Homework 13 Solution

Mohsen Nabian

8/14/2015

Using one or two high-dimensional datasets of your choice, estimate a shrinkage and an SVM model and test them out-of-sample. A large number of high-dimensional datasets can be found here: http://archive.ics.uci.edu/ml/datasets.html (http://archive.ics.uci.edu/ml/datasets.html) . Be sure to choose those that make your life easier, rather than something that takes a lot of manipulation to get into shape. But feel free to use other data or a dataset you have already used, as long as they have at least 10 independent variables and a continuous dependent variable (for lasso/ridge) and/or a binary dependent variable (for SVM). You can also convert a continous dependent variable to a binary for the SVM stage, as we did in the lesson.

1. Use your dataset with a continuous dependent variable:

a. Divide your data into two equal-sized samples, the in-sample and the out-sample. Estimate the elastic net model using at least three levels of alpha (ie, three positions in between full lasso and full ridge; eg, alpha = 0, 0.5, and 1), using cv.glmnet to find the best lambda level for each run. (Remember that glmnet prefers that data be in a numeric matrix format rather than a data frame.)
b. Choose the run (and lambda) with the best results (lowest error), and then test that model out-of-sample using the out-sample data.
c. Compare your out-of-sample results to regular multiple regression: fit the regression model in-sample, predict yhat out-of-sample, and estimate the error. Which works better?
d. Which coefficients are different between the multiple regression and the elastic net model? What, if anything, does this tell you substantively about the effects of your independent variables on your dependent variable?

Answer: I used the UCI website and I chose Online News Popularity created 2015. The goal is to provide an estimation of how many times a post of news would be shared?

Data information:

Attribute Information: 0. url: URL of the article (non-predictive) 1. timedelta: Days between the article publication and the dataset acquisition (non-predictive) 2. n-tokens-title: Number of words in the title 3. n-tokens-content: Number of words in the content 4. n-unique-tokens: Rate of unique words in the content 5. n-non-stop-words: Rate of non-stop words in the content 6. n-non-stop-unique-tokens: Rate of unique non-stop words in the content 7. num-hrefs: Number of links 8. num-self-hrefs: Number of links to other articles published by Mashable 9. num-imgs: Number of images 10. num-videos: Number of videos 11. average-token-length: Average length of the words in the content 12. num-keywords: Number of keywords in the metadata 13. data-channel-is-lifestyle: Is data channel 'Lifestyle'? 14. data-channel-is-entertainment: Is data channel 'Entertainment'? 15. data-channel-is-bus: Is data channel 'Business'? 16. data-channel-is-socmed: Is data channel 'Social Media'? 17. data-channel-is-tech: Is data channel 'Tech'? 18. data-channel-is-world: Is data channel 'World'? 19. kw-min-min: Worst keyword (min. shares) 20. kw-max-min: Worst keyword (max. shares) 21. kw-avg-min: Worst keyword (avg. shares) 22. kw-min-max: Best keyword (min. shares) 23. kw-max-max: Best keyword (max. shares) 24. kw-avg-max: Best keyword (avg. shares) 25. kw-min-avg: Avg. keyword (min. shares) 26. kw-max-avg: Avg. keyword (max. shares) 27. kw-avg-avg: Avg. keyword (avg. shares) 28. self-reference-min-shares: Min. shares of referenced articles in Mashable 29. self-reference-max-shares: Max. shares of referenced articles in Mashable 30. self-reference-avg-sharess: Avg. shares of referenced articles in Mashable 31. weekday-is-monday: Was the article published on a Monday? 32. weekday-is-tuesday: Was the article published on a Tuesday? 33. weekday-is-wednesday: Was the article published on a Wednesday? 34. weekday-is-thursday: Was the article published on a Thursday? 35. weekday-is-friday: Was the article published on a Friday? 36. weekday-is-saturday: Was the article published on a Saturday? 37. weekday-is-sunday: Was the article published on a Sunday? 38. is-weekend: Was the article published on the weekend? 39. LDA-00: Closeness to LDA topic 0 40. LDA-01: Closeness to LDA topic 1 41. LDA-02: Closeness to LDA topic 2 42. LDA-03: Closeness to LDA topic 3 43. LDA-04: Closeness to LDA topic 4 44. global-subjectivity: Text subjectivity 45. global-sentiment-polarity: Text sentiment polarity 46. global-rate-positive-words: Rate of positive words in the content 47. global-rate-negative-words: Rate of negative words in the content 48. rate-positive-words: Rate of positive words among non-neutral tokens 49. rate-negative-words: Rate of negative words among non-neutral tokens 50. avg-positive-polarity: Avg. polarity of positive words 51. min-positive-polarity: Min. polarity of positive words 52. max-positive-polarity: Max. polarity of positive words 53. avg-negative-polarity: Avg. polarity of negative words 54. min-negative-polarity: Min. polarity of negative words 55. max-negative-polarity: Max. polarity of negative words 56. title-subjectivity: Title subjectivity 57. title-sentiment-polarity: Title polarity 58. abs-title-subjectivity: Absolute subjectivity level 59. abs-title-sentiment-polarity: Absolute polarity level 60. shares: Number of shares (target)

1. Preparing in-sample and out-sample data:

```
setwd("C:/Users/nabian.m/Desktop")

raw_data<-read.csv("OnlineNewsPopularity.csv",header=TRUE,sep=",",stringsAsFactors=FALS
E,strip.white = TRUE, na.strings = c("NA",""))
#na.omit(raw_data)
#head(raw_data)


dim(raw_data)
```

```
## [1] 16391    61
```

```
y<-raw_data[,61]
length(y)
```

```
## [1] 16391
```

```
data<-raw_data[,c(-1,-61)]
#data<-data[,c(-10:-60)]  #reducing indp. variables
dim(data)
```

```
## [1] 16391    59
```

```
for (i in 1:length(data[1,]))
{
  data[,i]<-as.numeric(data[,i])
  data[,i]<-scale(data[,i])[,1]
}
x<-as.matrix(data)
class(x)
```

```
## [1] "matrix"
```

```
dim(x)
```

```
## [1] 16391    59
```

```
set.seed(126)
train<-sample(1:nrow(x),nrow(x)/2)
test<-(-train)
trainx<-x[train,]
trainy<-y[train]
testx<-x[test,]
testy<-y[test]
```

Now lets do the analysis:

```
set.seed(126)
#install.packages("glmnet")
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.2.1
```

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

```
## Warning: package 'foreach' was built under R version 3.2.1
```

```
## Loaded glmnet 2.0-2
```

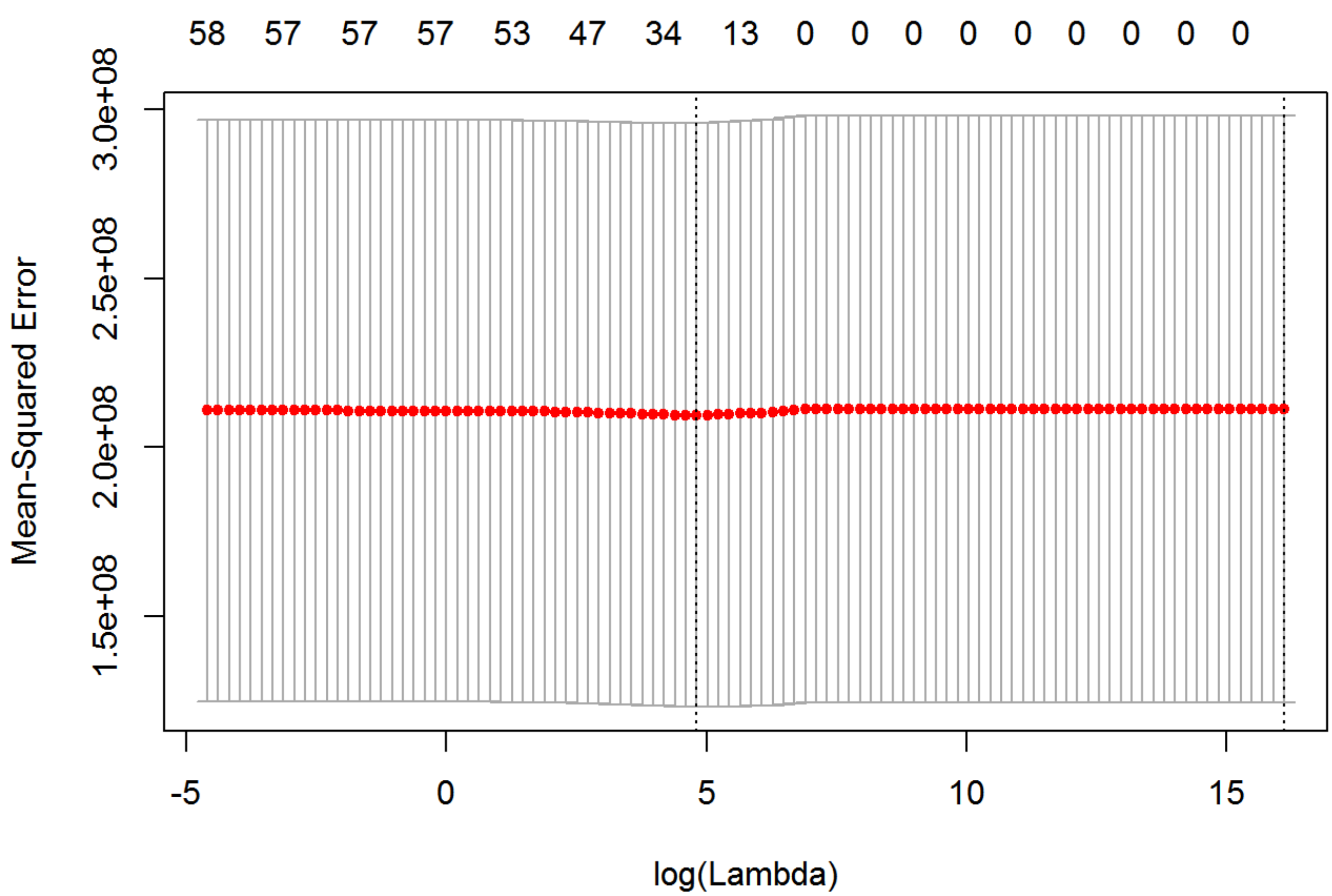
```
lambdalevels <- 10^seq(7,-2,length=100)

cv.lasso.mod_1=cv.glmnet(trainx,trainy,alpha=1,lambda=lambdalevels)
bestlambda <- cv.lasso.mod_1$lambda.min
lasso.mod=glmnet(x,y,alpha=1,lambda=lambdalevels)
predict(lasso.mod, type="coefficients",s=bestlambda)
```

```
## 60 x 1 sparse Matrix of class "dgCMatrix"
##                                        1
## (Intercept)                   3394.94650
## timedelta                         .
## n_tokens_title                    .
## n_tokens_content                  .
## n_unique_tokens                   .
## n_non_stop_words                  .
## n_non_stop_unique_tokens          .
## num_hrefs                      295.86820
## num_self_hrefs                 -72.94847
## num_imgs                        80.31435
## num_videos                      22.21505
## average_token_length              .
## num_keywords                      .
## data_channel_is_lifestyle         .
## data_channel_is_entertainment -223.02807
## data_channel_is_bus               .
## data_channel_is_socmed            .
## data_channel_is_tech              .
## data_channel_is_world         -104.19619
## kw_min_min                      23.20114
## kw_max_min                        .
## kw_avg_min                        .
## kw_min_max                        .
## kw_max_max                        .
## kw_avg_max                        .
## kw_min_avg                        .
## kw_max_avg                        .
## kw_avg_avg                     553.72687
## self_reference_min_shares      914.09358
## self_reference_max_shares       57.77307
## self_reference_avg_sharess        .
## weekday_is_monday                 .
## weekday_is_tuesday             -77.59942
## weekday_is_wednesday              .
## weekday_is_thursday               .
## weekday_is_friday                 .
## weekday_is_saturday            171.17091
## weekday_is_sunday                 .
## is_weekend                        .
## LDA_00                            .
```

```
## LDA_01                                  .
## LDA_02                          -76.31756
## LDA_03                          371.39729
## LDA_04                                  .
## global_subjectivity             178.33592
## global_sentiment_polarity               .
## global_rate_positive_words              .
## global_rate_negative_words              .
## rate_positive_words                     .
## rate_negative_words                     .
## avg_positive_polarity                   .
## min_positive_polarity           -46.33790
## max_positive_polarity                   .
## avg_negative_polarity          -139.55414
## min_negative_polarity          -109.28978
## max_negative_polarity                   .
## title_subjectivity                      .
## title_sentiment_polarity                .
## abs_title_subjectivity           61.58206
## abs_title_sentiment_polarity            .
```

```
plot(cv.lasso.mod_1)
```



```
cv.lasso.mod_half=cv.glmnet(trainx,trainy,alpha=0.5,lambda=lambdalevels)
bestlambda <- cv.lasso.mod_half$lambda.min
bestlambda
```

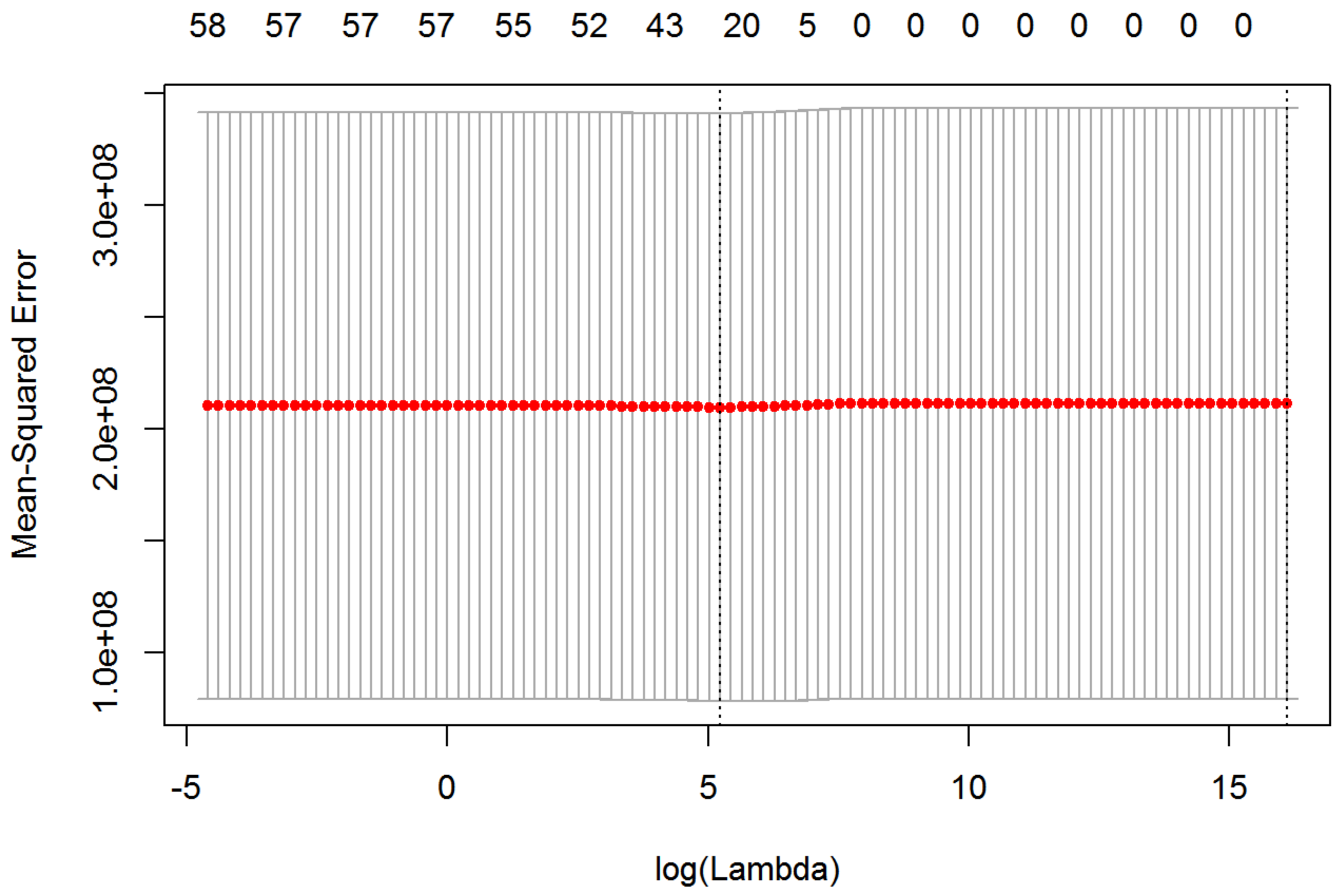
```
## [1] 187.3817
```

```
lasso.mod=glmnet(x,y,alpha=0.5,lambda=lambdalevels)
predict(lasso.mod, type="coefficients",s=bestlambda)
```

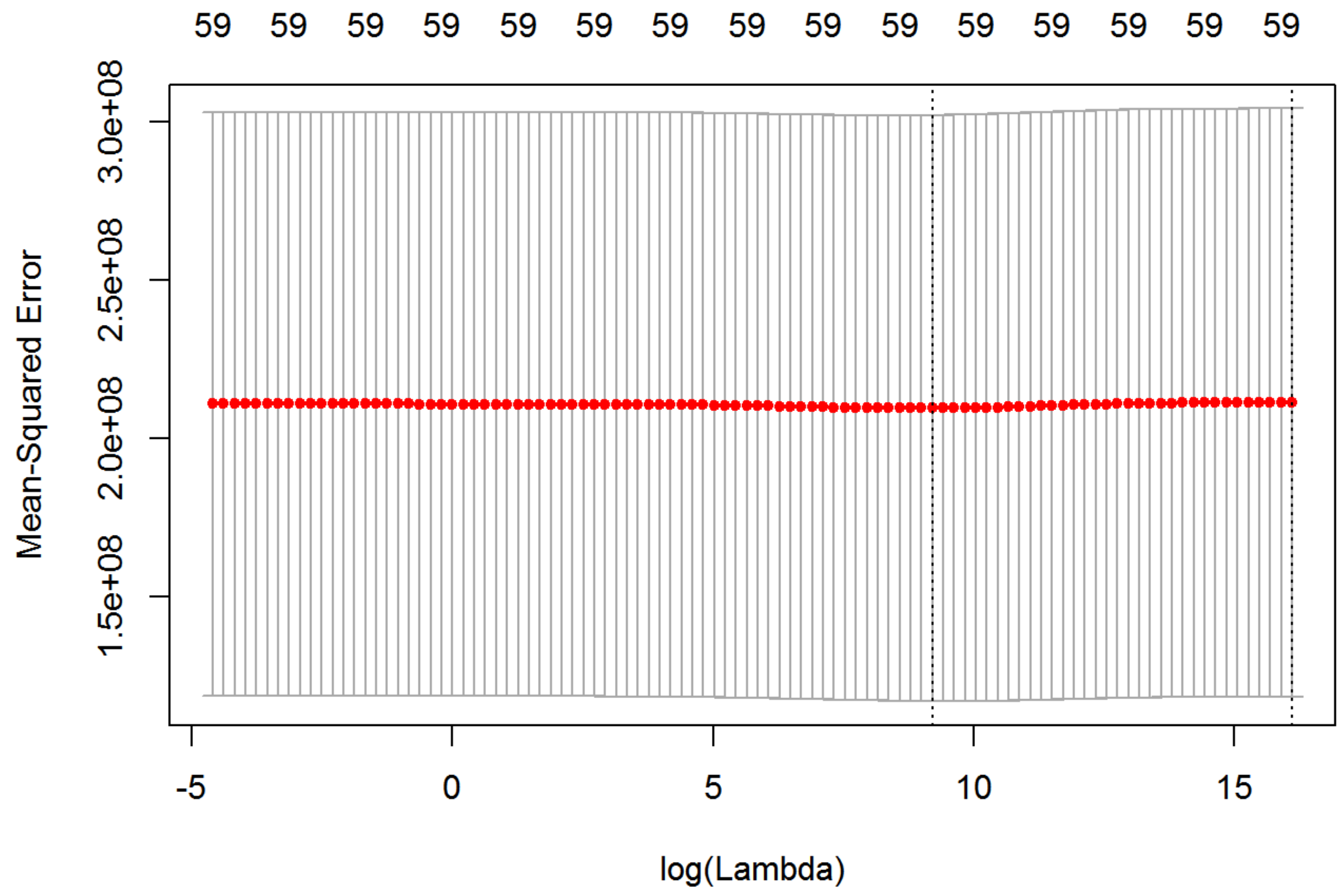
```
## 60 x 1 sparse Matrix of class "dgCMatrix"
##                                          1
## (Intercept)                   3394.9464950
## timedelta                        .        
## n_tokens_title                   5.9093761
## n_tokens_content                 7.4767742
## n_unique_tokens                  .        
## n_non_stop_words                -0.5237418
## n_non_stop_unique_tokens         .        
## num_hrefs                      328.1048380
## num_self_hrefs                -135.4516189
## num_imgs                       106.7656282
## num_videos                      45.6696122
## average_token_length             .        
## num_keywords                     .        
## data_channel_is_lifestyle        .        
## data_channel_is_entertainment -254.6126065
## data_channel_is_bus              .        
## data_channel_is_socmed           .        
## data_channel_is_tech             .        
## data_channel_is_world         -120.4683970
## kw_min_min                      77.1319316
## kw_max_min                       .        
## kw_avg_min                       .        
## kw_min_max                       .        
## kw_max_max                       .        
## kw_avg_max                      33.4514265
## kw_min_avg                       .        
## kw_max_avg                     -60.5314407
## kw_avg_avg                     606.8695005
## self_reference_min_shares      918.3625339
## self_reference_max_shares       86.8131523
## self_reference_avg_sharess       .        
## weekday_is_monday               30.5279024
## weekday_is_tuesday             -97.0591353
## weekday_is_wednesday             .        
## weekday_is_thursday              .        
## weekday_is_friday                .        
## weekday_is_saturday            200.4729433
```

```
## weekday_is_sunday                 .
## is_weekend                        .
## LDA_00                            .
## LDA_01                            .
## LDA_02                          -89.5921397
## LDA_03                          359.1542984
## LDA_04                            .
## global_subjectivity             192.0879637
## global_sentiment_polarity         .
## global_rate_positive_words        .
## global_rate_negative_words        .
## rate_positive_words               .
## rate_negative_words               .
## avg_positive_polarity             .
## min_positive_polarity           -80.7542887
## max_positive_polarity             .
## avg_negative_polarity          -158.6735046
## min_negative_polarity          -102.4031642
## max_negative_polarity             .
## title_subjectivity                .
## title_sentiment_polarity          .
## abs_title_subjectivity           93.7448541
## abs_title_sentiment_polarity      .
```

```
plot(cv.lasso.mod_half)
```




```
cv.lasso.mod_0=cv.glmnet(trainx,trainy,alpha=0,lambda=lambdalevels)
plot(cv.lasso.mod_0)
```



For alpha=0.5 with lambda=53.37 we end up to the lowest mean square error and highest number of variables that are shrunk. SO we choose alpha=0.5 at its best lambda.

```
yhat.l <- predict(cv.lasso.mod_half$glmnet.fit, s=cv.lasso.mod_half$lambda.min, newx=test
x)
# ^^ note how we give predict() our best lambda.min value to use for prediction

mse.las <- sum((testy - yhat.l)^2)/nrow(testx)
mse.las
```

```
## [1] 178931861
```

Now lets do the regular regression:

```
lmout <- lm(trainy~trainx)
lmout$coefficients[is.na(lmout$coefficients)] <- 0
summary(lmout)
```

```
##
## Call:
## lm(formula = trainy ~ trainx)
```

```
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -33539  -2445  -1177    225 832849
##
## Coefficients: (3 not defined because of singularities)
##                                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)                          3434.094    159.597  21.517  < 2e-16
## trainxtimedelta                      -104.592    310.204  -0.337 0.735995
## trainxn_tokens_title                  315.062    168.184   1.873 0.061060
## trainxn_tokens_content                524.761    299.666   1.751 0.079957
## trainxn_unique_tokens                   8.329    613.655   0.014 0.989171
## trainxn_non_stop_words               -277.723   1257.770  -0.221 0.825249
## trainxn_non_stop_unique_tokens          5.117    527.378   0.010 0.992259
## trainxnum_hrefs                       323.453    234.458   1.380 0.167754
## trainxnum_self_hrefs                 -625.254    188.013  -3.326 0.000886
## trainxnum_imgs                        191.416    203.653   0.940 0.347290
## trainxnum_videos                     -115.535    187.775  -0.615 0.538386
## trainxaverage_token_length            210.420    320.105   0.657 0.510977
## trainxnum_keywords                   -131.224    197.257  -0.665 0.505914
## trainxdata_channel_is_lifestyle      -606.443    277.234  -2.187 0.028736
## trainxdata_channel_is_entertainment -1116.286    256.419  -4.353 1.36e-05
## trainxdata_channel_is_bus           -1196.103    402.459  -2.972 0.002967
## trainxdata_channel_is_socmed         -613.891    263.963  -2.326 0.020060
## trainxdata_channel_is_tech          -1212.381    431.494  -2.810 0.004970
## trainxdata_channel_is_world         -1026.469    384.769  -2.668 0.007651
## trainxkw_min_min                      332.600    289.867   1.147 0.251241
## trainxkw_max_min                      670.284    543.517   1.233 0.217523
## trainxkw_avg_min                     -380.923    468.194  -0.814 0.415897
## trainxkw_min_max                      -85.929    182.404  -0.471 0.637587
## trainxkw_max_max                      -67.334    385.511  -0.175 0.861349
## trainxkw_avg_max                      -63.392    338.027  -0.188 0.851246
## trainxkw_min_avg                     -225.565    233.614  -0.966 0.334300
## trainxkw_max_avg                     -931.257    426.676  -2.183 0.029095
## trainxkw_avg_avg                     1157.163    500.482   2.312 0.020797
## trainxself_reference_min_shares      1040.880    408.900   2.546 0.010928
## trainxself_reference_max_shares      1869.091    449.748   4.156 3.27e-05
## trainxself_reference_avg_sharess    -1982.658    670.485  -2.957 0.003115
## trainxweekday_is_monday               100.907    277.035   0.364 0.715688
## trainxweekday_is_tuesday             -172.923    286.597  -0.603 0.546281
## trainxweekday_is_wednesday            247.536    288.592   0.858 0.391063
## trainxweekday_is_thursday             139.438    284.056   0.491 0.623522
## trainxweekday_is_friday               165.749    265.225   0.625 0.532030
## trainxweekday_is_saturday             444.879    216.167   2.058 0.039619
## trainxLDA_00                          -82.998    332.460  -0.250 0.802866
## trainxLDA_01                         -185.321    310.835  -0.596 0.551055
## trainxLDA_02                         -379.472    336.402  -1.128 0.259340
## trainxLDA_03                          -77.753    385.638  -0.202 0.840217
## trainxglobal_subjectivity             327.688    219.460   1.493 0.135435
## trainxglobal_sentiment_polarity       257.560    428.899   0.601 0.548179
```

```
## trainxglobal_rate_positive_words     -219.509    305.003  -0.720 0.471733
## trainxglobal_rate_negative_words     -244.486    410.682  -0.595 0.551648
## trainxrate_positive_words             431.919   2314.081   0.187 0.851941
## trainxrate_negative_words             539.870   2188.987   0.247 0.805201
## trainxavg_positive_polarity          -515.105    340.177  -1.514 0.130008
## trainxmin_positive_polarity          -248.794    224.671  -1.107 0.268168
## trainxmax_positive_polarity           145.346    264.630   0.549 0.582854
## trainxavg_negative_polarity            55.562    459.361   0.121 0.903730
## trainxmin_negative_polarity          -196.741    374.266  -0.526 0.599131
## trainxmax_negative_polarity            -2.907    289.450  -0.010 0.991988
## trainxtitle_subjectivity             -300.417    253.263  -1.186 0.235584
## trainxtitle_sentiment_polarity       -428.654    192.589  -2.226 0.026058
## trainxabs_title_subjectivity           72.860    193.337   0.377 0.706290
## trainxabs_title_sentiment_polarity    407.452    257.728   1.581 0.113932
##
## (Intercept)                          ***
## trainxtimedelta
## trainxn_tokens_title                 .
## trainxn_tokens_content               .
## trainxn_unique_tokens
## trainxn_non_stop_words
## trainxn_non_stop_unique_tokens
## trainxnum_hrefs
## trainxnum_self_hrefs                 ***
## trainxnum_imgs
## trainxnum_videos
## trainxaverage_token_length
## trainxnum_keywords
## trainxdata_channel_is_lifestyle      *
## trainxdata_channel_is_entertainment ***
## trainxdata_channel_is_bus            **
## trainxdata_channel_is_socmed         *
## trainxdata_channel_is_tech           **
## trainxdata_channel_is_world          **
## trainxkw_min_min
## trainxkw_max_min
## trainxkw_avg_min
## trainxkw_min_max
## trainxkw_max_max
## trainxkw_avg_max
## trainxkw_min_avg
## trainxkw_max_avg                     *
## trainxkw_avg_avg                     *
## trainxself_reference_min_shares      *
## trainxself_reference_max_shares      ***
## trainxself_reference_avg_sharess     **
## trainxweekday_is_monday
## trainxweekday_is_tuesday
## trainxweekday_is_wednesday
## trainxweekday_is_thursday
```

```
## trainxweekday_is_friday
## trainxweekday_is_saturday          *
## trainxLDA_00
## trainxLDA_01
## trainxLDA_02
## trainxLDA_03
## trainxglobal_subjectivity
## trainxglobal_sentiment_polarity
## trainxglobal_rate_positive_words
## trainxglobal_rate_negative_words
## trainxrate_positive_words
## trainxrate_negative_words
## trainxavg_positive_polarity
## trainxmin_positive_polarity
## trainxmax_positive_polarity
## trainxavg_negative_polarity
## trainxmin_negative_polarity
## trainxmax_negative_polarity
## trainxtitle_subjectivity
## trainxtitle_sentiment_polarity     *
## trainxabs_title_subjectivity
## trainxabs_title_sentiment_polarity
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14430 on 8138 degrees of freedom
## Multiple R-squared:  0.02056,    Adjusted R-squared:  0.01382
## F-statistic: 3.051 on 56 and 8138 DF,  p-value: 2.274e-13
```

```
yhat.r <- cbind(1,testx) %*% lmout$coefficients

head(yhat.r)
```

```
##             [,1]
## [1,] 1063.48468
## [2,]  130.27074
## [3,]  -62.84245
## [4,]  675.03615
## [5,] -278.60576
## [6,] -653.70780
```

```
mse.reg <- sum((testy - yhat.r)^2)/nrow(testx)
mse.reg
```

```
## [1] 179893236
```

with such a data set of 60 indep. variables, it is probable to have over fitting produced in the multiple regression method.

Comparing the error models of Elastic Net(alpha=0.5) and multiple regression, we can conclude the elastic net method has been more sucsessful in estimating the out-of-sample data.This could be due to the over-fitting effect that is caused by multiple regression.

2. Repeat the same process using your dataset with a binary dependent variable:

a. Divide your data into an in-sample and out-sample as before, and estimate an SVM using at least two different kernels and tune to find the best cost level for each.
b. Chose the kernel and cost with the best results, and then test that model out-of-sample using the out-sample data.
c. Compare your results to a logistic regression: fit the logit in-sample, predict yhat out-of-sample, and estimate the accuracy. Which works better?
d. Can you make any guesses as to why the SVM works better (if it does)? Feel to speculate, or to research a bit more the output of svm, the meaning of the support vectors, or anything else you can discover about SVMs (no points off for erroneous speculations!).

Answer:

we use the same dataset, but we assume one for shares more than 1000 and assume 0 for shares less than 1000.

```
y <- ifelse(y>1000,1,0)
dat <- data.frame(x=x, y=as.factor(y))
dat0<-cbind(dat[1:100,1:10],y=as.factor(y[1:100]))
class(dat0)
```

```
## [1] "data.frame"
```

```
dim(dat0)
```

```
## [1] 100  11
```

```
set.seed(126)
train<-sample(1:nrow(dat0),nrow(dat0)/2)
test<-(-train)
train_dt<-dat0[train,]
test_dt<-dat0[test,]
#install.packages("e1071")
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.2.1
```

```
costvalues <- 10^seq(-3,2,1)
tuned.svm <- tune(svm,y~., data=train_dt, ranges=list(cost=costvalues), kernel="linear")
summary(tuned.svm)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   100
##
## - best performance: 0.48
##
## - Detailed performance results:
##    cost error dispersion
## 1 1e-03  0.66 0.09660918
## 2 1e-02  0.66 0.09660918
## 3 1e-01  0.54 0.18973666
## 4 1e+00  0.54 0.21186998
## 5 1e+01  0.60 0.28284271
## 6 1e+02  0.48 0.21499354
```

So our svm is 62% accurate.(although we now it is over valued and the accuracy is less)

Now lets test the svm model with the out of sample data:

```
yhat <- predict(tuned.svm$best.model,newdata=test_dt)

table(predicted=yhat,truth=test_dt$y)
```

```
##          truth
## predicted  0  1
##         0 15 22
##         1  6  7
```

```
sum(yhat==test_dt$y)/length(test_dt$y)
```

```
## [1] 0.44
```

```
sum(yhat==rep(0,length(test_dt$y)))/length(test_dt$y)
```

```
## [1] 0.74
```

SO we had 44% correct out-of sample response. The result from svm might not be satisfactory since 44% is less than 50%. So if we even randomely choose the y, we would have better chance to be correct:(50>44) However the reason of this law value is beacause we had only 100 data in whole with 10 variables. we would expect much better performance with higher amount of data.

Lets compare to logistic regression:

```
reg <- glm(y~.,data=train_dt,family="binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(reg)
```

```
##
## Call:
## glm(formula = y ~ ., family = "binomial", data = train_dt)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8929  -0.9494  -0.2307   0.9642   1.8033
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)                1.091e+05  1.976e+06   0.055    0.956
## x.timedelta               -8.491e+01  5.497e+01  -1.545    0.122
## x.n_tokens_title           2.171e-01  4.082e-01   0.532    0.595
## x.n_tokens_content         2.461e+00  1.498e+00   1.643    0.100
## x.n_unique_tokens          2.432e+00  2.265e+00   1.074    0.283
## x.n_non_stop_words        -1.296e+06  2.351e+07  -0.055    0.956
## x.n_non_stop_unique_tokens -1.015e+00  1.625e+00  -0.625    0.532
## x.num_hrefs               -4.008e-01  1.059e+00  -0.378    0.705
## x.num_self_hrefs          -2.949e-01  4.658e-01  -0.633    0.527
## x.num_imgs                 2.865e-01  6.061e-01   0.473    0.636
## x.num_videos               9.292e+00  4.843e+00   1.919    0.055 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 69.315  on 49  degrees of freedom
## Residual deviance: 54.534  on 39  degrees of freedom
## AIC: 76.534
##
## Number of Fisher Scoring iterations: 7
```

```
reg$coefficients
```

```
##               (Intercept)                x.timedelta
##              1.091175e+05              -8.491268e+01
##            x.n_tokens_title          x.n_tokens_content
##              2.171343e-01               2.460836e+00
##           x.n_unique_tokens          x.n_non_stop_words
##              2.431805e+00              -1.296470e+06
## x.n_non_stop_unique_tokens                x.num_hrefs
##             -1.015185e+00              -4.008411e-01
##            x.num_self_hrefs                 x.num_imgs
##             -2.949387e-01               2.865173e-01
##              x.num_videos
##              9.292429e+00
```

```
yhat.r <- cbind(1,as.matrix(test_dt[,-11])) %*% reg$coefficients
yhat.r <- ifelse(yhat.r>0.5,1,0)
sum(yhat.r==test_dt$y)/length(test_dt$y)
```

```
## [1] 0.4
```

so as we see, by using logistic regression, we can estimate 40%. Still not acceptable, but comparing to svm it is less accurate for the out of sample data.

SVM could be more accuarate since it is a supervised learning while logistic regression is not supervised. ie svm has a one degree of freedom(cost value) to optimize its calculations based on the k fold data.

logistic regression might have over fitting which causes less accurate results. svm is more concentrated on finding a seprating curve, however, logistic reg. is focused more on the effect of individual variables.