Assignment 1

Part A

Question 1: Main Purpose of Regularization when training the models

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.

Question 2: Role of a loss function in predictive Modeling

In predictive Modeling, the ultimate goal is to predict the values as close as possible to the actual values. In other words, we aim to minimize the loss function as the training model is considered to be an optimization problem. Common loss functions in Regression and Classification models are as follows:

Loss functions of Regression Models-

1. Root Mean Square Error(RMSE):

RMSE is calculated by dividing sum of the square of the error by no'of data points and then calculating the square root of the result.

$$RMSE = \sqrt{\sum_{i=1}^{n}(y_i^- - y_i)^2/n}$$

Minimizing RMSE is same as minimizing SSE(Sum Squares Error). However, RMSE is preferred over SSE as it considers the number of data points in it's equation by dividing the residuals by 'n'.

SSE could get affected with the number of data points. For example, if there are more number of data points, then the SSE would be higher. So, the actual value of SSE doesn't speak much about the goodness of the model. However, that drawback can be rectified by using RMSE.

Another advantage RMSE has over SSE is that SSE is expressed in squared units of the dependent variables, whereas RMSE has the same unit as the dependent variable.

2. Coefficient of Determination(R^2):

R square is the proportion of the variance explained by the dependent variable in predicting the independent variable.

It is a square of the correlation coefficient and ranges between 0 and 1.

The higher the R square,better the model fits the data.

Since R square is not expressed in units, it can be used to compare different models.

Loss functions of Classification Models-

1. Log Loss:

Log Loss function indicates how close are the predicted values to the actual values. It is calculated as below:

Log Loss = -(y log(P) +)(1-y)log(1-P))

where, P=Predicted Probability

y= 0 or 1, based on the output values True or False

A model is considered to be a perfect model when it has a log loss of 0

2. ROC and AUC curve:

In ROC curve, the performance of the model is evaluated by a graph which takes two parameters on the X and Y axis- True Positive Rate and False Positive Rate.

AUC is the "Area Under the ROC Curve" and it ignores the actual probabilities as it rank orders the predicted probabilities. It ranges from 0 to 1.

AUC is both Scale-invariant and classification-threshold-invariant.

Question 3:

I wouldn't fully trust the model even if the training error is extremely small. In this case, as the dataset is relatively small and there are many hyper parameters,smaller training error does not represent a good model. In fact, it could also be a case of overfitting as the model can closely follow all the data points and such a model might not perform well on test data which means that the model might not generalize well

Question 4: Role of lambda parameter in regularized linear models:

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).

Part B

Importing required libraries

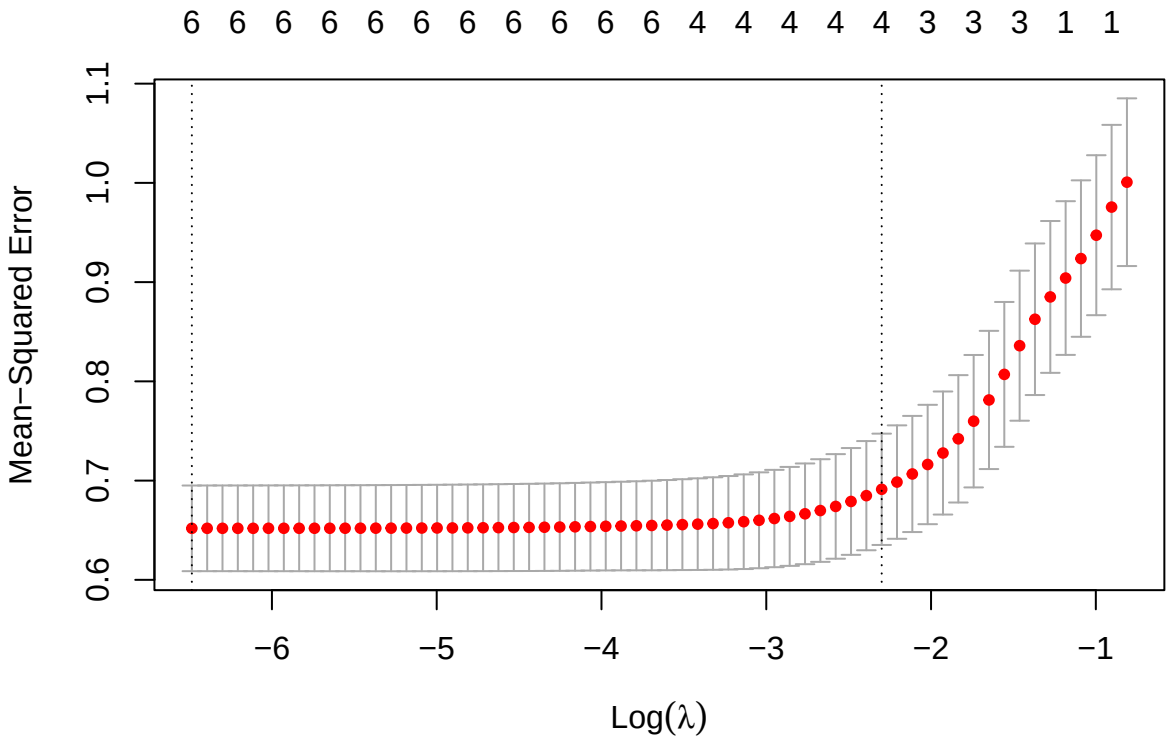
```
library(ISLR)
library(dplyr)
library(glmnet)
library(caret)
```

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)
```

Question 1:

```
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

Question 2:

```
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

Question 3:

```
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
```

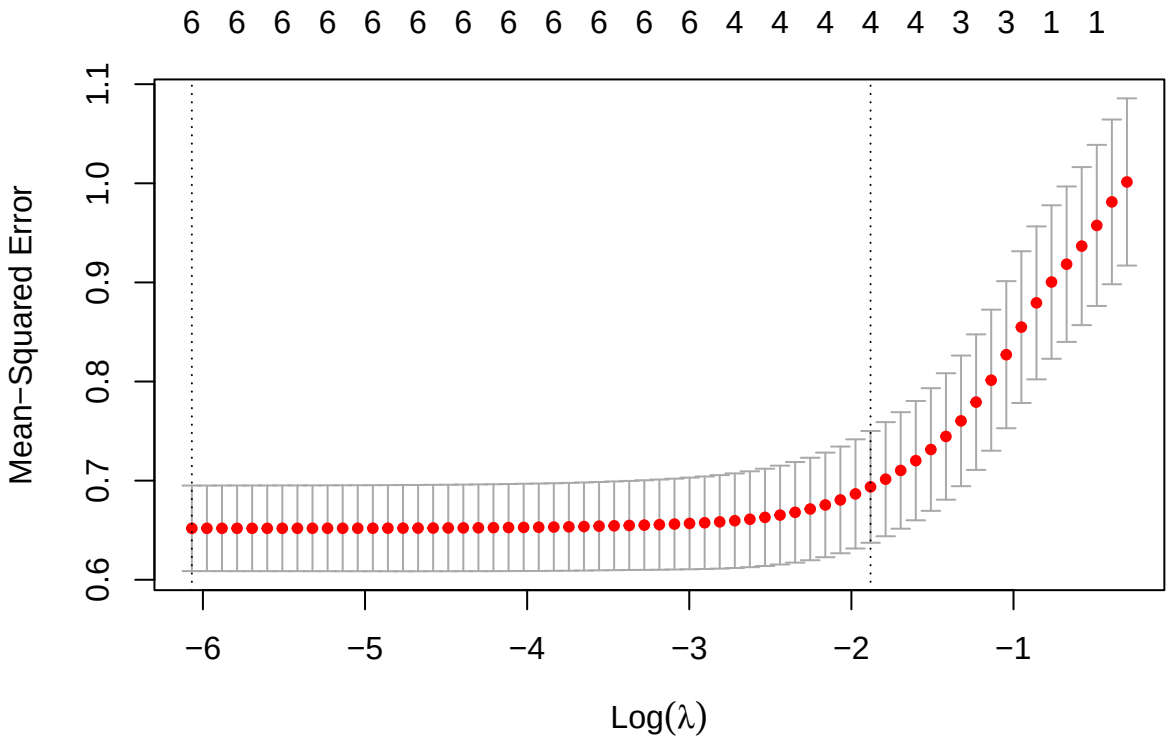
```
coef(cvfit,0.1)
```

```
## 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    .
```

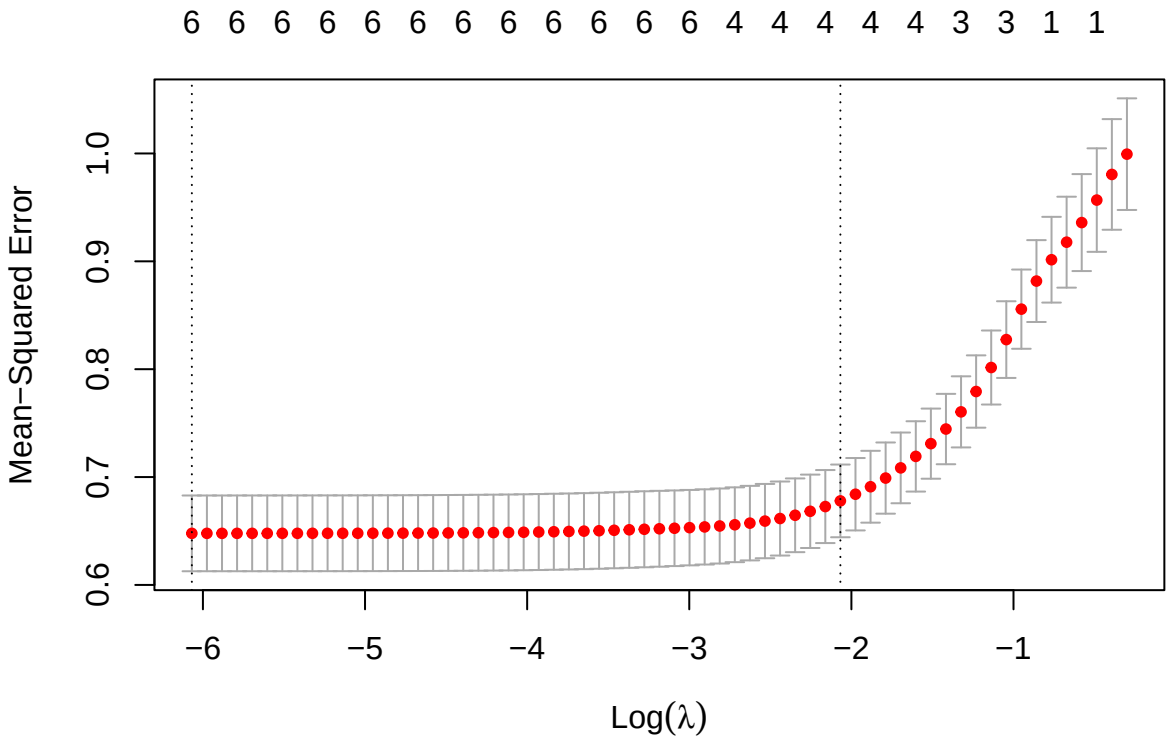
As the lambda increased from 0.01 to 0.1, number of variables with non-zero coefficients decreased.

Question 4:

```
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
```