MODULE FOUR PROJECT

REGULARIZATION OF COLLEGE DATA SET USING RIDGE AND LASSO

Submitted By: **HARSHIT GAUR**
MASTER OF PROFESSIONAL STUDIES IN ANALYTICS
ALY 6015 : INTERMEDIATE ANALYTICS
CRN : 21454
MARCH 20, 2022
WINTER 2022

Submitted To: **PROF. ROY WADA**

INTRODUCTION

Regularization is a technique used to reduce the error by fitting a function appropriately on the given training set and avoid overfitting.

It is a technique used for tuning the function by adding an additional penalty term in the error function. The additional term controls the excessively fluctuating function such that the coefficients don't take extreme values. This technique of keeping a check or reducing the value of error coefficients are called shrinkage methods or weight decay in case of neural networks.

The model will have a low accuracy if it is overfitting. This happens because the model is trying too hard to capture the noise in the training dataset. By noise, we mean the data points that don't really represent the true properties of the data, but random chance.

The fitting procedure involves a loss function, known as residual sum of squares or RSS. The coefficients are chosen, such that they minimize this loss function.

$$RSS = \sum_{k=1}^{n} (actual_i - predicted_i)^2$$

Figure 1.1: Formula for Sum of the Squared Residuals

$$SS = \sum_{i=i}^{n} \left(y_i - \beta_0 - \sum_{j=i}^{p} \beta_0 x_{ij} \right)^2$$

Figure 1.2: Elaborated Formula for Sum of the Squared Residuals

Least Absolute Shrinkage and Selection (LASSO)

One popular method that implements regularization is the Least Absolute Shrinkage and Selection Operator method, commonly known as Lasso. Lasso performs L1 regularization which adds a penalty equal to the absolute value of the magnitude of coefficients. It may shrink coefficients to zero, making it suitable for models showing high levels of multicollinearity and variable elimination/feature selection. The cost function for the Lasso method is shown here:

$$RSS + \lambda \sum_{k=1}^{n} |\beta_j|$$

Figure 1.3: Cost Function for LASSO Regression method

Ridge Regression

The Ridge regression method implements L2 regularization, which adds a penalty equal to the square of the magnitude of coefficients. In Ridge regression, the coefficients will approach zero, but will never equal zero. Ridge regression is suitable for models showing high levels of multicollinearity or when the number of predictors exceeds the number of observations. The cost function for the Ridge regression method is shown here:

$$RSS + \lambda \sum_{k=1}^{n} \beta_j^2$$

Figure 1.4: Cost Function for Ridge Regression method

The Lasso method has a major advantage over Ridge regression and produces a simpler and more interpretable model. That is because Lasso is able to reduce coefficients to zero, effectively performing feature selection and eliminating variables that do not add value to the model. The Lasso method is also generally considered to produce more accurate predictions compared to Ridge regression. Cross-validation can always be used in order to determine which approach is better on a particular data set.

ANALYSIS

With the medium of this project, we will perform regularization techniques on the data set to avoid overfitting of the model.

COLLEGE DATA SET -

The first 6 data points of the college data set has been presented below for a glimpse of the data set and what all variables are included in the data set.

Table 1: College Dataset (with 7 out 18 features)

	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Undergrad	Outstate
Abilene Christian University	Yes	1,660	1,232	721	23	52	2,885	537	7,440
Adelphi University	Yes	2,186	1,924	512	16	29	2,683	1,227	12,280
Adrian College	Yes	1,428	1,097	336	22	50	1,036	99	11,250
Agnes Scott College	Yes	417	349	137	60	89	510	63	12,960
Alaska Pacific University	Yes	193	146	55	16	44	249	869	7,560
Albertson College	Yes	587	479	158	38	62	678	41	13,500

Table 1.1: College Dataset

Exploratory Data Analysis -

We will analyse the data set to find out some insights and investigate the variables.

Table 2: Descriptive Statistics of College Dataset

	n	mean	sd	median	min	max	range	skew	kurtosis
Private*	777	1.73	0.45	2.0	1.0	2.0	1.0	-1.02	-0.96
Apps	777	3,001.64	3,870.20	1,558.0	81.0	48,094.0	48,013.0	3.71	26.52
Accept	777	2,018.80	2,451.11	1,110.0	72.0	26,330.0	26,258.0	3.40	18.75
Enroll	777	779.97	929.18	434.0	35.0	6,392.0	6,357.0	2.68	8.74
Top10perc	777	27.56	17.64	23.0	1.0	96.0	95.0	1.41	2.17
Top25perc	777	55.80	19.80	54.0	9.0	100.0	91.0	0.26	-0.57
F.Undergrad	777	3,699.91	4,850.42	1,707.0	139.0	31,643.0	31,504.0	2.60	7.61
P.Undergrad	777	855.30	1,522.43	353.0	1.0	21,836.0	21,835.0	5.67	54.52
Outstate	777	10,440.67	4,023.02	9,990.0	2,340.0	21,700.0	19,360.0	0.51	-0.43
Room.Board	777	4,357.53	1,096.70	4,200.0	1,780.0	8,124.0	6,344.0	0.48	-0.20
Books	777	549.38	165.11	500.0	96.0	2,340.0	2,244.0	3.47	28.06
Personal	777	1,340.64	677.07	1,200.0	250.0	6,800.0	6,550.0	1.74	7.04
PhD	777	72.66	16.33	75.0	8.0	103.0	95.0	-0.77	0.54
Terminal	777	79.70	14.72	82.0	24.0	100.0	76.0	-0.81	0.22
S.F.Ratio	777	14.09	3.96	13.6	2.5	39.8	37.3	0.66	2.52
perc.alumni	777	22.74	12.39	21.0	0.0	64.0	64.0	0.60	-0.11
Expend	777	9,660.17	5,221.77	8,377.0	3,186.0	56,233.0	53,047.0	3.45	18.59
Grad.Rate	777	65.46	17.18	65.0	10.0	118.0	108.0	-0.11	-0.22

Table 1.2: College Dataset Descriptive Statistics Summary

1. The data set contains 777 data points with 18 features.
2. The mean of Top 10 Percentage students from High School (Top10Perc) at 27.56 which is almost close to the median of 23. This shows that the distribution of this feature might be a normal distribution. We can check the normality using Q-Q Plot and Shapiro Wilks test as well. But, from the test results, we can somewhat say that the feature is deviating from the normal distribution with outliers present.

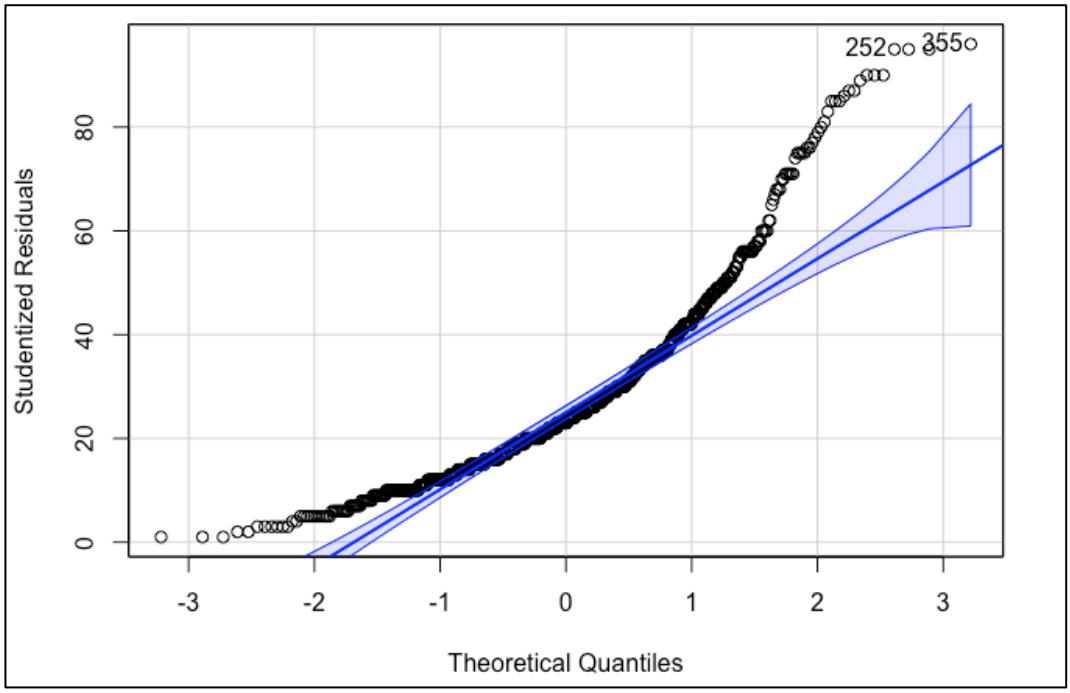

Figure 1.2: Quantile-Quantile Plot for Top10Perc feature

3. The mean of Top 25 Percentage students from High School (Top25Perc) at 55.80 which is almost close to the median of 54. This shows that the distribution of this feature might be a normal distribution. We can check the normality using Q-Q Plot and Shapiro Wilks test as well. From the test results, we can somewhat say that the feature is from the normal distribution with outliers present.
4. This feature is also the least skewed (right) amongst the features available in dataset.

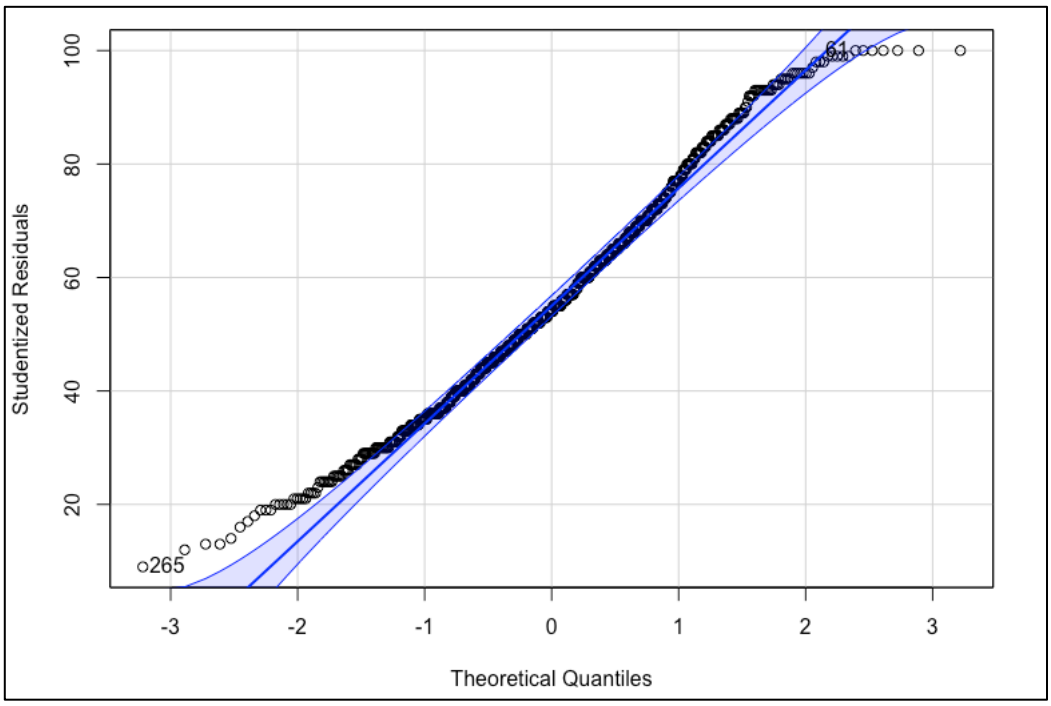

Figure 1.3: Quantile-Quantile Plot for Top25Perc feature

5. The maximum Out-of-state tuition cost from these colleges is $21,700.00.
6. The standard deviations of the feature are also significant which means the data is dispersed throughout the data set.
7. The wins distribution is slightly negatively skewed as suggested by the value of -0.18 of skewness.

Introduction of Interaction Variables and Squared Variables -

Before we start with the regularization techniques, we should add the interaction variables and square variables of the original features of the data set into it. This will help in avoiding the overfitting of models which are caused due to the interactions of these features with themselves and each other.

Table 3: Introduction of Squared Variables of features of College Dataset

Apps_2	Accept_2	Enroll_2	Top10perc_2	Top25perc_2	F.Undergrad_2	P.Undergrad_2	Outstate_2
2,755,600	1,517,824	519,841	529	2,704	8,323,225	288,369	55,353,600
4,778,596	3,701,776	262,144	256	841	7,198,489	1,505,529	150,798,400
2,039,184	1,203,409	112,896	484	2,500	1,073,296	9,801	126,562,500
173,889	121,801	18,769	3,600	7,921	260,100	3,969	167,961,600
37,249	21,316	3,025	256	1,936	62,001	755,161	57,153,600
344,569	229,441	24,964	1,444	3,844	459,684	1,681	182,250,000

Table 1.3: Square Variables added in the College data set

Square variables of the features were introduced to the data set and suffixed the header names using '_2'. The first 6 records with 8 columns from the dataset are being shown in the above table.

Table 4: Introduction of Interaction Variables of features of College Dataset

Apps:Accept	Apps:Enroll	Apps:Top10perc	Apps:Top25perc	Apps:F.Undergrad	Apps:P.Undergrad	Apps:Outstate
2,045,120	1,196,860	38,180	86,320	4,789,100	891,420	12,350,400
4,205,864	1,119,232	34,976	63,394	5,865,038	2,682,222	26,844,080
1,566,516	479,808	31,416	71,400	1,479,408	141,372	16,065,000
145,533	57,129	25,020	37,113	212,670	26,271	5,404,320
28,178	10,615	3,088	8,492	48,057	167,717	1,459,080
281,173	92,746	22,306	36,394	397,986	24,067	7,924,500

Table 1.4: Interaction Variables added in the College data set

Interaction variables of the features were introduced to the data set . Each feature of the data set is interacted (multiplied) with all of the other features and their header names are joined together delimited with the symbol ':'

After the introduction of these interaction and square variables, all of the data set are joined/merged which will be used for regularization models.

Note: The introduction of interaction variables adds up a total number of square of original features into the data set. We are only going to consider at max 50 features in total for regularization models. Therefore, we will consider only the first 15-16 interaction variables to be included in the main data set.

Splitting the Data into Train and Test Sets -

We will create a partition of the data set randomly in the ratio of 80:20 where 80% of the random data will go into the training set and the rest 20% of the data will move into the testing set.

Dependent variable - ***Grad.Rate***

The training and testing data sets have x and y variables which correspond to the predictor variables and predicting variable (responding variable).

RIDGE REGRESSION

Estimating the best value of Lambda using Cross Validation -

The best value of lambda is calculated/estimated using the cv.glmnet() function and the training set (x and y variables). The plot of the mean squared error and Logarithmic of lambda is below with its observations.

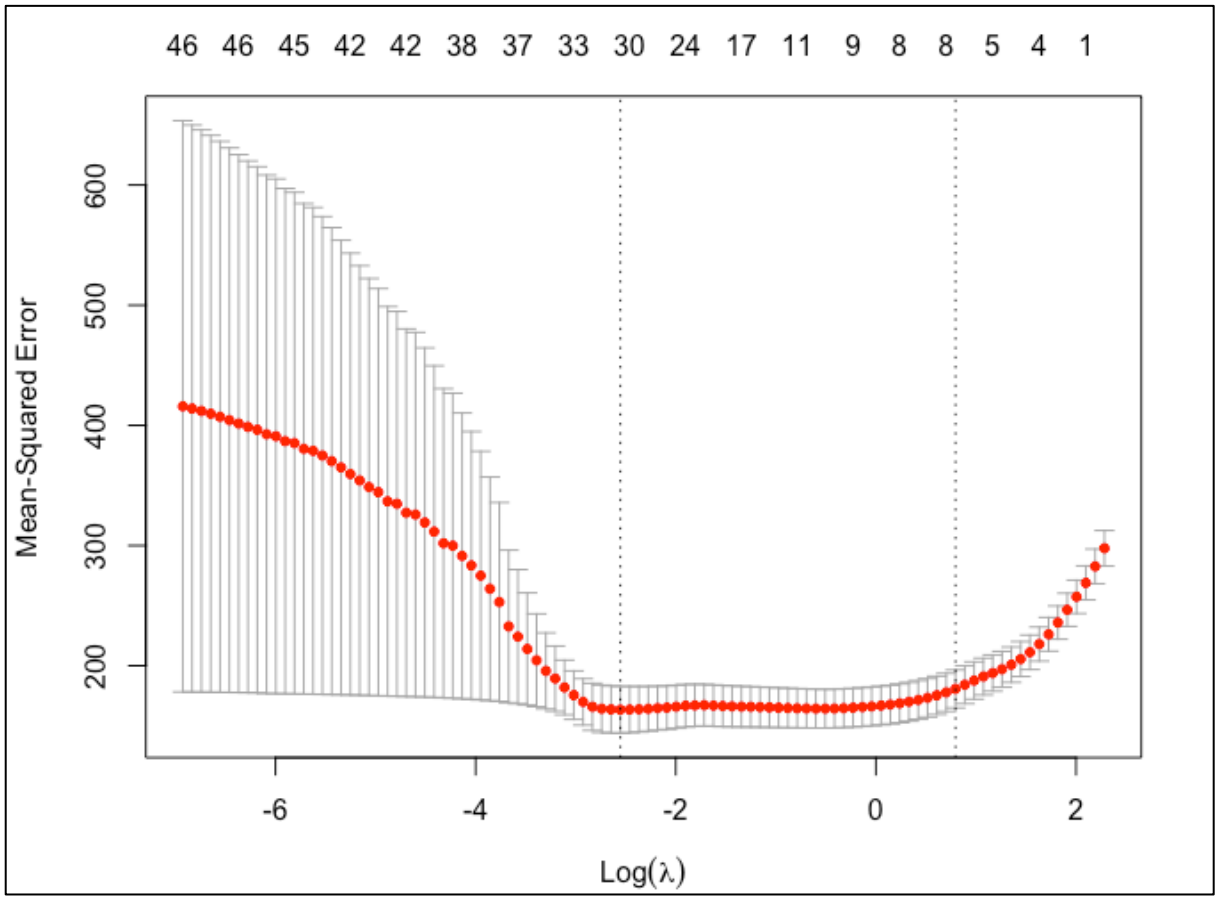

Figure 1.4: Plot to estimate the optimal Lambda value for Ridge Regression

Observations -

From the above plot of mean-squared error in relation with logarithm of lambda, we can figure out that

1. The minimum logarithmic value of Lambda: **-2.5538**
2. The logarithmic value of Lambda at one standard error: **0.7954**
3. The mean squared error of the data set is very high in the range of 600 and more when considering 46 features (original, square, interaction variables) from the data set.
4. The graph shows that for minimum mean-squared error, the number of features which can be considered is around 30-31.
5. For the mean-squared error at 1 standard error, the number of features which can be considered is around 8.

```
Measure: Mean-Squared Error

    Lambda Index Measure    SE Nonzero   Logarithmic Value
min 0.0778    53   163.4 19.45      31         -2.553754
1se 2.2155    17   180.8 15.82       8          0.7954607
```

1. The minimum value of Lambda: **0.0778**
2. The value of Lambda at one standard error: **2.2155**

Ridge Regression Model against Training Data -

The table for coefficients of the ridge regression model against the training data set is below. We will observe the shrinking factor (penalization) of the regularization model and analyse it after it. For this regularization, we're using lambda at 1 standard error.

Table 5: Ridge Regression on Training Data set using Lambda at 1 Standard Error

Features	Coefficients
(Intercept)	45.8695218354269585
PrivateYes	4.0509783276990952
Apps	0.0003202202227976
Accept	0.0004946434243770
Enroll	-0.0002545550663909
Top10perc	0.0684739732402727
perc.alumni	0.2044604169023669
Apps_2	-0.0000000086034345
Accept_2	-0.0000000067842835
Enroll_2	-0.0000000680007633
Top10perc_2	-0.0003200053144885
Top25perc_2	0.0002309548650111
perc.alumni_2	0.0001338940479512
Expend_2	-0.0000000036078681
`Apps:Accept`	-0.0000000108903095
`Apps:Enroll`	0.0000000072379063
`Apps:Top10perc`	-0.0000007823656320

Table 1.5: Ridge Regression on the training data set

Observations -

Firstly, since the total number of variables are 49, we haven't taken all of them out in the above table. We have added the original features along with some square and interaction variables.

From the table of Ridge Regression on the training data set for Regularization, we can figure out that the features with very less coefficient values have been shrunk (or penalized) by the ridge regression model. The more the penalization, the less significant these features are to the data set and model.

Below are some of the variables which have been penalized the most by the model and can be considered very less significant to the model.

1. *Apps_2* -0.000000024907091
2. *`Apps:Accept`* -0.000000025501595
3. `Apps:Enroll` 0.000000163229345

The variables with high value of coefficients have not been shrunk (penalized) by the ridge regression model very much. This means they are significant to the model and the data set and should be included in the predicting the *Grad.Rate.*

Making Prediction on the Training Data Set and Determining Performance -

The fit model was used to predict the *Grad.Rate* feature of the training data set. The performance of the model at predicting the training data feature can be interpreted using the value of root mean square error.

The RMSE value of the regression model is **12.29405**

The interpretation of this root mean square error value is that since the mean value of Grad.Rate is around 65 and the RMSE value of this model is decent enough for this prediction. The residuals of the data points in the training data are not spread very much and the data is concentrated to a good degree around the line of best fit as well.

Making Prediction on the Testing Data Set and Determining Performance -

The fit model was used to predict the *Grad.Rate* feature of the testing data set. The performance of the model at predicting the testing data feature can be interpreted using the value of root mean square error.

The RMSE value of the regression model is **13.59661**

The RMSE value of this model on the testing data is also good enough. The value is not very significantly deviating from the RMSE value of the training data set regression. The residuals of the data points in the testing data are not spread very much and the data is concentrated to a good degree around the line of best fit as well.

Is the Model Overfitting?

Since the values of Root Mean Squared Error (RMSE) for both the training and testing data set are almost same, we can say that the Ridge Regression Model is **not overfitting** on the data set.

LASSO REGULARIZATION

Estimating the best value of Lambda using Cross Validation -

The best value of lambda is calculated/estimated using the cv.glmnet() function and the training set (x and y variables). The plot of the mean squared error and Logarithmic of lambda is below with its observations.

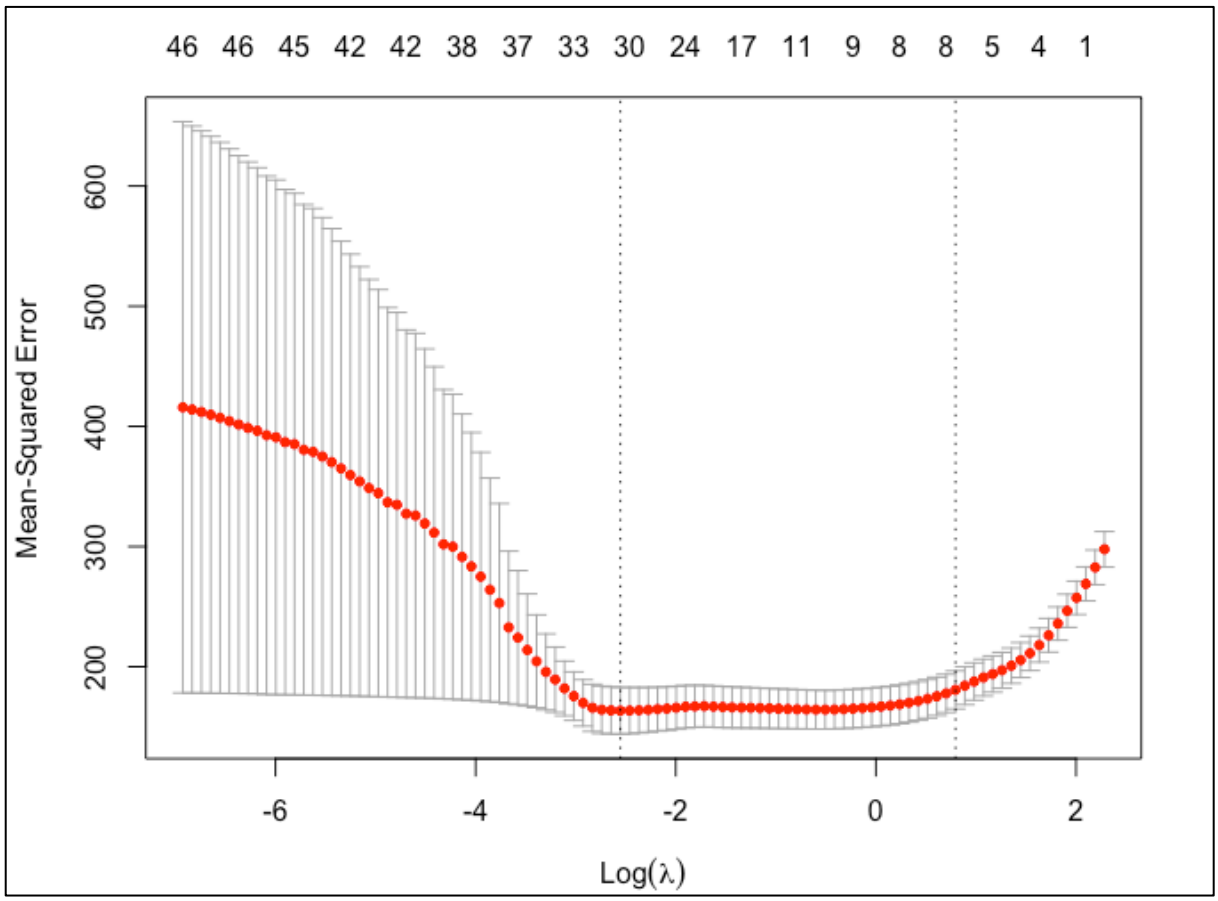

Figure 1.5: Plot to estimate the optimal Lambda value for Lasso Regularization

Observations -

From the above plot of mean-squared error in relation with logarithm of lambda,

1. The minimum logarithmic value of Lambda: **-2.5538**
2. The logarithmic value of Lambda at one standard error: **0.7954**
3. The mean squared error of the data set is very high in the range of 600 and more when considering 46 features (original, square, interaction variables) from the data set.
4. The graph shows that for minimum mean-squared error, the number of features which can be considered is around 30-31.
5. For the mean-squared error at 1 standard error, the number of features which can be considered is around 8.

```
Measure: Mean-Squared Error

    Lambda Index Measure    SE Nonzero   Logarithmic Value
min 0.0778    53   163.4 19.45      31         -2.553754
1se 2.2155    17   180.8 15.82       8          0.7954607
```

3. The minimum value of Lambda: **0.0778**
4. The value of Lambda at one standard error: **2.2155**

LASSO Regularization Model against Training Data -

The table for coefficients of the lasso regularization model against the training data set is below. We will observe the shrinking factor (penalization) of the regularization model and analyse it after it. For this regularization, we're using lambda at 1 standard error.

Table 6: Lasso Regularization on Training Data set using Lambda at 1 Standard Error

Features	Coefficients
(Intercept)	41.946820997659
PrivateYes	0.000000000000
Top10perc	0.014023583990
Top25perc	0.124125288409
F.Undergrad	0.000000000000
Outstate	0.001004768444
Room.Board	0.000582545794
Books	0.000000000000
Personal	-0.000836532367
perc.alumni	0.166873269459

Table 1.6: Lasso Regularization on the training data set

Table 6: Lasso Regularization on Training Data set using Lambda at 1 SE (Continued)

Features	Coefficients
Apps_2	0.000000000000
Accept_2	0.000000000000
Enroll_2	0.000000000000
Top10perc_2	0.000000000000
`Apps:S.F.Ratio`	0.000000000000
`Apps:perc.alumni`	0.000008880482
`Apps:Expend`	0.000000000000

Table 1.6: Lasso Regularization on the training data set (Continued)

Observations -

Firstly, since the total number of variables are 49, we haven't taken all of them out in the above table. We have added the original features along with some square and interaction variables.

From the table of Lasso Regularization on the training data set, we can figure out that the features with coefficient values as 0 have been penalized by the lasso regularization model. This means that these features are either very less significant or almost insignificant to the model and the data set.

Below are some of the variables which have been penalized by the regularization model and can be considered very less significant or almost insignificant to the model.

1. *F.Undergrad* 0.0000000
2. *Books* 0.0000000
3. *Enroll_2* 0.0000000
4. *`Apps:S.F.Ratio`* 0.0000000
5. *`Apps:Expend`* 0.0000000

The variables with high value of coefficients have not been penalized heavily (to either very low coefficient value or 0) by the lasso regularization model. This means they are significant to the model and the data set and should be included in the predicting the *Grad.Rate* feature.

One example of the significant features is **perc.alumni:** 0.166873269459

Making Prediction on the Training Data Set and Determining Performance -

The fit model was used to predict the *Grad.Rate* feature of the training data set. The performance of the model at predicting the training data feature can be interpreted using the value of root mean square error.

The RMSE value of the regularization model is **13.25964**

The interpretation of this root mean square error value is that since the mean value of Grad.Rate is around 65 and the RMSE value of this model is decent enough for this prediction. The residuals of the data points in the training data are not spread very much and the data is concentrated to a good degree around the line of best fit as well.

Making Prediction on the Testing Data Set and Determining Performance -

The fit model was used to predict the *Grad.Rate* feature of the testing data set. The performance of the model at predicting the testing data feature can be interpreted using the value of root mean square error.

The RMSE value of the regularization model is **13.42679**

The RMSE value of this model on the testing data is also good enough. The value is almost close and not very significantly deviating from the RMSE value of the training data set regression. The residuals of the data points in the testing data are not spread very much and the data is concentrated to a good degree around the line of best fit as well.

Is the Model Overfitting?

Since the values of Root Mean Squared Error (RMSE) for both the training and testing data set are almost same, we can say that the Lasso Regularization Model is **not overfitting** on the data set.

Which model performed better?

The values of RMSE for both the Ridge and Lasso Regularization models are **13.6** and **13.4** respectively and the fact that insignificant or very less significant features have not been included in the Lasso regularization model means that the number of features have varied a lot for both the modelling techniques. Therefore, we can confirm that the **Lasso Regularization model performed better** in terms of considering less number of features (getting a best subset of significant features) and achieving a less root mean squared error value against the ridge regression model.

STEPWISE SELECTION *(FEATURE SELECTION)*

We will now use the Stepwise method to find out the best possible subset of features with the best possible optimization of the regression model fit. We are going to use the method of Stepwise-Stepwise Selection (Bidirectional or Both Forward-Backward) on the regression model.

To apply Stepwise Selection method, we will apply linear regression on the training data set with *Grad.Rate* feature as predictor variable using all the features available in it and then perform Stepwise Selection (Bidirectional) on it. The summary of the model is below

Table 7: Linear Regression Model on Training Data set using features from Stepwise Selection

Features	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	53.1589	14.9276	3.5611	0.0004
Apps	0.0022	0.0028	0.7958	0.4265
Accept	0.0013	0.0025	0.5349	0.5929
Enroll	-0.0033	0.0062	-0.5360	0.5922
Top10perc	0.1538	0.1849	0.8317	0.4059
Top25perc	0.6168	0.2017	3.0584	0.0023
F.Undergrad	-0.0006	0.0011	-0.5137	0.6077
P.Undergrad	-0.0009	0.0009	-0.9577	0.3386
Enroll_2	0.0000	0.0000	-0.3837	0.7013
Top10perc_2	0.0000	0.0021	0.0160	0.9873
Top25perc_2	-0.0042	0.0018	-2.3064	0.0214
F.Undergrad_2	0.0000	0.0000	0.0487	0.9611
perc.alumni_2	-0.0016	0.0029	-0.5346	0.5931
Expend_2	0.0000	0.0000	1.5535	0.1209
`Apps:Accept`	0.0000	0.0000	0.5868	0.5576
`Apps:Enroll`	0.0000	0.0000	0.2949	0.7681
`Apps:Top10perc`	0.0000	0.0000	0.1430	0.8864
`Apps:Top25perc`	0.0000	0.0000	-0.4049	0.6857
`Apps:F.Undergrad`	0.0000	0.0000	0.0958	0.9237

Table 1.7: Linear Regression Model on the Training Data Set using features from Stepwise Selection

Adjusted R-Squared value : 0.5078
Root Mean Square Error value : 11.87449

We will now apply Stepwise Selection method on the testing data set and find the best subset of features to use for linear regression and

Table 8: Linear Regression Model on Testing Data set using features from Stepwise Selection

Features	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-78.9092	44.4055	-1.7770	0.0784
Apps	0.0263	0.0113	2.3356	0.0214
Accept	-0.0232	0.0099	-2.3474	0.0208
Enroll	0.0588	0.0207	2.8401	0.0054
Top10perc	0.1241	0.4314	0.2876	0.7742
Top25perc	0.2992	0.5126	0.5837	0.5607
F.Undergrad	-0.0059	0.0032	-1.8577	0.0660
P.Undergrad	0.0052	0.0039	1.3219	0.1891
Enroll_2	0.0000	0.0000	0.0434	0.9654
Top10perc_2	-0.0023	0.0054	-0.4273	0.6700
Top25perc_2	-0.0051	0.0049	-1.0387	0.3013
P.Undergrad_2	0.0000	0.0000	-0.5500	0.5835
Outstate_2	0.0000	0.0000	-0.0858	0.9318
`Apps:Outstate`	0.0000	0.0000	-0.2730	0.7854
`Apps:Room.Board`	0.0000	0.0000	0.5650	0.5733
`Apps:Books`	0.0000	0.0000	-1.9646	0.0521
`Apps:Personal`	0.0000	0.0000	-0.4349	0.6646
`Apps:PhD`	0.0001	0.0002	0.6323	0.5286

Table 1.8: Linear Regression Model on the Testing Data Set using features from Stepwise Selection

Adjusted R-Squared value : 0.514
Root Mean Square Error value : 10.64478

Which model performed better?

The root mean square error (RMSE) values of Stepwise Selection (Bidirectional/Both) is the least from the Stepwise Selection, Lasso Regularization, and Ridge Regression. Therefore, for this data set, I would say Stepwise Selection performed better for me. One of the reasons could be that since various features in the data set are multi-collinear in nature and the fact that the regularization models (Lasso and Ridge) do not work efficiently with multi-collinear data, we can say that because of these reasons, stepwise selection outperformed them a little in my case.

CONCLUSION

We have conducted/performed the *Regularization techniques of LASSO and RIDGE along with Stepwise Selection* on the College data set to predict the Grad.Rate of the data set.

We split our College data set into a training and a testing data set in the ratio of 80-20. Randomly, 80% of data points from the data set were assigned to the training data set, and similarly, the rest 20% of the data points were assigned to the testing data set. We used this training data set to fit our regularization models.

We applied the Ridge Regression model and found out that various features from the pool of original variables, squared variables, interaction variables were either very less significant or almost insignificant analysed from their coefficient values. The root mean square value (RMSE) also came out to be around 12-13 which is decent enough for the data set to be considered good fitted and predicted for both training and testing data. This also aided in analysing that the model was not overfitted.

We applied the Lasso Regularization model as well and found out that various features from the pool of original variables, squared variables, interaction variables were either very less significant or almost insignificant analysed from their coefficient values. These features were removed from the model. The root mean square value (RMSE) for this model came out to be around 11-13 which guaranteed that the Lasso Regularization model has performed better than the Ridge regression model. We were also able to figure out that the model was not overfitted.

Later, we applied Stepwise Selection (bi-directional/both) method to the regression model to find out the best subset of features from the dataset for the model. After applying the regression model using the subset of features found from stepwise selection, the model gave out **Adjusted R-Squared value** of around **0.52** and the root mean square error value (**RMSE**) was around **10.64.** This value is less than the RMSE values of Ridge and Lasso regularization models and therefore, in my case, the Stepwise Selection method performed better than these two.

BIBLIOGRAPHY

1. *Home - RDocumentation*. (2021). Functions in R - Documentation. https://www.rdocumentation.org/

2. ALY 6015 - Prof Roy Wada - *Lesson 4-1 — Regularization* (2022, March), https://northeastern.instructure.com/courses/98028/pages/lesson-4-1-regularization?module_item_id=6647018

3. ALY 6015 - Prof Roy Wada - *Lesson 4-2 — Lasso Regression* (2022, March), https://northeastern.instructure.com/courses/98028/pages/lesson-4-2-lasso-regression?module_item_id=6647019

4. ALY 6015 - Prof Roy Wada - *Lesson 4-3 -- Ridge Regression* (2022, March), https://northeastern.instructure.com/courses/98028/pages/lesson-4-3-ridge-regression?module_item_id=6647020

5. ALY 6015 - Prof Roy Wada - *Module 4 Assignment - Regularization* (2022, March), https://northeastern.instructure.com/courses/98028/assignments/1207977?module_item_id=6647028

6. *Interpreting the Root Mean Squared Error (RMSE)!* (2018, August 14). Data Science Stack Exchange. https://datascience.stackexchange.com/questions/36945/interpreting-the-root-mean-squared-error-rmse

APPENDIX

```
#-------- ALY6015_M4_Lasso&RidgeRegularization_HarshitGaur --------#

print("Author : Harshit Gaur")
print("ALY 6015 Week 4 Assignment - Lasso and Ridge Regularization")

# Declaring the names of packages to be imported
packageList <- c("tidyverse", "ISLR", "caret", "car", "glmnet", "Metrics",
                 "RColorBrewer", "psych", "flextable", "MASS")

for (package in packageList) {
  if (!package %in% rownames(installed.packages()))
  { install.packages(package) }

  # Import the package
  library(package, character.only = TRUE)
}


######################################################################
# College Dataset
######################################################################

# Import/Attach the data set
attach(College)

collegeDatasetHead <- College %>%
  select(Private, Apps, Accept, Enroll, Top10perc, Top25perc, F.Undergrad, P.Undergrad, Outstate)
save_as_docx('College Dataset' = flextable(data = cbind(rownames(head(collegeDatasetHead)),
head(collegeDatasetHead))),
          path = 'Documents/Northeastern University/MPS Analytics/ALY 6015/Class
4/Assignment/Tables/College_Data_Table.docx')

# Get the glimpse of data set
glimpse(College)

describeCollegeFlex <- College %>%
  psych::describe(quant = c(.25, .75), IQR = TRUE) %>%
  select(n, mean, sd, median, min, max, range, skew, kurtosis)

describeCollegeFlex <- round(describeCollegeFlex, 2)
describeCollegeFlex <- cbind(e = rownames(describeCollegeFlex), describeCollegeFlex)

save_as_docx('Descriptive Statistics of College Dataset' = flextable(data = describeCollegeFlex),
          path = 'Documents/Northeastern University/MPS Analytics/ALY 6015/Class
3/Assignment/Tables/College_Desc_Stats_Table_main.docx')


# Normality Check for features using Q-Q Plot and Shapiro-Wilks Test.
qqPlot(College$Top10perc, ylab = "Studentized Residuals", xlab = "Theoretical Quantiles")
shapiro.test(College$Top10perc)

qqPlot(College$Top25perc, ylab = "Studentized Residuals", xlab = "Theoretical Quantiles")
shapiro.test(College$Top25perc)


######################################################################
# Adding Interaction Variables and Square Variables of the features
######################################################################
```

```
interactionSquareFeaturesCollege <- College[, which(!names(College) %in% c("Private", "Grad.Rate"))]

# Adding Square Variables
squareTemp <- setNames(as.data.frame(
  cbind(interactionSquareFeaturesCollege,interactionSquareFeaturesCollege^2))
  , c(names(interactionSquareFeaturesCollege),paste0(names(interactionSquareFeaturesCollege),'_2')))

squareTemp <- squareTemp[, -which(names(squareTemp) %in% names(College))]
save_as_docx('Introduction of Square Variables of features of College Dataset' = flextable(data =
head(squareTemp)),
        path = 'Documents/Northeastern University/MPS Analytics/ALY 6015/Class
4/Assignment/Tables/Sqaure_Variables_Table.docx')

# Adding Interaction Variables
interactionTemp <- as.data.frame.matrix(model.matrix(~ . ^2, data = interactionSquareFeaturesCollege))
interactionTemp <- interactionTemp[, which(!names(interactionTemp) %in% c("(Intercept)", names(College)))]
save_as_docx('Table 5: Introduction of Interaction Variables of features of College Dataset' = flextable(data =
head(interactionTemp[,1:10])),
        path = 'Documents/Northeastern University/MPS Analytics/ALY 6015/Class
4/Assignment/Tables/Interaction_Variables_Table.docx')

# Merge all data sets
College <- cbind(College, squareTemp, interactionTemp[, 1:15])



#####################################################################
# Split data into train and test data
#####################################################################
set.seed(454)
trainIndex <- createDataPartition(College$Grad.Rate, p = 0.80, list = FALSE)
train <- College[trainIndex,]
test <- College[-trainIndex,]

train_x <- model.matrix(Grad.Rate ~ ., train)[, -which(colnames(College) == "Grad.Rate")]
test_x <- model.matrix(Grad.Rate ~ ., test)[, -which(colnames(College) == "Grad.Rate")]

train_y <- train$Grad.Rate
test_y <- test$Grad.Rate


#####################################################################
# Find best value of Lambda using Cross-Validation
#####################################################################
set.seed(454)
cv.lasso_ridge <- cv.glmnet(train_x, train_y, nfolds = 10)
plot(cv.lasso_ridge)
cv.lasso_ridge
save_as_docx('Table 6: Estimating Lambda using Cross Validation' = flextable(data =
model.matrix(cv.lasso_ridge)),
        path = 'Documents/Northeastern University/MPS Analytics/ALY 6015/Class
4/Assignment/Tables/Estimate_Lambda_Table.docx')


#####################################################################
# Optimal Value of Lambda; Minimizes the Prediction Error
# Lambda Min - Minimizes out of sample loss
# Lambda 1SE - Largest value of Lambda within 1 Standard Error of Lambda Min.
#####################################################################
log(cv.lasso_ridge$lambda.min)
log(cv.lasso_ridge$lambda.1se)
```

```
#####################################################################
# Fit the Ridge Regularization Model on Training Set based on Lambdas
# alpha = 1 for Lasso (L2 Regularization)
# alpha = 0 for Ridge (L1 Regularization)
#####################################################################

########## RIDGE Regularization ##########

# Fit the model on training set using lambda.min
model.ridge.train.min <- glmnet(train_x, train_y, alpha = 0, lambda = cv.lasso_ridge$lambda.min)

# Display Regression Coefficients
coef(model.ridge.train.min)
plot(coef(model.ridge.train.min))


# Fit the model on training set using lambda.1se
model.ridge.train.1se <- glmnet(train_x, train_y, alpha = 0, lambda = cv.lasso_ridge$lambda.1se)

# Display Regression Coefficients
coef(model.ridge.train.1se)

save_as_docx('Table 7: Ridge Regression on Training Data set using Lambda at 1 Standard Error' =
flextable(data = cbind(rownames(coef(model.ridge.train.1se)),
as.data.frame.matrix(coef(model.ridge.train.1se)))),
          path = 'Documents/Northeastern University/MPS Analytics/ALY 6015/Class
4/Assignment/Tables/Ridge_Training_1SE_Table.docx')

# Display coefficients of OLS model with no regularization
ols_train <- lm(Grad.Rate ~ ., data = train)
coef(ols_train)

# View RMSE of the full model
predict.ridge.train.ols <- predict(ols_train, new = test)
rmse(test$Grad.Rate, predict.ridge.train.ols)


#####################################################################
# Make Prediction on the Training Data
#####################################################################
predict.ridge.train.1se <- predict(model.ridge.train.1se, newx = train_x)
ridge.train.rmse <- rmse(train_y, predict.ridge.train.1se)


#####################################################################
# Make Prediction on the Testing Data
#####################################################################
predict.ridge.test.1se <- predict(model.ridge.train.1se, newx = test_x)
ridge.test.rmse <- rmse(test_y, predict.ridge.test.1se)

ridge.train.rmse
ridge.test.rmse



########## LASSO Regularization ##########

# Fit the model on training set using lambda.min
model.lasso.train.min <- glmnet(train_x, train_y, alpha = 1, lambda = cv.lasso_ridge$lambda.min)

# Display Regression Coefficients
```

```
coef(model.lasso.train.min)

# Fit the model on training set using lambda.1se
model.lasso.train.1se <- glmnet(train_x, train_y, alpha = 1, lambda = cv.lasso_ridge$lambda.1se)

# Display Regression Coefficients
coef(model.lasso.train.1se)

save_as_docx('Table 7: Lasso Regularization on Training Data set using Lambda at 1 Standard Error' =
flextable(data = cbind(rownames(coef(model.lasso.train.1se)),
as.data.frame.matrix(coef(model.lasso.train.1se)))),
          path = 'Documents/Northeastern University/MPS Analytics/ALY 6015/Class
4/Assignment/Tables/Lasso_Training_1SE_Table.docx')

# Display coefficients of OLS model with no regularization
ols_train <- lm(Grad.Rate ~ ., data = train)
coef(ols_train)

# View RMSE of the full model
predict.lasso.train.ols <- predict(ols_train, new = test)
rmse(test$Grad.Rate, predict.lasso.train.ols)


#####################################################################
# Make Prediction on the Training Data
#####################################################################
predict.lasso.train.1se <- predict(model.lasso.train.1se, newx = train_x)
lasso.train.rmse <- rmse(train_y, predict.lasso.train.1se)


#####################################################################
# Make Prediction on the Testing Data
#####################################################################
predict.lasso.test.1se <- predict(model.lasso.train.1se, newx = test_x)
lasso.test.rmse <- rmse(test_y, predict.lasso.test.1se)

predict.lasso.test.1se

lasso.train.rmse
lasso.test.rmse



#####################################################################
# Stepwise Stepwise Selection (Training Data)
#####################################################################

######### Linear Regression for Model on Training Data #########
regressionFittingFeatures <-  train[sapply(train, is.numeric)]
fit <- lm(formula = Grad.Rate ~ ., data = regressionFittingFeatures)

# Stepwise Stepwise Selection
stepAIC(fit, direction = "both")

# Regression Model Fit using Features from Stepwise Stepwise Selection on Training Data
fit <- lm(formula = Grad.Rate ~ Accept + Enroll + Top10perc + Top25perc +
          P.Undergrad + Outstate + Books + Personal + Terminal + perc.alumni +
          Expend + Accept_2 + Top25perc_2 + Outstate_2 + Room.Board_2 +
          Books_2 + Personal_2 + S.F.Ratio_2 + Expend_2 + `Apps:Room.Board` +
          `Apps:PhD` + `Apps:perc.alumni`, data = regressionFittingFeatures)

lm.summary <- summary(fit)
```

```
save_as_docx('Table 7: Linear Regression Model on Training Data set using features from Stepwise Selection' =
flextable(data = cbind(rownames(lm.summary$coefficients), as.data.frame(round(lm.summary$coefficients,
4)))),
        path = 'Documents/Northeastern University/MPS Analytics/ALY 6015/Class
4/Assignment/Tables/LR_train_Table.docx')
# RMSE of Linear Regression Model
rmse_stepwise_train <- sqrt(mean(lm.summary$residuals^2))


#####################################################################
# Stepwise Stepwise Selection (Testing Data)
#####################################################################

######### Linear Regression for Model on Testing Data #########
regressionFittingFeatures <-  test[sapply(test, is.numeric)]
fit <- lm(formula = Grad.Rate ~ ., data = regressionFittingFeatures)
lm.summary <- summary(fit)

save_as_docx('Table 8: Linear Regression Model on Testing Data set using features from Stepwise Selection' =
flextable(data = cbind(rownames(lm.summary$coefficients), as.data.frame(round(lm.summary$coefficients,
4)))),
        path = 'Documents/Northeastern University/MPS Analytics/ALY 6015/Class
4/Assignment/Tables/LR_test_Table.docx')

# Stepwise Stepwise Selection
stepAIC(fit, direction = "both")

# Regression Model Fit using Features from Stepwise Stepwise Selection on Training Data
fit <- lm(formula = Grad.Rate ~ Apps + Accept + Enroll + F.Undergrad +
        P.Undergrad + Outstate + PhD + S.F.Ratio + Expend + Accept_2 +
        Top25perc_2 + F.Undergrad_2 + Books_2 + PhD_2 + Terminal_2 +
        S.F.Ratio_2 + perc.alumni_2 + Expend_2 + `Apps:Enroll` +
        `Apps:Top10perc` + `Apps:Top25perc` + `Apps:F.Undergrad` +
        `Apps:P.Undergrad` + `Apps:Room.Board` + `Apps:Books` + `Apps:Terminal` +
        `Apps:S.F.Ratio` + `Apps:Expend`, data = regressionFittingFeatures)

lm.summary <- summary(fit)

# RMSE of Linear Regression Model
rmse_stepwise_test <- sqrt(mean(lm.summary$residuals^2))



#-------- END --------#
```