**Question 3 HW2**

In this Question we are dealing With shrinkage models namely

* Redge regression
* Lasso Regression
* Principle Component Regression
* Partial Least Squares Regression

In this exercise, we will predict the number of applications received using the other variables in the **College** data set in the ISLR package.

**Step 1**

In this step I Got to know to know about the necessary packages that are required to Excute the models in R and I imported them .And I also imported the Collage Data set That are Inbuilt in the R Studio.

Graphical user interface, text, application, email

Description automatically generated

**Step 2**

Here I Removed The college Names Variables and assigned Numbers to the dataset, The Names of the collages serves no Purpose and it has nothing to do with the Predictions.

A picture containing text

Description automatically generated

**Step 3 Encoding The Categorical Variables**

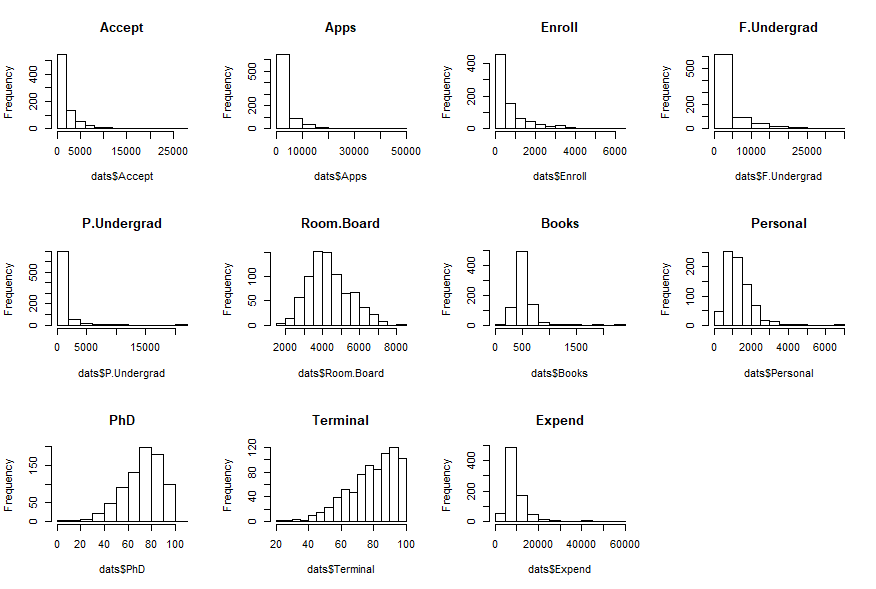
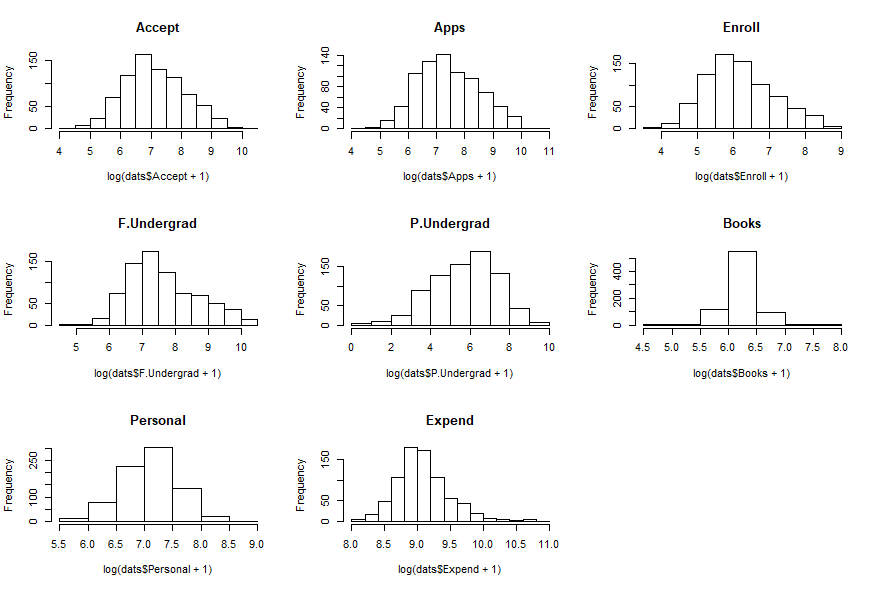
The variable “Private” In the dataset is a categorical variable and it is having values YES, NO, I assigned 0 to No

And 1 to “Yes” so that it becomes Easy for the Computation of the model. By using the “factor” class as shown below

Text

Description automatically generated

**Step 4 Data Transformation**



Variables before Normalization Variables After Normalization

1)When we look at the nature of the Variables closely it has been observed that some of the variables are Either **Positively** skewed or **negatively** Skewed.

2)As the data is not following the Normal distribution with unique variance there is a need here for the Transformation Of the data by applying the Logarithmic Function.

3)The graph is shown above represent the data distribution before and after the Transformation

4)Later I replaced the variables in the dataset with their transformations. And the code for that is shown below

5)The Graphs are plotted using the **Hist** function in r in a 4 X 4 Matrix format using the mfrow() function

6)

Text

Description automatically generated

**Step 6 Splitting the data set into train and test sets\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

the seed() function is set to a random variable to get the same values when executed again. The data is spitted into 80:20 ratio so that 80% of the data is used for training and remaining 20% is used for testing

Text

Description automatically generated

**Step 7 Linear Regression**

I applied linear regression to the training data by with the help of lm() function with Mean R-Squared error of **0**.**9672**

The test MSE appeared to be **0.03704768**

A picture containing text

Description automatically generated

**Step 8 Lasso Regression**

Here for fitting lasso regression I used the library called glmnet() **Glmnet** is a package that fits a generalized linear model via penalized maximum likelihood.

**Type.measure** is set to **mse** indicating that we are using mean squared error to evaluate the model

Alpha a=1 Lasso regression

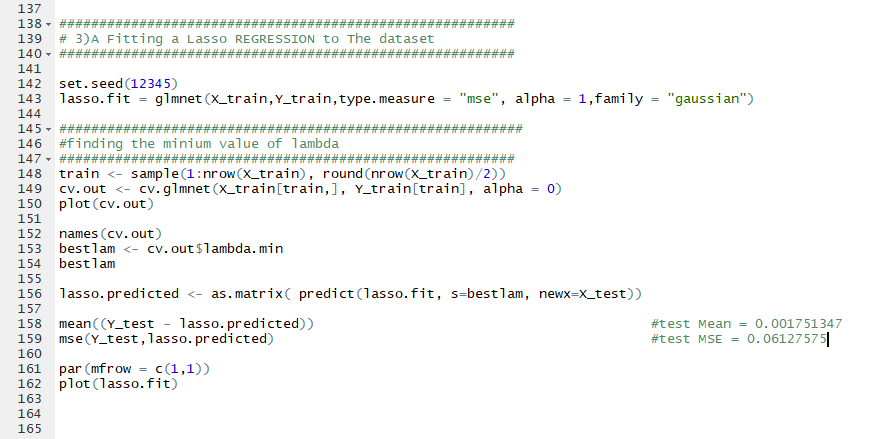
taken cv.glmnet() function ‘cv’ part says we want to use cross validation to obtain optimal values of Lambda

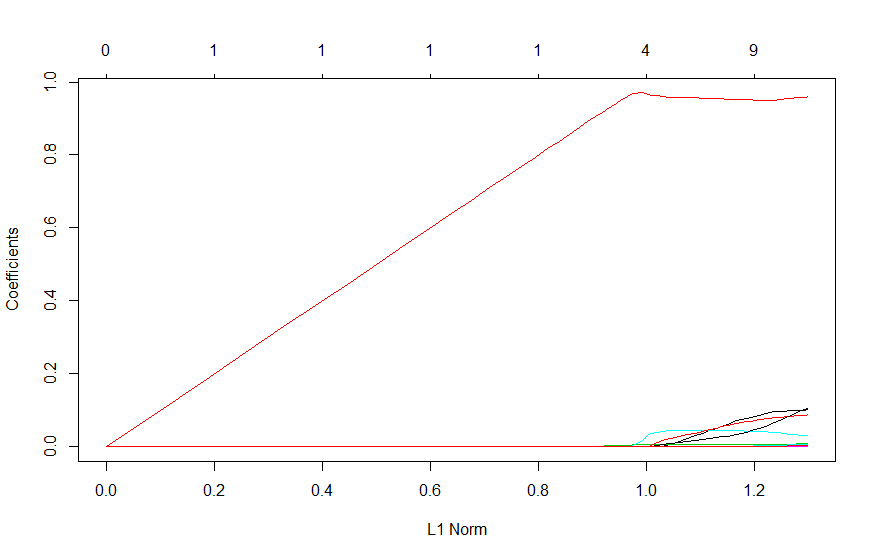
By default cv.glmnet uses 10 fold cross validation

Family is set to **gaussian** indicating that we are using Linear regression in and may be set to **binomial** if we were using logistic regression

* Altogether this call to cv.glmnet() will fit linear regression with a Lasso regression and penalty using 10-fold cross validation to find optimal values of Lambda

The Test MSE Appeared to be **0.06127575**



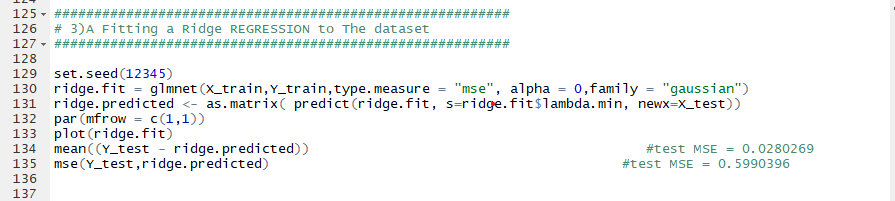


**Step 8 Ridge Regression**

Here for fitting lasso regression I used the library called glmnet() **Glmnet** is a package that fits a generalized linear model via penalized maximum likelihood.

**Type.measure** is set to **mse** indicating that we are using mean squared error to evaluate the model

Alpha a=0 ridge regression



S size of the penalty here is set to one of the optimal values of the lambda stored in **Lambda.min**

here the value for Lambda stored in **ridge.fit** that resulted in the simplest model (i.e. the model with fewest non zero parameters) and was within **one** standard error of the lambda that had the smallest sum

**S = lambda.min** which would be the lambda that resulted in smallest sum

