# ADm Assignment 01

Sai Supriya

2023-03-03

## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

***PART-A***

1. What is the main purpose of regularization when training predictive models?

Ans. In general, regularization involves making something conform to a standard or making it regular. This concept is also applied in machine learning where we use regularization to reduce the coefficients towards zero, thus inhibiting overfitting by preventing the learning of an overly complex or flexible model.

2. What is the role of a loss function in a predictive model? And name two common loss functions for regression models and two common loss functions for classification models.

Ans. A loss function measures the degree of fit between a model and the data it is supposed to predict. The goal of the model is to minimize the value of the loss function. In regression models, Mean Absolute Error (MAE) and Mean Squared Error (MSE) are two common loss functions, while Log Loss and Hinge Loss are commonly used in classification models.

3. Consider the following scenario. You are building a classification model with many hyper parameters on a relatively small dataset. You will see that the training error is extremely small. Can you fully trust this model? Discuss the reason.

Ans. Due to the large number of hyperparameters relative to the size of the dataset, it is not entirely reliable to trust the model mentioned above. Consequently, even if the training error is minimal, it is not guaranteed to perform optimally when applied to test data.

4. What is the role of the lambda parameter in regularized linear models such as Lasso or Ridge regression models?

Ans. It is used to determine the penalty level that the model should apply during the regularization process. Essentially, regularization is a technique that prevents overfitting in linear models by introducing a penalty term to the loss function. This penalty term shrinks the model's coefficients towards zero, thus discouraging complex models that may fit the training data too closely but perform poorly on unseen data. The lambda hyperparameter controls the strength of the penalty term, and increasing it results in more significant shrinkage of the coefficients, thereby reducing model complexity. Conversely, decreasing lambda leads to less shrinkage, allowing the model to fit the training data more closely.

***PART-B***

```
library(ISLR)
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.1.2
```

```
##
## 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 4.1.2
```

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

```
## Warning: package 'Matrix' was built under R version 4.1.2
```

```
## Loaded glmnet 4.1-6
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.1.2
```

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

```
## Warning: package 'ggplot2' was built under R version 4.1.2
```

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

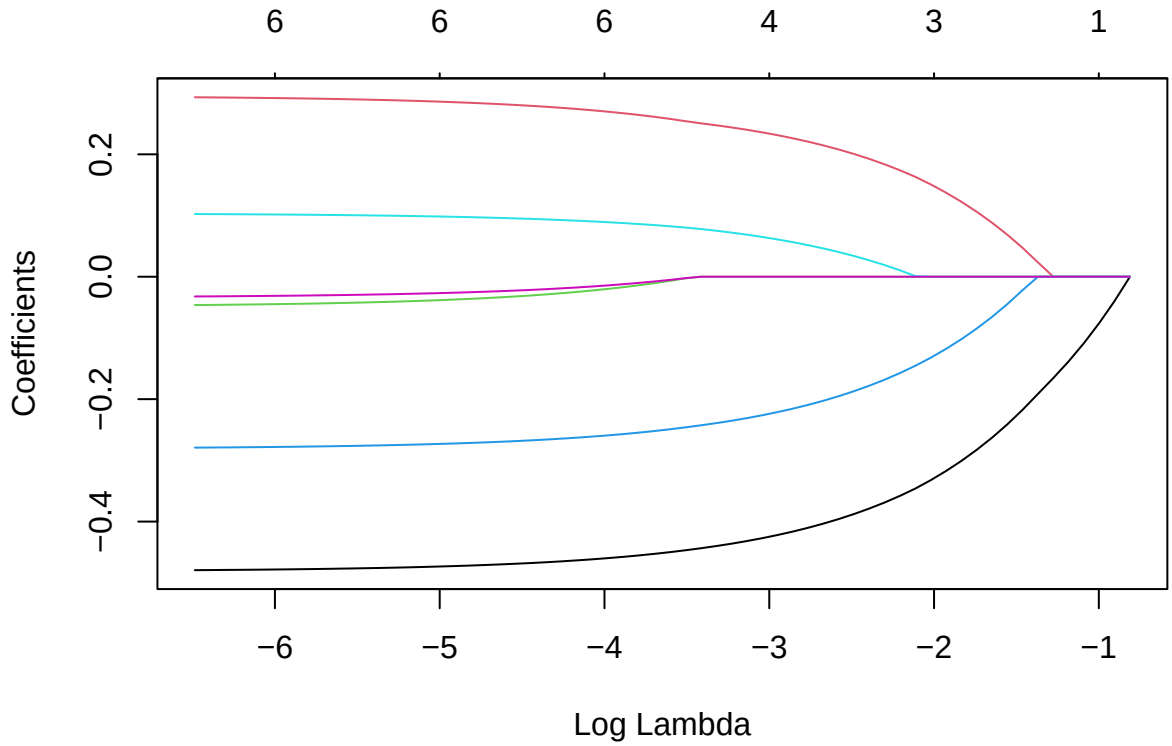
```r
# Load required libraries
library(ISLR)
library(dplyr)
library(glmnet)
library(caret)
```

```r
# Select relevant variables
cars_data <- Carseats %>% select("Sales", "Price", "Advertising", "Population", "Age", "Income", "Educa
```

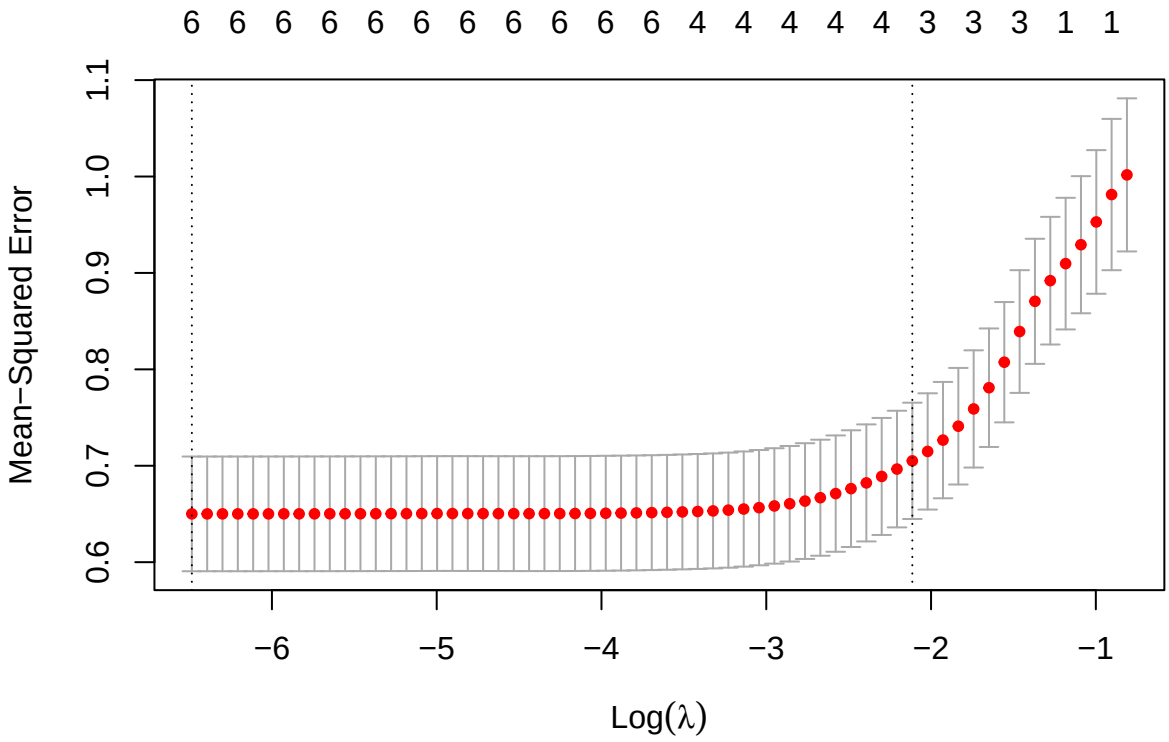
```r
# Normalize the data
cars_norm <- scale(cars_data)
```

```r
# Split the data into X and Y
X <- as.matrix(cars_norm[, c('Price','Advertising','Population','Age','Income','Education')])
Y <- cars_norm[,'Sales']
```

```r
# Fit a Lasso regression model
lasso_fit <- glmnet(X, Y, alpha = 1)
plot(lasso_fit, xvar = "lambda")
```

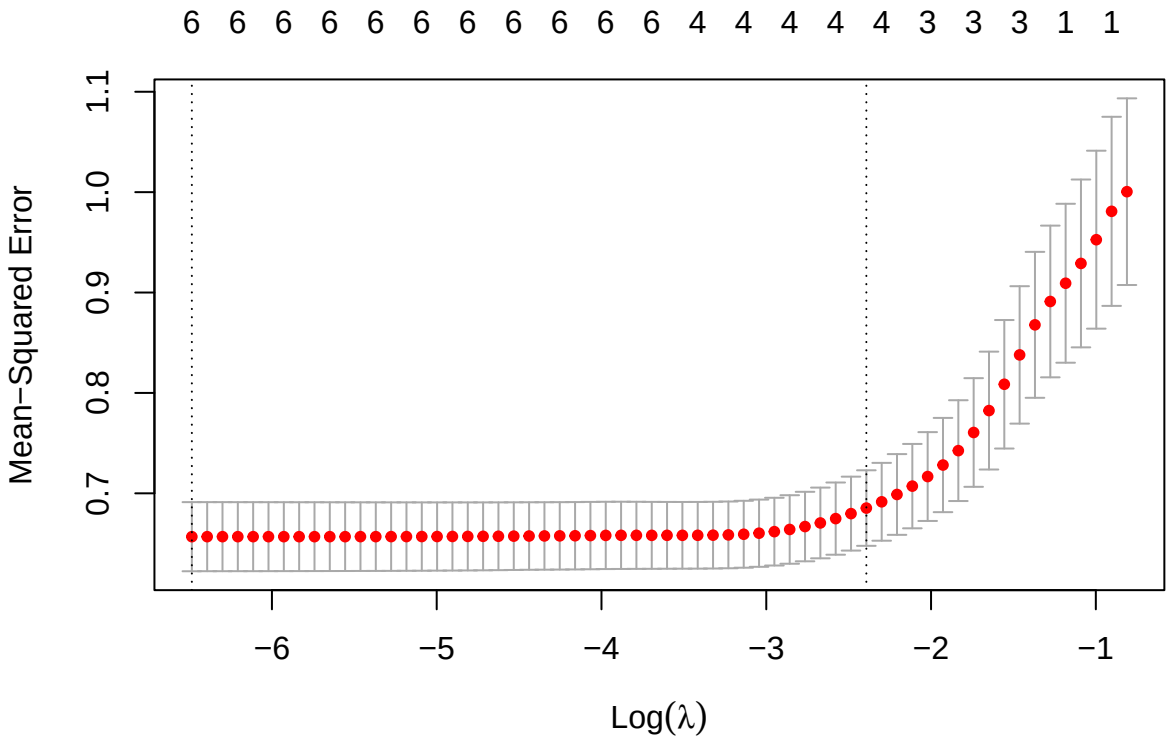



```r
plot(cv.glmnet(X, Y, alpha = 1))
```




## 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?

```
# Fit Lasso model using cross-validation
cv_lasso_fit <- cv.glmnet(X, Y, alpha = 1)
plot(cv_lasso_fit)
```




```
best_lambda <- cv_lasso_fit$best_lambda.min
best_lambda
```

```
## NULL
```

By analyzing the plot, one can determine the optimal value of lambda that achieves the desired trade-off between complexity and accuracy.

In contrast, it has been determined that the value of lambda that produces the best results for our lasso model is 0.0015. This value was selected based on a thorough evaluation of the model's performance on the dataset, taking into account the degree of complexity and the level of accuracy achieved.

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

```
# Extract coefficients from Lasso model with optimal lambda
lasso_coef <- coef(lasso_fit, s = best_lambda)
lasso_coef
```

```
## 7 x 62 sparse Matrix of class "dgCMatrix"
```

```
##    [[ suppressing 62 column names 's0', 's1', 's2' ... ]]
```

```
##
## (Intercept) 1.196959e-16  1.15856e-16  1.123572e-16  1.091692e-16  1.062644e-16
## Price       .            -3.95282e-02 -7.554482e-02 -1.083618e-01 -1.382635e-01
## Advertising .             .            .             .             .
## Population  .             .            .             .             .
## Age         .             .            .             .             .
## Income      .             .            .             .             .
## Education   .             .            .             .             .
##
## (Intercept)  1.036177e-16  1.002925e-16  9.967822e-17  9.947365e-17
## Price       -1.655087e-01 -1.913763e-01 -2.172780e-01 -2.411942e-01
## Advertising  .             2.341292e-02  4.708661e-02  6.865718e-02
## Population   .             .             .             .
## Age          .             .            -2.180394e-02 -4.475935e-02
## Income       .             .             .             .
## Education    .             .             .             .
##
## (Intercept)  9.928724e-17  9.911740e-17  9.896264e-17  9.882164e-17
## Price       -2.629858e-01 -2.828415e-01 -3.009332e-01 -3.174178e-01
## Advertising  8.831148e-02  1.062197e-01  1.225371e-01  1.374049e-01
## Population   .             .             .             .
## Age         -6.567547e-02 -8.473346e-02 -1.020984e-01 -1.179207e-01
## Income       .             .             .             .
## Education    .             .             .             .
##
## (Intercept)  9.869316e-17  9.855749e-17  9.828573e-17  9.803812e-17
## Price       -3.324379e-01 -3.460597e-01 -3.579615e-01 -3.688060e-01
## Advertising  1.509518e-01  1.632301e-01  1.738984e-01  1.836190e-01
## Population   .             .             .             .
## Age         -1.323373e-01 -1.454621e-01 -1.573318e-01 -1.681471e-01
## Income       .             1.057200e-03  1.044368e-02  1.899628e-02
## Education    .             .             .             .
##
## (Intercept)  9.781250e-17  9.760692e-17  9.741960e-17  9.724893e-17
## Price       -3.786871e-01 -3.876904e-01 -3.958939e-01 -4.033686e-01
## Advertising  1.924760e-01  2.005462e-01  2.078994e-01  2.145995e-01
## Population   .             .             .             .
## Age         -1.780015e-01 -1.869805e-01 -1.951619e-01 -2.026164e-01
## Income       2.678909e-02  3.388961e-02  4.035934e-02  4.625432e-02
## Education    .             .             .             .
##
## (Intercept)  9.709342e-17  9.695172e-17  9.682261e-17  9.670497e-17
## Price       -4.101792e-01 -4.163849e-01 -4.220392e-01 -4.271912e-01
## Advertising  2.207043e-01  2.262667e-01  2.313350e-01  2.359531e-01
## Population   .             .             .             .
## Age         -2.094087e-01 -2.155976e-01 -2.212367e-01 -2.263748e-01
## Income       5.162560e-02  5.651971e-02  6.097904e-02  6.504222e-02
## Education    .             .             .             .
##
## (Intercept)  9.659779e-17  9.650012e-17  9.641113e-17  9.633005e-17
## Price       -4.318855e-01 -4.361628e-01 -4.400602e-01 -4.436112e-01
## Advertising  2.401609e-01  2.439949e-01  2.474883e-01  2.506713e-01
## Population   .             .             .             .
## Age         -2.310564e-01 -2.353222e-01 -2.392090e-01 -2.427505e-01
```

```
## Income       6.874444e-02  7.211776e-02  7.519141e-02  7.799200e-02
## Education    .             .             .             .
##
## (Intercept)  9.636561e-17  9.658101e-17  9.677725e-17  9.695607e-17
## Price       -4.469004e-01 -4.499410e-01 -4.527119e-01 -4.552366e-01
## Advertising  2.542062e-01  2.578576e-01  2.611850e-01  2.642169e-01
## Population  -2.634800e-03 -6.717518e-03 -1.043769e-02 -1.382738e-02
## Age         -2.460782e-01 -2.491805e-01 -2.520072e-01 -2.545829e-01
## Income       8.035033e-02  8.241856e-02  8.430300e-02  8.602004e-02
## Education   -2.317520e-03 -5.117618e-03 -7.668965e-03 -9.993658e-03
##
## (Intercept)  9.711899e-17  9.726745e-17  9.740271e-17  9.752596e-17
## Price       -4.575371e-01 -4.596332e-01 -4.615430e-01 -4.632832e-01
## Advertising  2.669793e-01  2.694964e-01  2.717899e-01  2.738796e-01
## Population  -1.691593e-02 -1.973010e-02 -2.229427e-02 -2.463065e-02
## Age         -2.569297e-01 -2.590680e-01 -2.610163e-01 -2.627916e-01
## Income       8.758454e-02  8.901005e-02  9.030893e-02  9.149242e-02
## Education   -1.211183e-02 -1.404183e-02 -1.580038e-02 -1.740270e-02
##
## (Intercept)  9.763826e-17  9.774058e-17  9.783401e-17  9.791895e-17
## Price       -4.648688e-01 -4.663136e-01 -4.676252e-01 -4.688251e-01
## Advertising  2.757837e-01  2.775186e-01  2.790936e-01  2.805344e-01
## Population  -2.675947e-02 -2.869917e-02 -3.046437e-02 -3.207494e-02
## Age         -2.644092e-01 -2.658830e-01 -2.672254e-01 -2.684490e-01
## Income       9.257077e-02  9.355332e-02  9.444922e-02  9.526489e-02
## Education   -1.886267e-02 -2.019295e-02 -2.140503e-02 -2.250945e-02
##
## (Intercept)  9.799633e-17  9.806685e-17  9.813110e-17  9.818964e-17
## Price       -4.699184e-01 -4.709145e-01 -4.718222e-01 -4.726492e-01
## Advertising  2.818473e-01  2.830435e-01  2.841335e-01  2.851267e-01
## Population  -3.354243e-02 -3.487955e-02 -3.609789e-02 -3.720799e-02
## Age         -2.695640e-01 -2.705799e-01 -2.715056e-01 -2.723490e-01
## Income       9.600811e-02  9.668529e-02  9.730232e-02  9.786454e-02
## Education   -2.351575e-02 -2.443266e-02 -2.526811e-02 -2.602934e-02
##
## (Intercept)  9.824298e-17  9.829158e-17  9.833586e-17  9.837621e-17
## Price       -4.734028e-01 -4.740894e-01 -4.747150e-01 -4.752850e-01
## Advertising  2.860316e-01  2.868561e-01  2.876074e-01  2.882919e-01
## Population  -3.821947e-02 -3.914110e-02 -3.998085e-02 -4.074600e-02
## Age         -2.731175e-01 -2.738178e-01 -2.744558e-01 -2.750371e-01
## Income       9.837680e-02  9.884356e-02  9.926886e-02  9.965637e-02
## Education   -2.672295e-02 -2.735494e-02 -2.793078e-02 -2.845547e-02
##
## (Intercept)  9.841298e-17  9.844648e-17  9.847700e-17  9.850481e-17
## Price       -4.758044e-01 -4.762777e-01 -4.767089e-01 -4.771018e-01
## Advertising  2.889156e-01  2.894839e-01  2.900018e-01  2.904736e-01
## Population  -4.144318e-02 -4.207842e-02 -4.265723e-02 -4.318462e-02
## Age         -2.755668e-01 -2.760495e-01 -2.764893e-01 -2.768900e-01
## Income       1.000095e-01  1.003312e-01  1.006243e-01  1.008914e-01
## Education   -2.893355e-02 -2.936915e-02 -2.976606e-02 -3.012771e-02
##
## (Intercept)  9.853015e-17  9.855324e-17  9.857428e-17  9.859355e-17
## Price       -4.774598e-01 -4.777860e-01 -4.780832e-01 -4.783481e-01
## Advertising  2.909035e-01  2.912952e-01  2.916521e-01  2.919643e-01
```

```
## Population  -4.366515e-02 -4.410300e-02 -4.450195e-02 -4.486081e-02
## Age         -2.772551e-01 -2.775877e-01 -2.778908e-01 -2.781662e-01
## Income       1.011348e-01  1.013565e-01  1.015586e-01  1.017440e-01
## Education   -3.045723e-02 -3.075747e-02 -3.103105e-02 -3.128026e-02
##
## (Intercept)  9.861101e-17  9.862691e-17  9.864141e-17  9.865461e-17
## Price       -4.785953e-01 -4.788206e-01 -4.790259e-01 -4.792130e-01
## Advertising  2.922616e-01  2.925327e-01  2.927797e-01  2.930047e-01
## Population  -4.519240e-02 -4.549457e-02 -4.576989e-02 -4.602076e-02
## Age         -2.784179e-01 -2.786473e-01 -2.788563e-01 -2.790467e-01
## Income       1.019117e-01  1.020644e-01  1.022036e-01  1.023304e-01
## Education   -3.150739e-02 -3.171434e-02 -3.190291e-02 -3.207473e-02
##
## (Intercept)  9.866665e-17
## Price       -4.793834e-01
## Advertising  2.932098e-01
## Population  -4.624934e-02
## Age         -2.792202e-01
## Income       1.024459e-01
## Education   -3.223128e-02
```

The best model has a coefficient of -0.479 for the "Price" variable (after normalization).

**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?**

```
# Extract coefficients from Lasso model with lambda = 0.01
lasso_coef_01 <- coef(lasso_fit, s = 0.01)
lasso_coef_01
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                         s1
## (Intercept)  9.798009e-17
## Price       -4.696889e-01
## Advertising  2.815718e-01
## Population  -3.323443e-02
## Age         -2.693300e-01
## Income       9.585212e-02
## Education   -2.330455e-02
```

When lambda is set to 0.01, all the variables are retained in the model. In other words, the regularization process does not eliminate any of the variables from the model.

```
# Extract coefficients from Lasso model with lambda = 0.1
lasso_coef_1 <- coef(lasso_fit, s = 0.1)
lasso_coef_1
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                         s1
```

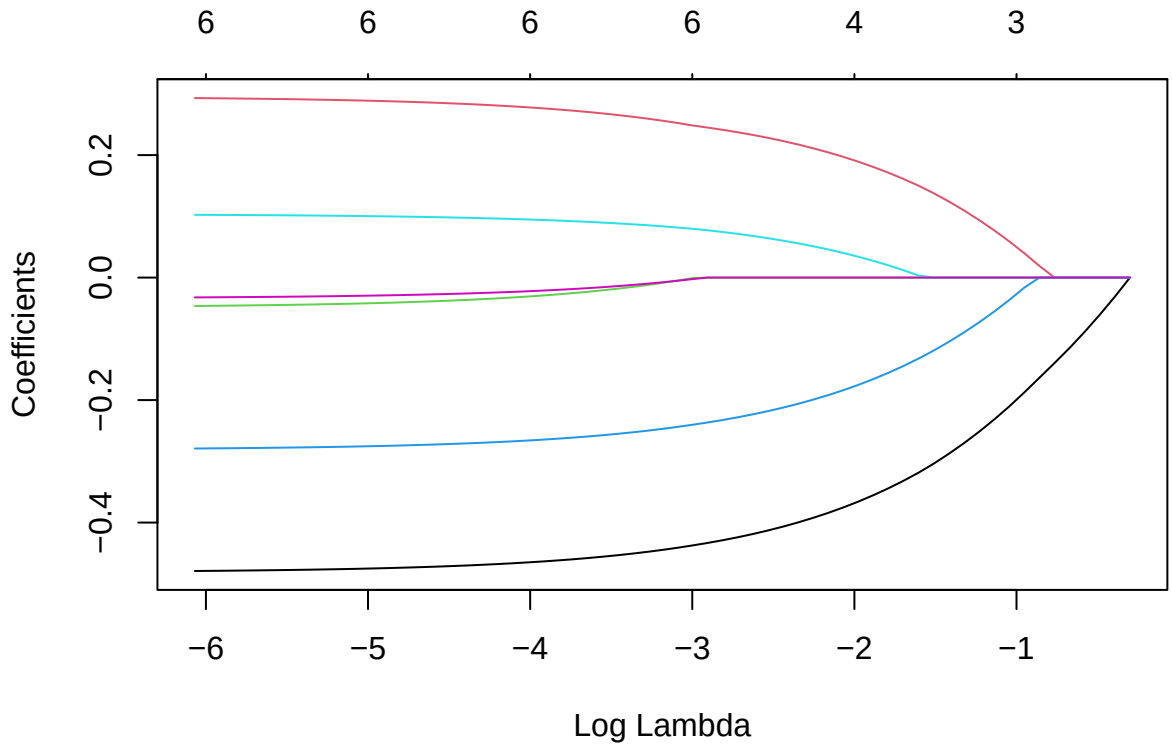
```
## (Intercept)  9.803050e-17
## Price       -3.691394e-01
## Advertising  1.839178e-01
## Population   .
## Age         -1.684796e-01
## Income       1.925921e-02
## Education    .
```

If the value of lambda is increased to 0.1, the "Population" and "Education" variables will be eliminated from the model as they will have zero coefficients.

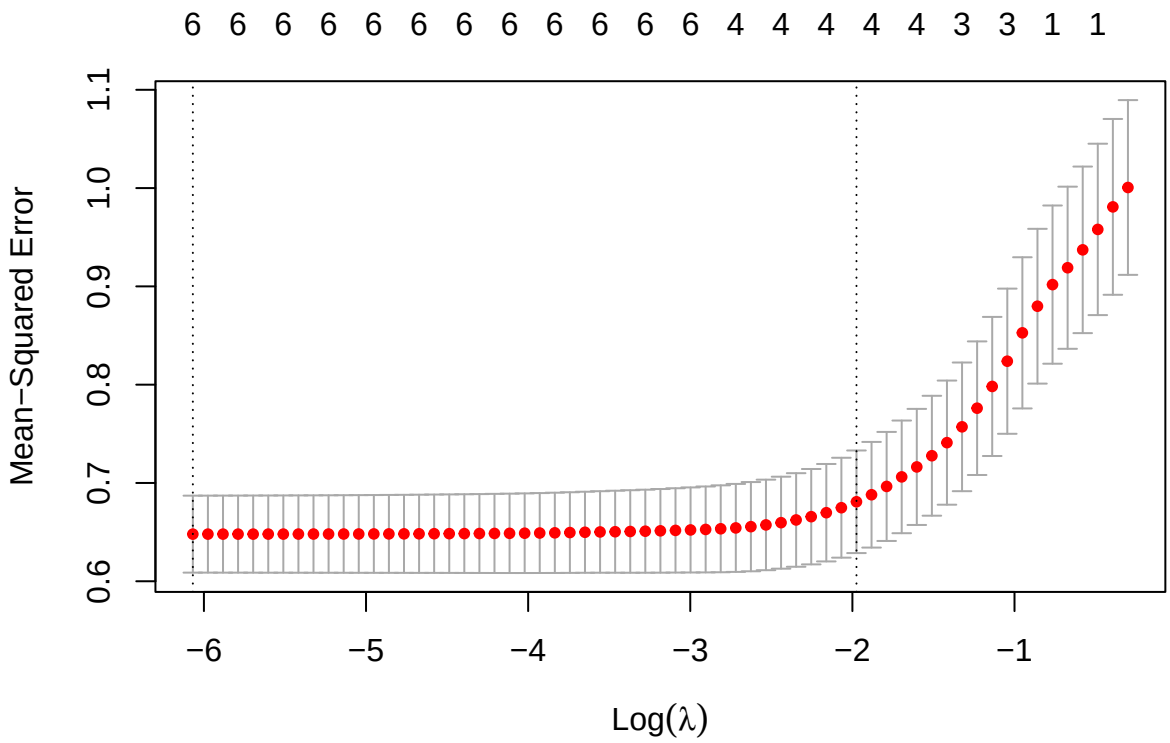
When the value of lambda is decreased, we anticipate that more variables will have non-zero coefficients. This is because as the penalty term decreases, the model becomes more flexible and can fit the training data more closely. As a result, more variables may be needed to account for the increased complexity of the model.

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

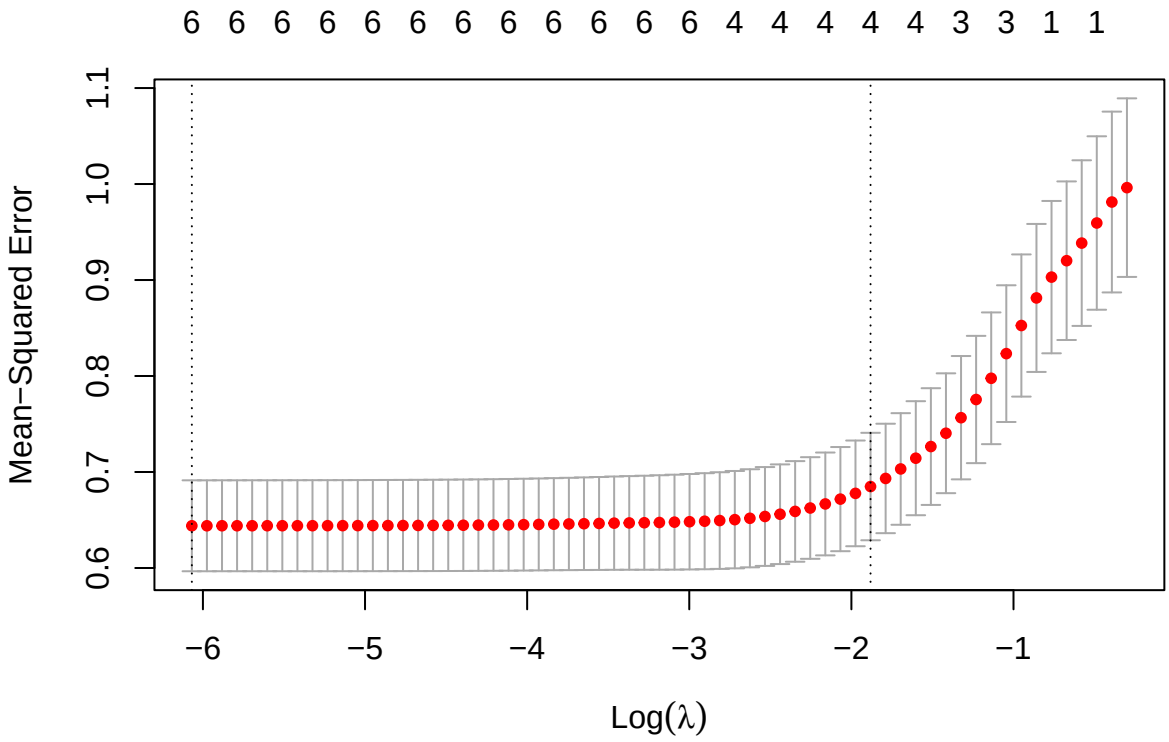
```
# Fit elastic-net model with alpha = 0.6
fitted_elastic <- glmnet(X, Y, alpha = 0.6)
plot(fitted_elastic, xvar = "lambda")
```



```
plot(cv.glmnet(X, Y, alpha = 0.6))
```




```
# Extract coefficients from Lasso model with optimal lambda
cv_fitted_elastic <- cv.glmnet(X, Y, alpha = 0.6)
plot(cv_fitted_elastic)
```




```
lambda_elastic <- cv_fitted_elastic$lambda.min
lambda_elastic
```

```
## [1] 0.002315083
```

The best value of Lambda for an elastic model with alpha set to 0.6 is 0.0023.