Generalized Linear Regression using LASSO

Overview of Regularization

Regularization is used to optimize the performance of the model on the training set so that it does not underfit. As the model becomes too complex, it penalizes the model to avoid overfitting. Basically, Regularization is used to avoid underfitting and overfitting of the model as it improves model's performance by simplifying it.

Lambda is the regularization parameter and it is used to reduce the loss on the training data while minimizing the amplitude of the coefficients of the model.

If the lambda is too large, then we are penalizing all the parameters and all the parameters could be zero. This could be a case of underfitting due to the absense of any parameters(effectively).

If lambda is small, then there are chances that all the parameteres are retained which makes the model complex and could lead to overfitting.

Hence, it is important to find the soft spot while choosing lambda value so that the model neither underfits nor overfits.

In Lasso and Ridge Regresssion models, the hyperparameter lambda uses an l1 penalty (absolute value) on the error term in case of Lasso and for the latter one, it uses the l2 penalty(sum squares of errors).

Purpose of the Project: *The goal of the assignment is to build models to predict the sales of thecarseats ("Sales" attribute) using the other attributes.*

Importing required libraries

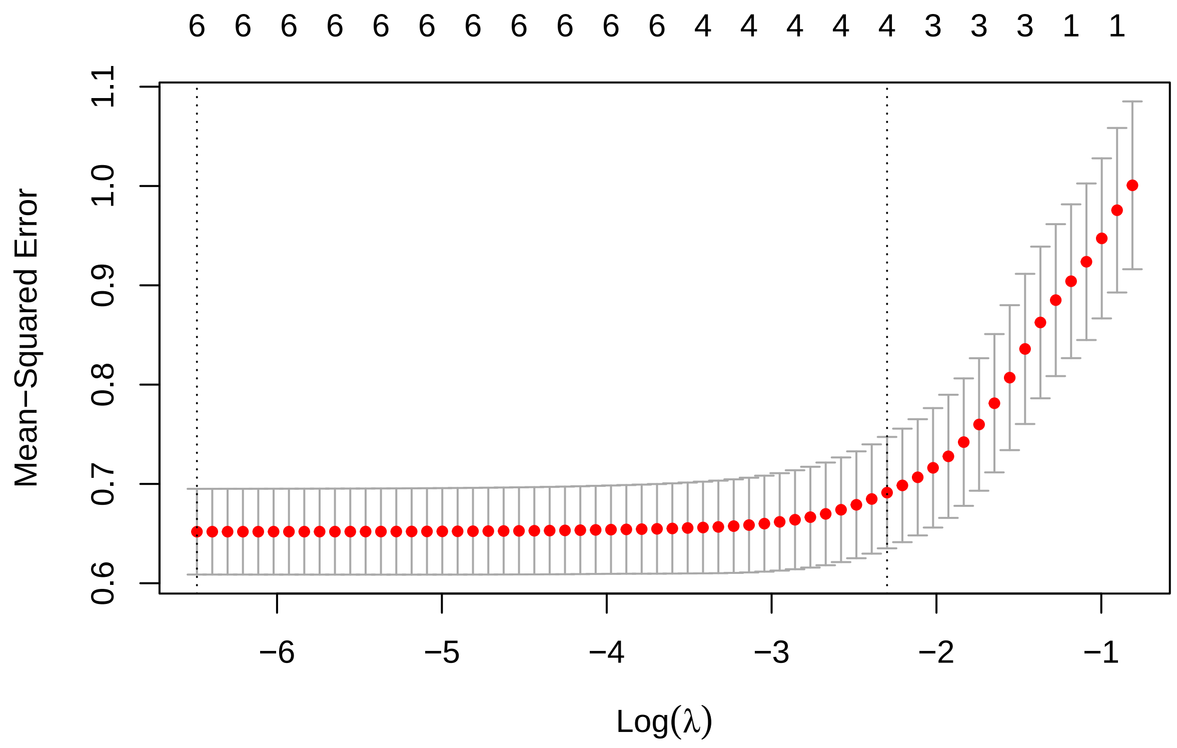
```
library(ISLR)
library(dplyr)
library(glmnet)
library(caret)
```

Data Preparation:

```
#Selecting the required variables from the dataset:
Carseats_Filtered <- Carseats %>% select("Sales", "Price",
"Advertising","Population","Age","Income","Education")

#Normalizing the data:
Normalization <- preProcess(Carseats_Filtered[,1:7],method = c("center","scale"))
Norm_data<-predict(Normalization,Carseats_Filtered)
```

Building a Lasso regression model to predict Sales based on all other attributes ("Price", "Advertising", "Population", "Age", "Income" and "Education").:

```
set.seed(123)
x = model.matrix(Sales~.,Norm_data)[,-1]

y=Norm_data %>% select(Sales) %>% unlist() %>% as.numeric()

cvfit=cv.glmnet(x,y)
plot(cvfit)
```



```
cvfit$lambda.min
```

```
## [1] 0.001524481
```

```
cvfit$lambda.1se
```

```
## [1] 0.1003006
```

Best value of Lambda is 0.001524

```
set.seed(123)
coef(cvfit,s="lambda.min")
```

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

Coeff of Price in the best model is -4.793834e-01

Finding the number of attributes remain in the model for different values of Lambda

```
coef(cvfit,s=0.01)
```

```
## 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
```

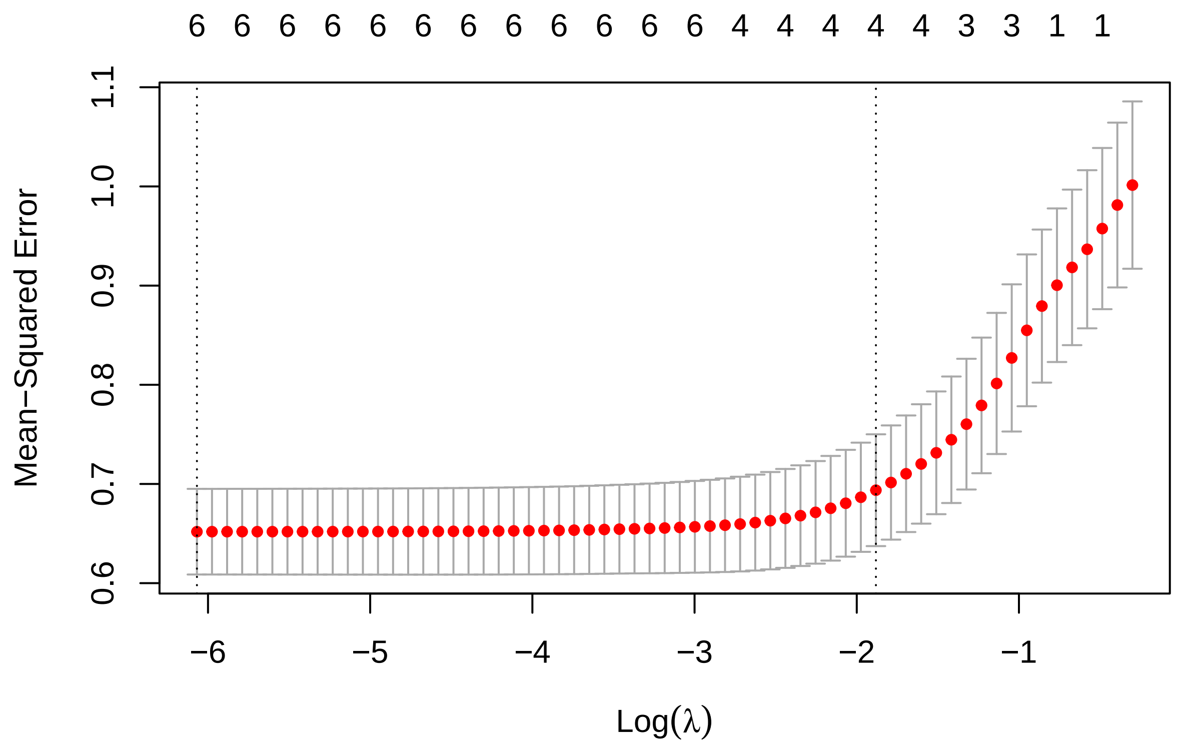
```
coef(cvfit,0.1)
```

```
## 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    .
```

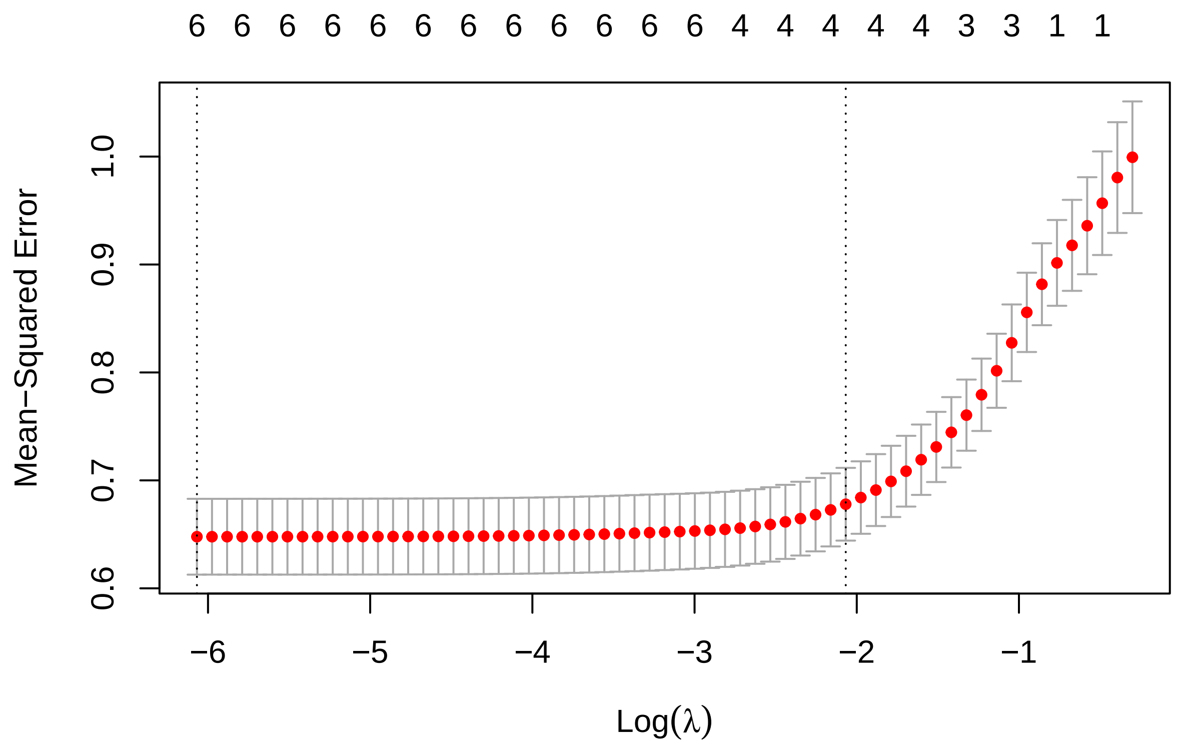
As the lambda increased from 0.01 to 0.1, number of variables with non-zero coefficients decreased.

Building an elastic-net model with alpha set to 0.6

```
set.seed(123)
fit.elasticnet<- cv.glmnet(x,y,alpha=0.6)


plot(fit.elasticnet,xvar="lambda")
```



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



```
fit.elasticnet$lambda.min
```

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

```
#Best value of Lambda is 0.002315
```