

Advanced Data Mining and Predictive Analytics Assignment 1

Sai Gautham Sabhavathu

2022-10-26

Part A QA1. What is the main purpose of regularization when training predictive models? Answer:- Regularization means to make things acceptable. Regularization is a technique used to reduce the error by fitting a function appropriately on the training set and to avoid over fitting. It penalizes the model for being too complex and will shrink the coefficients towards zero. It limits the model's capabilities to learn with a given set of techniques.

QA2. What is the role of a loss function in a predictive model? And name two common Answer:- Loss functions for regression models and two common loss functions for classification models. Loss function is a measure of how good your prediction model does in terms of being able to predict the expected outcome (or value). The role of a loss function in a predictive model is to measure the penalty. It calculates the distance between the output of the algorithm and the expected output. The two common loss functions for regression models are Sum Square of Error (SSE) and Mean absolute error (MAE). The two common loss functions for classification models are binary-cross-entropy and negative log likelihood.

QA3. Consider the following scenario. You are building a classification model with many hyper parameters on a relatively small data set. You will see that the training error is extremely small. Can you fully trust this model? Discuss the reason. Answer:- When you are dealing with a small data set, the model will try to fit every data point which means that the training error will be low. This may lead to over fitting. Hence, the model cannot be trusted as the training error is extremely small.

QA4. What is the role of the lambda parameter in regularized linear models such as Lasso or Ridge regression models? Answer:- The lambda parameter controls the amount of regularization applied to the model. A non-negative value represents a shrinkage parameter, which multiplies in the objective. The lambda balances between minimizing the sum square of the residuals. Larger the lambda, the more the coefficients are shrunk toward zero (and each other).

Part B

```
# Loading the required libraries
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(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.1.3
```

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

```
## Loaded glmnet 4.1-4
```

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
attach(Carseats)  
summary(Carseats)
```

```
##      Sales      CompPrice      Income      Advertising  
## Min.   : 0.000   Min.   : 77   Min.   : 21.00   Min.   : 0.000  
## 1st Qu.: 5.390   1st Qu.:115   1st Qu.: 42.75   1st Qu.: 0.000  
## Median : 7.490   Median :125   Median : 69.00   Median : 5.000  
## Mean   : 7.496   Mean   :125   Mean   : 68.66   Mean   : 6.635  
## 3rd Qu.: 9.320   3rd Qu.:135   3rd Qu.: 91.00   3rd Qu.:12.000  
## Max.   :16.270   Max.   :175   Max.   :120.00   Max.   :29.000  
##      Population      Price      ShelfLoc      Age      Education  
## Min.   : 10.0   Min.   : 24.0   Bad   : 96   Min.   :25.00   Min.   :10.0  
## 1st Qu.:139.0   1st Qu.:100.0   Good  : 85   1st Qu.:39.75   1st Qu.:12.0  
## Median :272.0   Median :117.0   Medium:219   Median :54.50   Median :14.0  
## Mean   :264.8   Mean   :115.8           Mean   :53.32   Mean   :13.9  
## 3rd Qu.:398.5   3rd Qu.:131.0           3rd Qu.:66.00   3rd Qu.:16.0  
## Max.   :509.0   Max.   :191.0           Max.   :80.00   Max.   :18.0  
## Urban      US  
## No :118    No :142  
## Yes:282    Yes:258  
##  
##  
##  
##
```

QB1. Build a Lasso regression model to predict Sales based on all other attributes (“Price”, “Advertising”, “Population”, “Age”, “Income” and “Education”). What is the best value of lambda for such a lasso model? (Hint1: Do not forget to scale your input attributes – you can use the `caret` `preprocess()` function to scale and center the data. Hint 2: `glmnet` library expect the input attributes to be in the matrix format. You can use the `as.matrix()` function for converting)

```
#Taking all the input attributes into Carseats.Filtered and scaling the input attributes.
```

```
Carseats.Filtered <- Carseats %>% select( "Price", "Advertising", "Population", "Age", "Income", "Educa
```

```
#Converting the input attributes to matrix format using the glmnet library.
```

```
x <- Carseats.Filtered
```

```
#Storing the response variable into y in matrix format
```

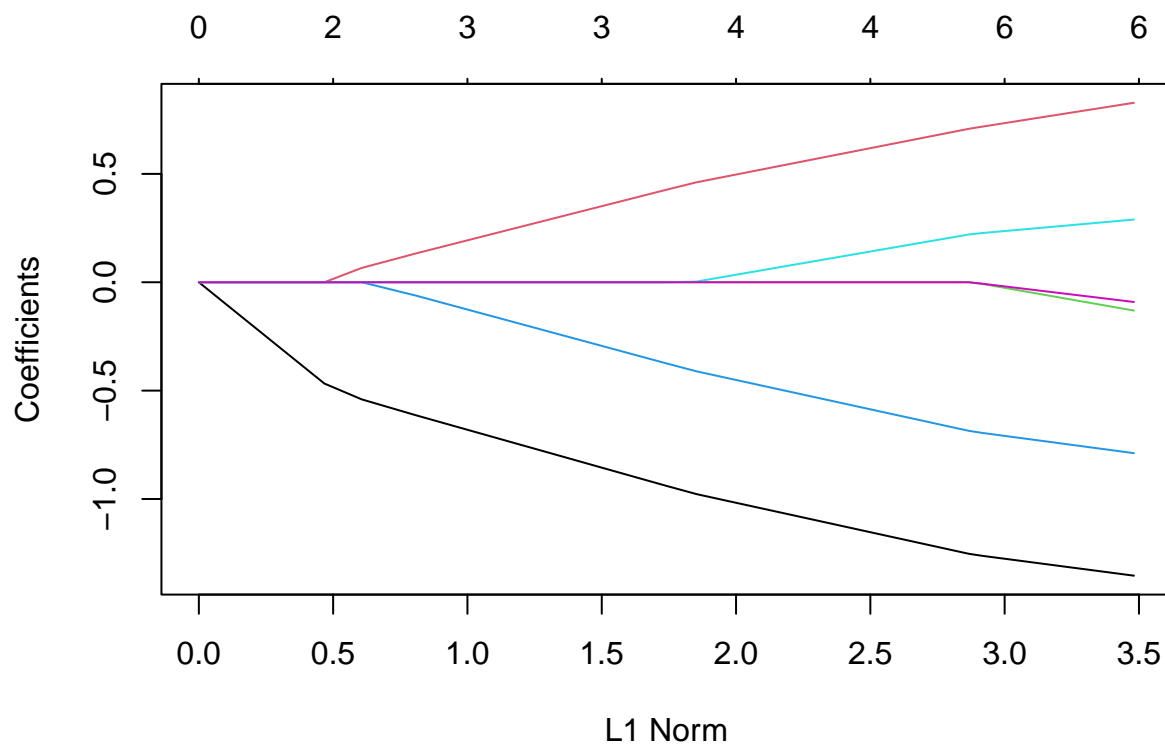
```
y <- Carseats %>% select("Sales") %>% as.matrix()
```

```
#Building the model
```

```
fit = glmnet(x, y)
summary(fit)
```

```
##          Length Class      Mode
## a0         62   -none-   numeric
## beta       372 dgCMatrix S4
## df          62   -none-   numeric
## dim          2   -none-   numeric
## lambda       62   -none-   numeric
## dev.ratio    62   -none-   numeric
## nulldev       1   -none-   numeric
## npasses       1   -none-   numeric
## jerr          1   -none-   numeric
## offset        1   -none-   logical
## call          3   -none-   call
## nobs          1   -none-   numeric
```

```
plot(fit)
```



```
print(fit)
```

```
##
## Call:  glmnet(x = x, y = y)
##
##      Df  %Dev  Lambda
##  1    0   0.00 1.25500
##  2    1   3.36 1.14400
##  3    1   6.15 1.04200
##  4    1   8.47 0.94940
##  5    1  10.39 0.86500
##  6    1  11.99 0.78820
##  7    2  14.62 0.71820
##  8    3  18.08 0.65440
##  9    3  21.12 0.59620
## 10   3  23.64 0.54330
## 11   3  25.73 0.49500
## 12   3  27.46 0.45100
## 13   3  28.91 0.41100
## 14   3  30.10 0.37450
## 15   4  31.12 0.34120
## 16   4  32.13 0.31090
## 17   4  32.97 0.28330
## 18   4  33.67 0.25810
## 19   4  34.25 0.23520
## 20   4  34.73 0.21430
## 21   4  35.13 0.19520
## 22   4  35.46 0.17790
## 23   4  35.74 0.16210
## 24   4  35.97 0.14770
## 25   4  36.16 0.13460
## 26   4  36.31 0.12260
## 27   4  36.45 0.11170
## 28   4  36.55 0.10180
## 29   4  36.64 0.09276
## 30   6  36.75 0.08451
## 31   6  36.86 0.07701
## 32   6  36.95 0.07017
## 33   6  37.02 0.06393
## 34   6  37.09 0.05825
## 35   6  37.14 0.05308
## 36   6  37.18 0.04836
## 37   6  37.21 0.04407
## 38   6  37.24 0.04015
## 39   6  37.27 0.03658
## 40   6  37.29 0.03333
## 41   6  37.30 0.03037
## 42   6  37.32 0.02767
## 43   6  37.33 0.02522
## 44   6  37.34 0.02298
## 45   6  37.35 0.02094
## 46   6  37.35 0.01908
## 47   6  37.36 0.01738
```

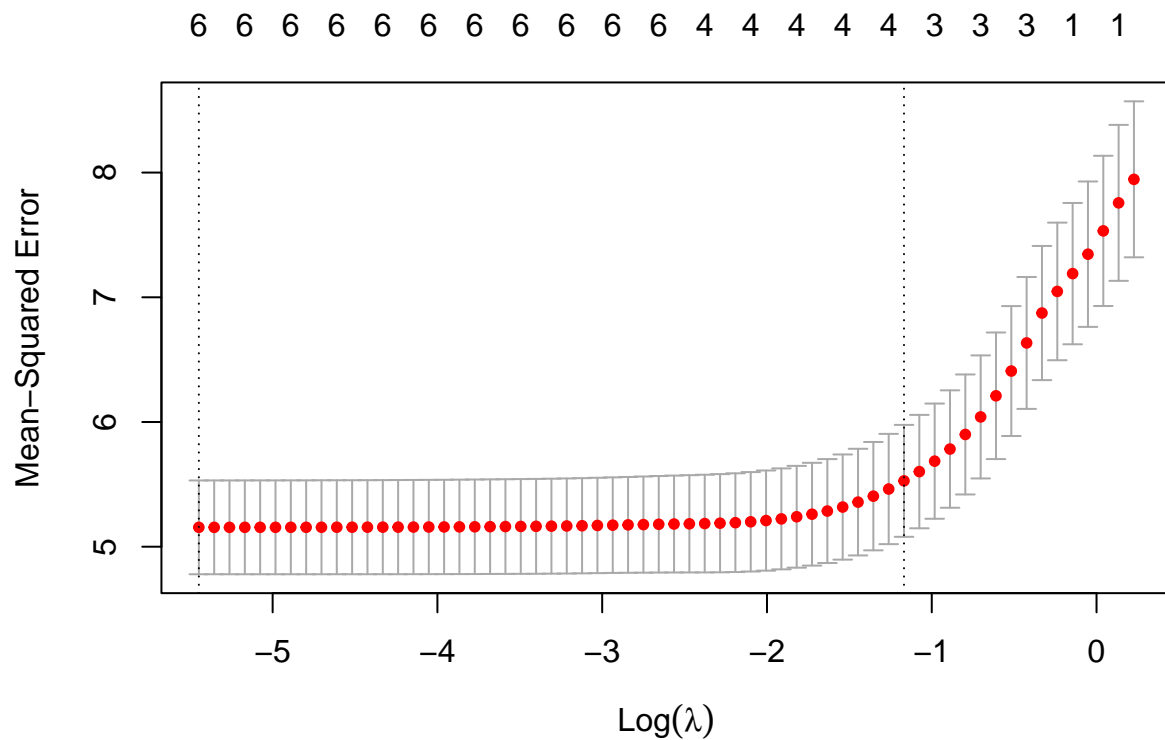
```
## 48 6 37.36 0.01584
## 49 6 37.37 0.01443
## 50 6 37.37 0.01315
## 51 6 37.37 0.01198
## 52 6 37.38 0.01092
## 53 6 37.38 0.00995
## 54 6 37.38 0.00906
## 55 6 37.38 0.00826
## 56 6 37.38 0.00752
## 57 6 37.38 0.00686
## 58 6 37.38 0.00625
## 59 6 37.38 0.00569
## 60 6 37.38 0.00519
## 61 6 37.38 0.00472
## 62 6 37.38 0.00430
```

```
cv_fit <- cv.glmnet(x, y, alpha = 1)

#Finding the minimum lambda value
best_lambda <- cv_fit$lambda.min
best_lambda
```

```
## [1] 0.004305309
```

```
plot(cv_fit)
```



From the above observation, it can be interpreted that the best lambda value is 0.004305309.

QB2. What is the coefficient for the price (normalized) attribute in the best model (i.e. model with the optimal lambda)?

```
best_model <- glmnet(x, y, alpha = 1, lambda = best_lambda)
coef(best_model)
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##              s0
## (Intercept)  7.49632500
## Price       -1.35384596
## Advertising  0.82808291
## Population  -0.13062237
## Age         -0.78855156
## Income      0.28931642
## Education   -0.09102494
```

The coefficient for the price attribute in the best model which is the optimal lambda is -1.35384596.

QB3. How many attributes remain in the model if lambda is set to 0.01? How that number changes if lambda is increased to 0.1? Do you expect more variables to stay in the model (i.e., to have non-zero coefficients) as we increase lambda?

#Lambda when set to 0.01

```
best_model <- glmnet(x, y, alpha = 1, lambda = 0.01)
coef(best_model)
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##              s0
## (Intercept)  7.49632500
## Price       -1.34733223
## Advertising  0.82026088
## Population  -0.12187685
## Age         -0.78190633
## Income      0.28488631
## Education   -0.08502707
```

No coefficients have been eliminated when the lambda is set to 0.01.

#Lambda when set to 0.1.

```
best_model <- glmnet(x, y, alpha = 1, lambda = 0.1)
coef(best_model)
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##              s0
## (Intercept)  7.4963250
## Price       -1.2447745
## Advertising  0.7007230
## Population   .
## Age         -0.6775428
## Income      0.2139222
## Education   .
```

It clearly indicates that the independent attributes have shrunk a bit as two of the coefficients of the attributes have been eliminated when the lambda is set to 0.1.

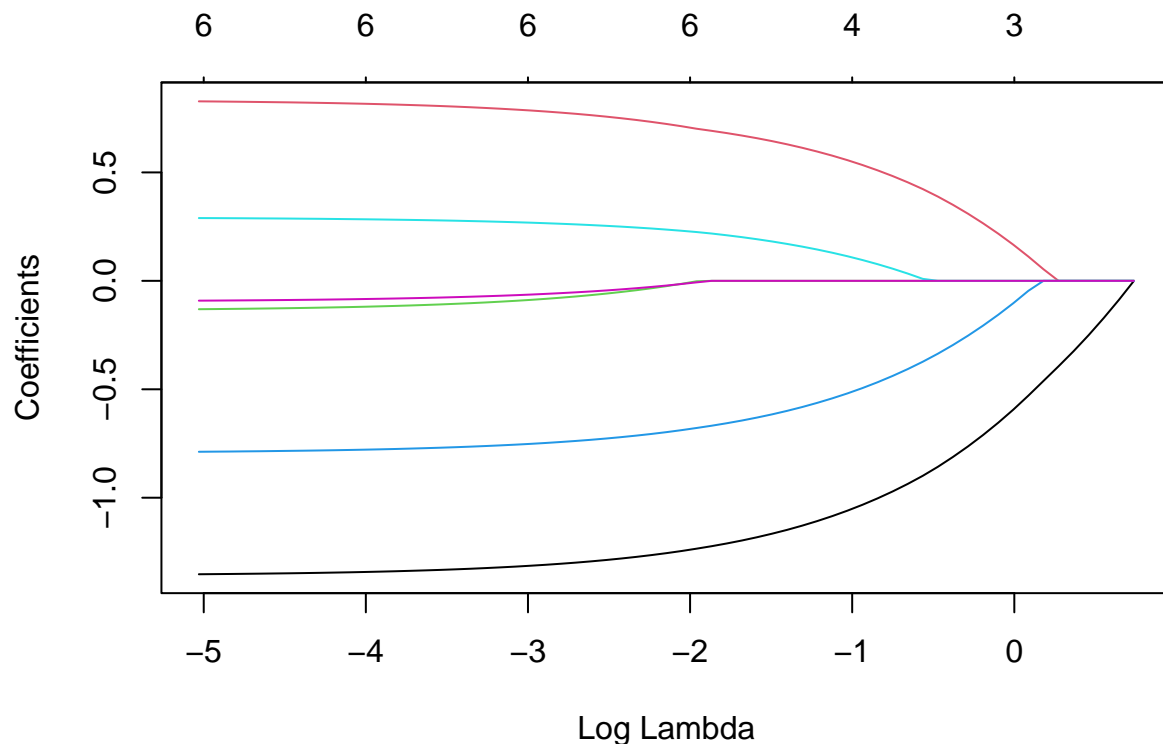
```
#Lambda when set to 0.3
best_model <- glmnet(x, y, alpha = 1, lambda = 0.3)
coef(best_model)
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##              s0
## (Intercept)  7.49632500
## Price       -1.02298693
## Advertising  0.50192192
## Population   .
## Age         -0.45635365
## Income       0.03900787
## Education    .
```

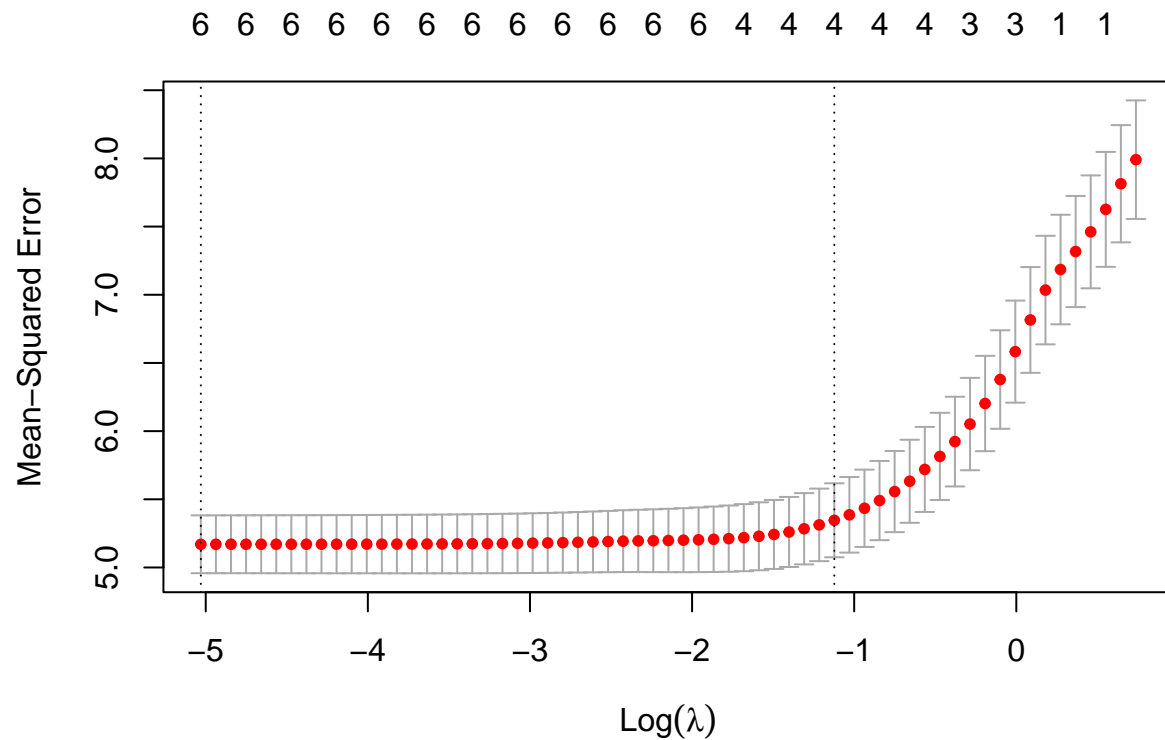
When lambda is set at 0.3, the independent attributes have further shrunk and two of the coefficients of the attributes have already been eliminated. Hence, we can say that as we increase the lambda, we cannot expect to have non- zero coefficients. As discussed previously, Larger the lambda, the more the coefficients are shrunk toward zero and hence, we cannot expect more variables to stay.

QB4. Build an elastic-net model with alpha set to 0.6. What is the best value of lambda for such a model?

```
el_net = glmnet(x, y, alpha = 0.6)
plot(el_net, xvar = "lambda")
```



```
plot(cv.glmnet(x, y, alpha = 0.6))
```



```
summary(el_net)
```

```
##          Length Class      Mode
## a0         63   -none-   numeric
## beta      378 dgCMatrix S4
## df         63   -none-   numeric
## dim         2   -none-   numeric
## lambda      63   -none-   numeric
## dev.ratio  63   -none-   numeric
## nulldev     1   -none-   numeric
## npasses     1   -none-   numeric
## jerr        1   -none-   numeric
## offset      1   -none-   logical
## call        4   -none-   call
## nobs        1   -none-   numeric
```

```
print(el_net)
```

```
##
## Call:  glmnet(x = x, y = y, alpha = 0.6)
##
##      Df %Dev  Lambda
```



```

## 1  0  0.00 2.09200
## 2  1  2.67 1.90600
## 3  1  5.03 1.73700
## 4  1  7.09 1.58200
## 5  1  8.90 1.44200
## 6  1 10.47 1.31400
## 7  2 12.89 1.19700
## 8  3 16.00 1.09100
## 9  3 18.95 0.99370
## 10 3 21.49 0.90540
## 11 3 23.67 0.82500
## 12 3 25.55 0.75170
## 13 3 27.15 0.68490
## 14 3 28.52 0.62410
## 15 4 29.75 0.56860
## 16 4 30.91 0.51810
## 17 4 31.89 0.47210
## 18 4 32.72 0.43020
## 19 4 33.43 0.39190
## 20 4 34.02 0.35710
## 21 4 34.52 0.32540
## 22 4 34.93 0.29650
## 23 4 35.29 0.27020
## 24 4 35.58 0.24620
## 25 4 35.83 0.22430
## 26 4 36.04 0.20440
## 27 4 36.21 0.18620
## 28 4 36.36 0.16970
## 29 4 36.48 0.15460
## 30 6 36.60 0.14090
## 31 6 36.73 0.12830
## 32 6 36.84 0.11690
## 33 6 36.93 0.10660
## 34 6 37.01 0.09709
## 35 6 37.07 0.08846
## 36 6 37.12 0.08060
## 37 6 37.17 0.07344
## 38 6 37.20 0.06692
## 39 6 37.23 0.06097
## 40 6 37.26 0.05556
## 41 6 37.28 0.05062
## 42 6 37.30 0.04612
## 43 6 37.31 0.04203
## 44 6 37.33 0.03829
## 45 6 37.34 0.03489
## 46 6 37.34 0.03179
## 47 6 37.35 0.02897
## 48 6 37.36 0.02639
## 49 6 37.36 0.02405
## 50 6 37.37 0.02191
## 51 6 37.37 0.01997
## 52 6 37.37 0.01819
## 53 6 37.37 0.01658
## 54 6 37.38 0.01510

```

```
## 55 6 37.38 0.01376
## 56 6 37.38 0.01254
## 57 6 37.38 0.01143
## 58 6 37.38 0.01041
## 59 6 37.38 0.00949
## 60 6 37.38 0.00864
## 61 6 37.38 0.00788
## 62 6 37.38 0.00718
## 63 6 37.38 0.00654
```

The best lambda value for such elastic-net model is .00654