually used in tree construction:
## [1] "Country"   "Age"       "Oscore"    "SS"        "Education" "Cscore"
## Number of terminal nodes:  9
## Residual mean deviance:  0.8696 = 948.8 / 1091
## Misclassification error rate: 0.1873 = 206 / 1100
```

f)

```r
set.seed(123)
best.tree = cv.tree(drug.tree, FUN = prune.misclass, K = 5)
best.tree
```

```
## $size
## [1] 9 7 5 4 2 1
##
## $dev
## [1] 238 238 256 253 277 510
##
## $k
## [1] -Inf    0    8   12   18  240
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

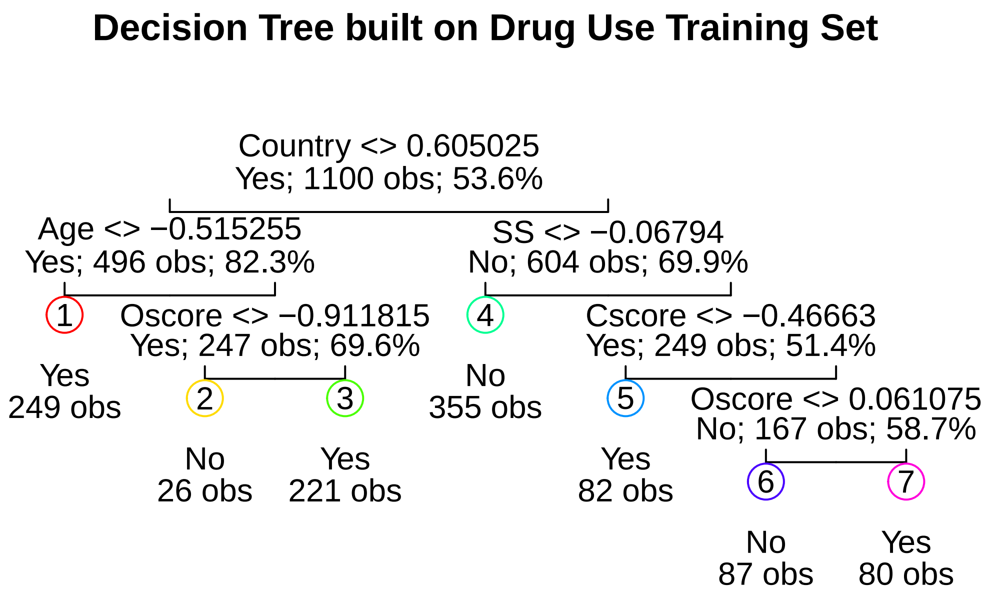
```r
# We have a between 9 and 7 but we perfer the tree with the smallest size
best_size = min(best.tree$size[best.tree$dev == min(best.tree$dev)])
best_size
```

```
## [1] 7
```

The best sized tree we obtained with the miniminum cv rate is a tree of size 7.

g)

```r
drug.prune.tree = prune.misclass(drug.tree, best = best_size)
draw.tree(drug.prune.tree, nodeinfo = TRUE)
title("Decision Tree built on Drug Use Training Set")
```



The variable Country is split first in this decision tree.

h)

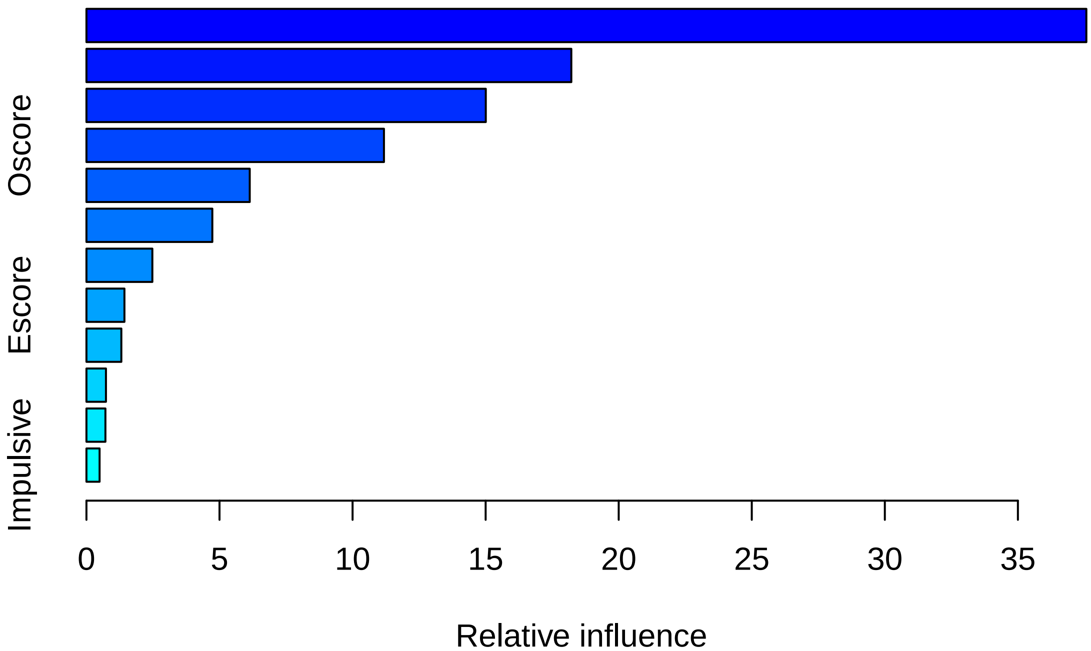
```
pred.drug.tree = predict(drug.prune.tree, drug.test, type = "class")
conf_matrix = table(true = drug.test$recent_cannabis_use, pred = pred.drug.tree)
conf_matrix
```

```
##       pred
## true   No Yes
##   No  282  94
##   Yes  67 342
```

True positive rate (TPR) is $\frac{\text{TP}}{\text{TP + FN}}$ and False positive rate (FPR) is $\frac{\text{FP}}{\text{FP + TN}}$. So our TPR is $\frac{342}{342+67}$ which equals 0.8361858 and our FPR is $\frac{94}{94+282}$ which equals 0.25.

i)

```
set.seed(123)
drug.boost = gbm(ifelse(recent_cannabis_use == "Yes", 1, 0)~., data = drug.train, distribution = "gauss
summary(drug.boost)
```




```
##                 var    rel.inf
## Country     Country 37.5719164
## SS               SS 18.2188512
## Age             Age 15.0057984
## Oscore       Oscore 11.1796197
## Cscore       Cscore  6.1367103
## Education Education  4.7299449
## Gender       Gender  2.4765997
## Escore       Escore  1.4276477
## Ethnicity Ethnicity  1.3118482
## Nscore       Nscore  0.7336041
## Ascore       Ascore  0.7138204
## Impulsive Impulsive  0.4936390
```

Country appears to be the most important variable.

j)

```
set.seed(123)
drug.rf = randomForest(recent_cannabis_use~., data = drug.train, important = TRUE)
drug.rf
```

```
##
## Call:
##  randomForest(formula = recent_cannabis_use ~ ., data = drug.train,      important = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
```

```
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 18.91%
## Confusion matrix:
##      No Yes class.error
## No  395 115   0.2254902
## Yes  93 497   0.1576271
```

```r
importance(drug.rf)
```

```
##           MeanDecreaseGini
## Age              55.890458
## Gender           15.911407
## Education        37.528247
## Country          94.830059
## Ethnicity         5.666161
## Nscore           39.758323
## Escore           37.525869
## Oscore           67.431039
## Ascore           37.557718
## Cscore           50.123868
## Impulsive        31.846478
## SS               70.670110
```

\textbf{The out-of-bag estimate error is 0.1891 or 18.91%. 3 variables were randomly considered at each split in the trees. 500 trees were used to fit the data. The order of important variables for random forrest models and boosting models are similar however there are some differences such as between Age and Oscore and between Ascore and Education.}

k)

```r
set.seed(123)
prob.boost = predict(drug.boost,newdata = drug.test, type = "response")
```

```
## Using 1000 trees...
```

```r
yhat.boost = ifelse(prob.boost >= 0.2, "Yes", "No")
prob.rf = predict(drug.rf, newdata = drug.test, type = "prob")
yhat.tree = ifelse(prob.rf[, 2] >= 0.2, "Yes", "No")
boost_conf_matrix = table(true = drug.test$recent_cannabis_use, pred = yhat.boost)
boost_conf_matrix
```

```
##      pred
## true   No Yes
##   No  156 220
##   Yes  15 394
```

```r
rf_conf_matrix = table(true = drug.test$recent_cannabis_use, pred = yhat.tree)
rf_conf_matrix
```

```
##      pred
## true   No Yes
##   No  181 195
##   Yes  24 385
```

In the random tree model, $\frac{385}{580}$ or 0.6637931 of the people predicted to use cannabis recently did in fact use cannabis recently.