**ASSIGNMENT – 1  
ADVANCED DATA MINING AND PREDICTIVE ANALYTICS**

***PROFESSOR: Dr Rouzbeh Razavi***

***BY:- NIKHIL REDDY ADDULA***

***PART-A***

***Q1:- What is the main purpose of regularization when training predictive models?***

The primary goal of regularization in machine learning is to prevent model overfitting during the training phase. Overfitting happens when a model is very complicated and overly well matches the training data, resulting in poor performance on new, previously unknown data.

Regularization strategies try to restrict the model's complexity by adding a penalty term to its objective function, making it less likely to overfit. The penalty term effectively decreases the model's potential to overfit the training data and pushes it to learn more generic patterns that are more likely to apply to future data.

L1 regularization (also known as Lasso), L2 regularization (also known as Ridge), and dropout regularization are examples of regularization techniques. L1 regularization introduces a penalty term proportional to the absolute value of the model's weights, encouraging sparse models that discard some of the less essential characteristics. L2 regularization introduces a penalty term proportional to the square of the model's weights, promoting models with lower overall weights. Dropout regularization is the process of randomly removing neurons from the model during training, forcing the surviving neurons to acquire more robust representations.

Regularization is especially crucial when working with sophisticated models with numerous parameters or when dealing with high-dimensional data. It prevents the model from overfitting and can increase its generalization performance on fresh data. Nevertheless, the regularization approach and penalty term strength should be properly calibrated to avoid underfitting or over-regularization.

***Q2:- 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.***

A loss function, also known as a cost function or objective function, is essential in a predictive model since it measures the difference between the model's predictions and the actual target values. The loss function is used during training to improve the model's parameters and find the best fit for the data.

***The purpose of regression problems is to forecast a continuous numerical variable, like as the price of a property or the temperature of a city. The following are two frequent loss functions for regression models:***

**Mean Squared Error (MSE):** This is a common loss function that calculates the average squared difference between the model's predictions and the actual target values. It is defined as the total of the squared discrepancies between the anticipated and actual values divided by the number of samples in the dataset.

**Mean Absolute Error (MAE):** This is another frequent loss function for regression models that evaluates the average absolute difference between the predicted and actual target values. It is defined as the total of the absolute differences between both the predicted and actual values divided by the number of samples included in the data set.

***The purpose of classification problems is to predict a discrete label or category, such as whether an email is a spam or not. For classification models, two frequent loss functions are:***

**Binary Cross-Entropy Loss:** This is a common loss function for binary classification issues that evaluates the difference between the expected probability of a sample belonging to the positive class and the real probability of the sample belonging to the positive class. It is defined as the negative log-likelihood of the true class given the expected probability.

**Categorical Cross-Entropy Loss:** This is a typical loss function for multi-class classification issues that evaluates the difference between the expected and true probability distributions for all classes. It is defined as the sum of the true class's negative log-likelihood given the anticipated probability for all classes.

***Q3:- Consider the following scenario. You are building a classification model with many hyperparameters on a relatively small dataset. You will see that the training error is extremely small. Can you fully trust this model? Discuss the reason.***In this case, it is not prudent to have complete faith in the model based merely on the training error being exceedingly tiny. The model may be overfitting to the training data, which indicates that it is learning the noise in the data rather than the genuine underlying patterns.

Overfitting happens when a model is overly complicated and has too many parameters in comparison to the quantity of available training data. As a result, while the model may perfectly match the training data, it will not generalize well to new, previously unknown data. This is due to the model's learning to memorize the training data rather than learning broad patterns that apply to the problem.

The tiny training error, in this case, indicates that the model is closely fitting the training data. Unfortunately, this does not guarantee that the model will perform well on new, untested data. To confirm that the model generalizes properly, its performance should be evaluated on a different validation or test set. If the model performs well on the validation or test set, it may be fair to trust it.

If the model performs poorly on the validation or test set, it may be required to rethink the hyperparameters or the model architecture to reduce overfitting. Regularization techniques like as L1 or L2 regularization or dropout, for example, can be used to lower the model's capacity and prevent overfitting. Instead, a simpler model with fewer parameters may be more suited for the quantity of available data.

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

In regularized linear models such as Lasso or Ridge regression models, the lambda parameter, also known as the regularization parameter, is critical. The lambda parameter governs the intensity of the regularization penalty applied to the model's objective function during training.

The lambda parameter in Lasso regression governs the amount of L1 regularization performed to the model. L1 regularization introduces a penalty term proportional to the absolute value of the model's coefficients, which promotes sparsity by pushing certain coefficients to zero. The greater the value of lambda, the more severe the penalty and the more coefficients are pushed to zero, resulting in a simpler model with fewer features.

The lambda parameter in Ridge regression governs the degree of L2 regularization performed to the model. L2 regularization introduces a penalty term proportional to the square of the model's coefficients, which encourages lower coefficients and lessens the influence of data outliers. The bigger the lambda value, the stronger the penalty and the smaller the coefficients, resulting in a smoother model that is less susceptible to data noise.

The lambda parameter is often set using cross-validation approaches in both Lasso and Ridge regression to determine the ideal value that balances the trade-off between model complexity and generalization performance. If the lambda parameter is too little, the model will overfit the training data; if it is too high, the model will underfit and perform poorly on fresh data.

***PART-B***

#library functions activation.  
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)

## Loading required package: Matrix

## Loaded glmnet 4.1-6

library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

#getting the inbuilt dataset  
attach(Carseats)  
#summarising dataset  
summary(Carseats)

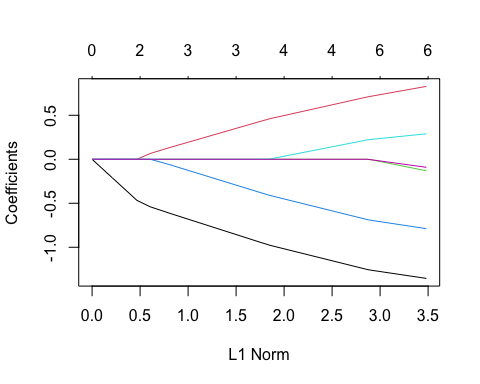
## Sales CompPrice Income Advertising   
## Min. : 0.000 Min. : 77 Min. : 21.00 Min. : 0.000   
## 1st Qu.: 5.390 1st Qu.:115 1st Qu.: 42.75 1st Qu.: 0.000   
## Median : 7.490 Median :125 Median : 69.00 Median : 5.000   
## Mean : 7.496 Mean :125 Mean : 68.66 Mean : 6.635   
## 3rd Qu.: 9.320 3rd Qu.:135 3rd Qu.: 91.00 3rd Qu.:12.000   
## Max. :16.270 Max. :175 Max. :120.00 Max. :29.000   
## Population Price ShelveLoc 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?##  
#Taking the required attributes.  
Filtered <- Carseats %>% select( "Price", "Advertising", "Population", "Age", "Income", "Education") %>% scale(center = TRUE, scale = TRUE) %>% as.matrix()  
# The input characteristics are converted to matrix format using the glmnet package.  
n <- Filtered  
# The response variable is stored in matrix format as " r ".  
r <- Carseats %>% select("Sales") %>% as.matrix()

##QB2. What is the coefficient for the price (normalized) attribute in the best model (i.e. model with the optimal lambda)?##  
# model building  
nr = glmnet(n, r)   
summary(nr)

## 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(nr)



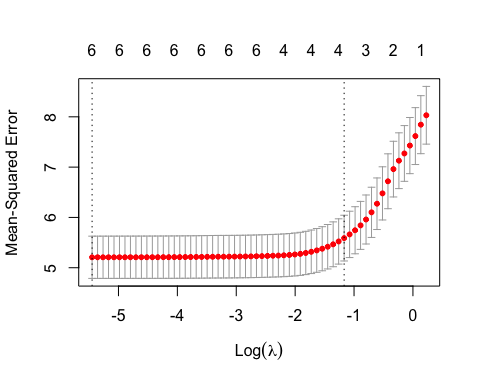
print(nr)

##   
## Call: glmnet(x = n, y = r)   
##   
## 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

nr\_fit <- cv.glmnet(n, r, alpha = 1)  
# minimum lambda value  
min\_lambda <- nr\_fit$lambda.min  
min\_lambda

## [1] 0.004305309

plot(nr\_fit)

 Hence, based on the data above, we can see that there is only 37.38% variation in the target variable, sales with regularization, and a best lambda value of 0.0043.

##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? ##  
best\_m <- glmnet(n, r, alpha = 1, lambda = min\_lambda)  
coef(best\_m)

## 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 Price attribute’s coefficient with the best lambda value is -1.35384596.

# Let us look at the coefficients of the characteristics that remain after lambda is set to 0.01.  
best\_m <- glmnet(n, r, alpha = 1, lambda = 0.01)  
coef(best\_m)

## 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

# Let us look at the coefficients of the characteristics that remain after lambda is set to 0.2.  
best\_m <- glmnet(n, r, alpha = 1, lambda = 0.2)  
coef(best\_m)

## 7 x 1 sparse Matrix of class "dgCMatrix"  
## s0  
## (Intercept) 7.4963250  
## Price -1.1338807  
## Advertising 0.6013225  
## Population .   
## Age -0.5669482  
## Income 0.1264650  
## Education .

# Let us look at the coefficients of the characteristics that remain after lambda is set to 0.3.  
best\_m <- glmnet(n, r, alpha = 1, lambda = 0.3)  
coef(best\_m)

## 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 .

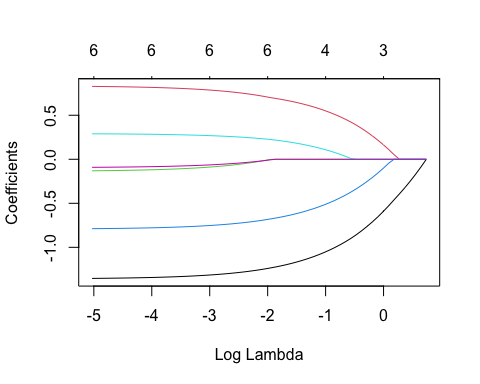
When the lambda value is 0.3, we can observe that two of the attribute coefficients are deleted and the independent attributes have shrunk even further.

# Let us look at the coefficients of the characteristics that remain after lambda is set to 0.5.  
best\_m <- glmnet(n, r, alpha = 1, lambda = 0.5)  
coef(best\_m)

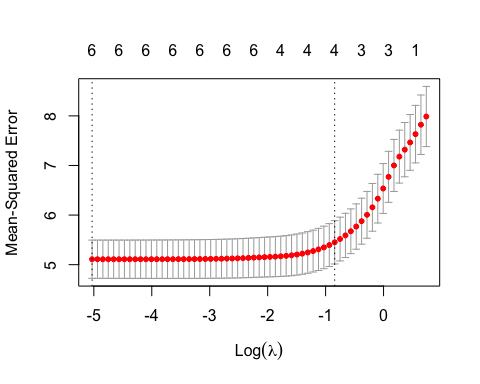
## 7 x 1 sparse Matrix of class "dgCMatrix"  
## s0  
## (Intercept) 7.4963250  
## Price -0.7929743  
## Advertising 0.2947434  
## Population .   
## Age -0.2337276  
## Income .   
## Education .

When the lambda value is 0.5, three of the coefficients of the characteristics are deleted, and the independent attributes have shrunk even further.

###QB4. Build an elastic-net model with alpha set to 0.6. What is the best value of lambda for such a model? ###  
# Building an elastic\_net model with alpha = 0.6  
elastic = glmnet(n, r, alpha = 0.6)  
plot(elastic, xvar = "lambda")



plot(cv.glmnet(n, r, alpha = 0.6))



summary(elastic)

## 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(elastic)

##   
## Call: glmnet(x = n, y = r, 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 variation in the dependent variable (Sales) is 37.38, which is explained by the supplied characteristics to perform regularization by setting the alpha value to 0.6 and the optimal lambda value is 0.00654.