Week 3 Assignment

ALY 6015 Intermediate Analytics

Submitted to:

Joseph Manseau

Date :03/10/2020

Submitted by:

Ashlesha Kshirsagar(001082234)

Introduction

Everything should be made simple as possible, but not simpler – Albert Einstein"

Keeping our model simple is the mail aim for Machine learning. Overfitting happens when the model has freedom to fit the training data. This freedom can be reduced by adding a penalty on different parameters and is called as regularization. Hence, the model will be less likely to fit the noise of the training data and will improve the generalization abilities of the model. There are 3 types of regularization which are

- The L1 regularization (also called Lasso)
- The L2 regularization (also called Ridge)
- The L1/L2 regularization (also called Elastic net)

Ridge regression and the lasso are closely related, but only the Lasso has the ability to select predictors. Like OLS, ridge attempts to minimize residual sum of squares of predictors in a given model. However, ridge regression includes an additional 'shrinkage' term – the square of the coefficient estimate – which shrinks the estimate of the coefficients towards zero. The impact of this term is controlled by lambda (determined separately).

In this report we have used "Hitters" from package ISLR and performed regression analysis and predict Salary on the Hitters data. In this report we have used "**glmnet**" package in order to perform ridge regression and the lasso. We have also used the "**caret**" package which has functions to streamline the model training process for complex regression and classification problems. The Psych Functions are primarily for multivariate analysis and scale construction using factor analysis.

Dataset

We have used “Hitters” Data from Major League Baseball Data from the 1986 and 1987 seasons

Format A data frame with 322 observations of major league players on the following 20

variables.

1. AtBat Number of times at bat in 1986
2. Hits Number of hits in 1986
3. HmRun Number of home runs in 1986
4. Runs Number of runs in 1986
5. RBI Number of runs batted in in 1986
6. Walks Number of walks in 1986
7. Years Number of years in the major leagues
8. CAtBat Number of times at bat during his career
9. CHits Number of hits during his career
10. CHmRun Number of home runs during his career
11. CRuns Number of runs during his career
12. CRBI Number of runs batted in during his career
13. CWalks Number of walks during his career
14. League A factor with levels A and N indicating player’s league at the end of 1986
15. Division A factor with levels E and W indicating player’s division at the end of 1986
16. PutOuts Number of put outs in 1986
17. Assists Number of assists in 1986
18. Errors Number of errors in 1986
19. Salary 1987 annual salary on opening day in thousands of dollars.
20. NewLeague A factor with levels A and N indicating player’s league at the beginning of 1987
21.

We have loaded required libraries and the data.

```
library(glmnet)
library(psych)
library(caret)
library(corrplot)
```

```
data <- read.csv('Hitters.csv')
```

```
data
```

```
data <- na.omit(data)
```

```
data$Salary <- log(data$Salary)
```

```
sum(is.na(data))
```

```
0
```

```
nrow(data)
```

```
263
```

This dataset has some missing data in the response Salaray. We use the na.omit() function the clean the dataset.

```
# Feature scaling
scaled_data <- as.data.frame(scale(data))
```

```
head(scaled_data)
```

Feature scaling is a method used to normalize the range of independent variables. The objective is to improve predictive accuracy and not allow a particular feature impact the prediction due to large numeric value range. Thus, we may need to normalize or scale values under different features such that they fall under common range

```
str(scaled_data)
```

```
'data.frame':  322 obs. of  17 variables:
 $ AtBat  : num  -0.573 -0.43 0.639 0.75 -0.391 ...
 $ Hits   : num  -0.754 -0.431 0.624 0.861 -0.302 ...
 $ HmRun  : num  -1.1218 -0.4329 0.8302 1.0598 -0.0884 ...
 $ Runs   : num  -0.803 -1.034 0.58 0.541 -0.458 ...
 $ RBI    : num  -0.727 -0.383 0.916 1.145 -0.23 ...
 $ Walks  : num  -1.1434 0.0119 1.7218 -0.0805 -0.404 ...
 $ Years  : num  -1.308 1.331 -0.902 0.722 -1.105 ...
 $ CAtBat : num  -1.014 0.344 -0.441 1.282 -0.969 ...
 $ CHits  : num  -0.996 0.179 -0.398 1.31 -0.942 ...
 $ CHmRun : num  -0.79395 -0.00569 -0.07524 1.80267 -0.66643 ...
 $ CRuns  : num  -0.984 -0.113 -0.403 1.404 -0.93 ...
 $ CRBI   : num  -0.904 0.252 -0.192 1.524 -0.853 ...
 $ CWalks : num  -0.922 0.4297 0.0103 0.3511 -0.8509 ...
 $ PutOuts: num  0.56 1.222 2.106 -0.317 1.838 ...
 $ Assists: num  -0.54 -0.467 -0.182 -0.701 -0.489 ...
 $ Errors : num  1.878 0.308 0.936 -0.791 -0.634 ...
 $ Salary : num  NA -0.1349 -0.1238 -0.0795 -0.9862 ...
```

```
scaled_data <- na.omit(scaled_data)
```

```
options(repr.plot.width = 3, repr.plot.height = 5)
boxplot(log(scaled_data$Salary))
```

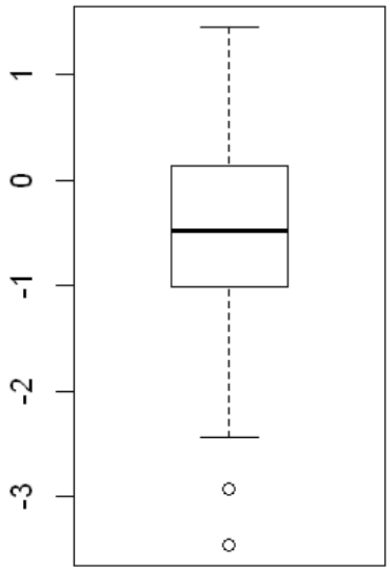

Figure 1.1 Boxplot of scaled data

The boxplot shows that the scaled data is normally distributed.

```
numeric_variables <- which(sapply(scaled_data, is.numeric))
num_names <- names(numeric_variables)
```

We have checked the correlation between Salary and other variables.and ploted as below

```
num_data <- scaled_data[, numeric_variables]
corr <- cor(num_data, use = 'pairwise.complete.obs')

sorted_cor <- as.matrix(sort(corr[, 'Salary'], decreasing = T))
CorHigh <- names(which(apply(sorted_cor, 1, function(x) abs(x)>0.5)))
cor_numVar <- corr[CorHigh, CorHigh]

options(repr.plot.width = 10, repr.plot.height = 8)
corrplot.mixed(corr, tl.col="black", tl.pos = "lt")
```

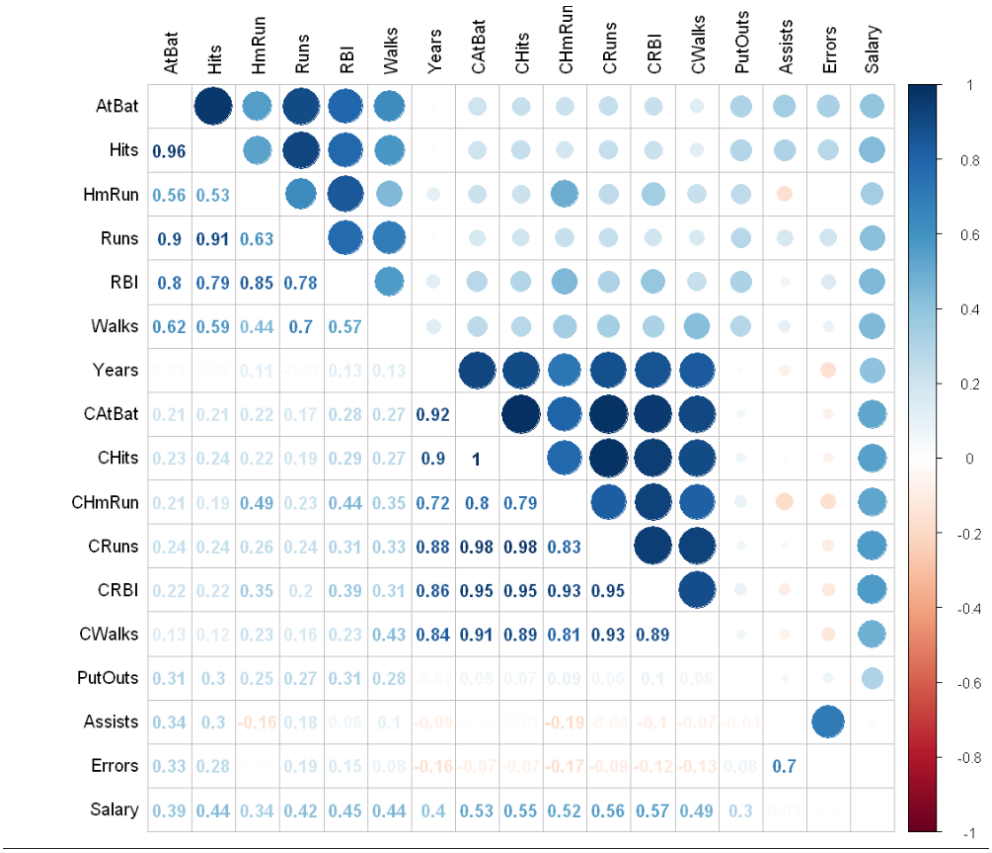

Figure: 1.2 Correlation pattern between different independent variables

From correlation figure we can say that there exists a multicollinearity amongst the independent variables. There exists the correlation more than 90% for many variables such as CRBI, CHmRun. To overcome this problem, we have to use feature selection, regularization, dimensionality reduction etc in this report we have used regression methods Ridge Regression, Lasso, and Elastic Net.

```
set.seed(45)

split <- sample(2, nrow(scaled_data), replace = T, prob = c(0.7, 0.3))
train_data <- scaled_data[split == 1,]
test_data <- scaled_data[split == 2,]
```

We build our model on the training set and evaluate its performance on the test set. To avoid introducing a bias in test using train-data, the train-test split should be performed as a part of data preparation steps.

We set the random seed as 45 for reproducibility. After that we created index for randomly sampling observations for data partitioning. After that we created training and test set. The train set contains 70 %of the data while the test set contains the remaining 30%

```
head(train_data)
```

```
head(test_data)
```

```
nrow(train_data)
nrow(test_data)
```

187

76

```
custom_parameter <- trainControl(method = 'repeatedcv',
                                 number = 10,
                                 repeats = 5,
                                 verboseIter = T)
```

To modify the resampling method, we have used trainControl function . The option ‘repeatedcv ‘ is used to specify the repeated *K*-fold cross-validation . Here we are using 10 folds cross validation which repeats for 5 times

Linear Regression

The simplest form of regression is linear regression, which assumes that the predictors have a linear relationship with the target variable.

```
set.seed(42)
lm <- train(Salary ~ .,
            train_data, method = 'lm', trControl = custom_parameter)
```

```
lm
```

```
summary(lm)
```

```
Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
    Min      1Q  Median      3Q     Max
-2.4099 -0.5369 -0.0079  0.4397  3.2117

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.03621    0.05432  -0.667   0.5060
AtBat       -0.41111    0.26283  -1.564   0.1196
Hits         0.49656    0.31613   1.571   0.1181
HmRun        0.13448    0.16228   0.829   0.4084
Runs        -0.01137    0.21184  -0.054   0.9573
RBI         -0.03182    0.20039  -0.159   0.8740
Walks        0.23117    0.12327   1.875   0.0625 .
Years        0.27456    0.16167   1.698   0.0913 .
CAtBat      -0.20862    0.86025  -0.243   0.8087
CHits        0.71856    1.26764   0.567   0.5716
CHmRun       0.21699    0.36878   0.588   0.5570
CRuns        0.41479    0.64258   0.646   0.5195
CRBI        -0.60222    0.65464  -0.920   0.3589
CWalks      -0.27430    0.23675  -1.159   0.2482
PutOuts      0.10387    0.05897   1.762   0.0799 .
Assists      0.08659    0.08983   0.964   0.3364
Errors      -0.09590    0.08154  -1.176   0.2412
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7284 on 170 degrees of freedom
Multiple R-squared:  0.5443, Adjusted R-squared:  0.5015
F-statistic: 12.69 on 16 and 170 DF, p-value: < 2.2e-16
```

Observation

From the above output we can see that there are many variables are significantly related to variable salary. This data will be used as comparison with the regularized model data to check if the variables that are eliminated were statistically significant or not. The adjusted R squared value indicates this model is 50% significant.

```
options(repr.plot.width = 5, repr.plot.height = 5)
plot(lm$finalModel)
```

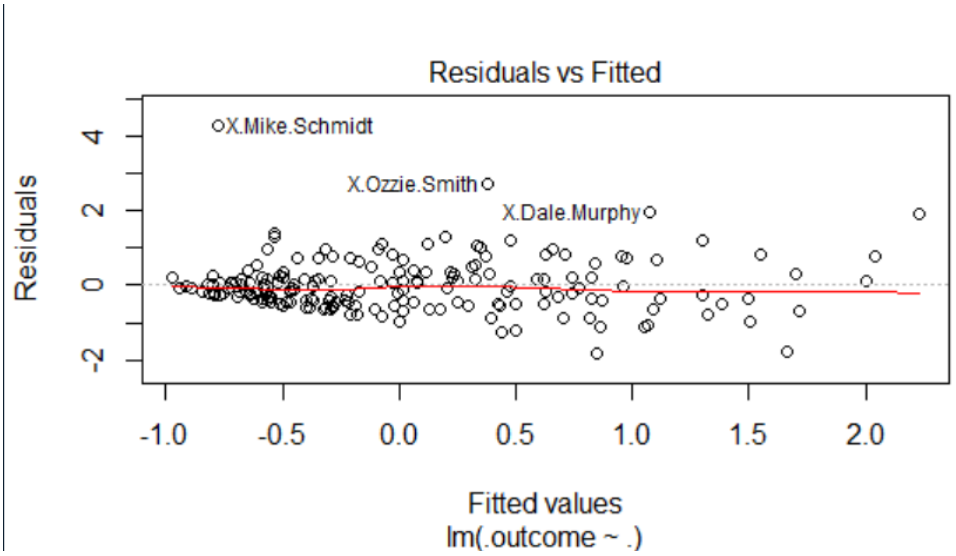

Figure 1.3 Residual vs fitted values plot

From the above figure we can say that there exist the heteroskedasticity which means indicates signs of non-linearity in the data which has not been captured by the model.

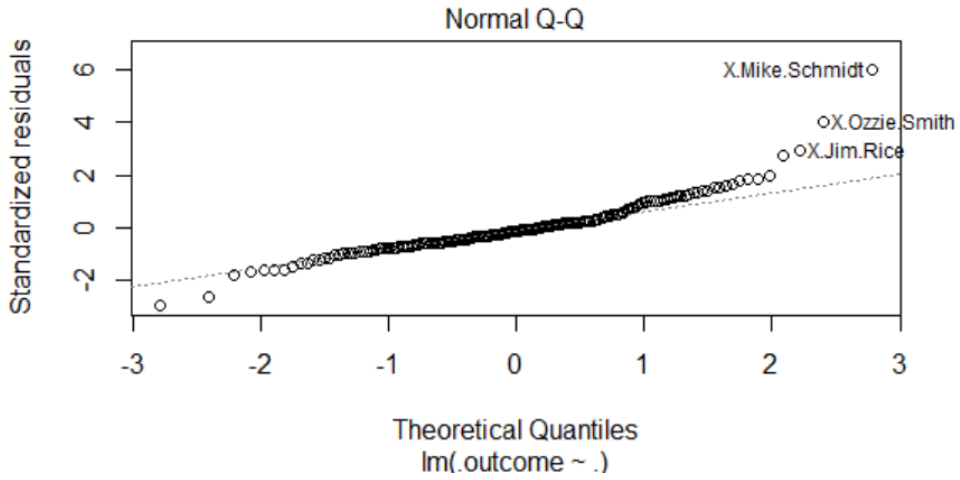

Figure 1.4 Standardizes residuals plot Vs Theoretical Quantities

When you use the standardized residuals, the expected value of the residuals is zero and the variance is one. From the graph the we can see that 95% of residuals to lie between -1.96 and 1.96. So we can say that theoretical residuals in a linear model are independent identically normally distributed

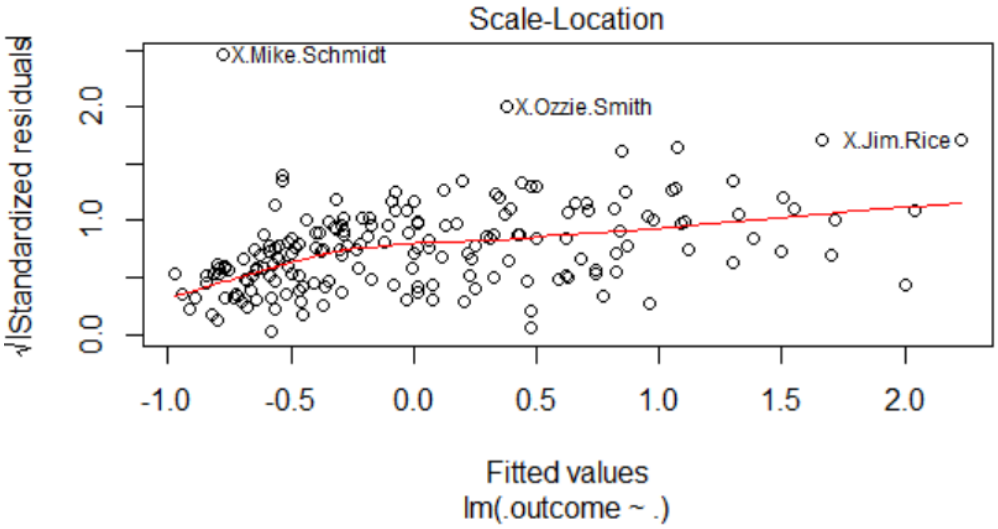

Figure 1.5 Standardizes residuals plot Vs Fitted values

The Graph of Standardizes residuals plot Vs Fitted values indicates that the the residuals begin to spread wider along the x-axis as it passes around 0.5. Because the residuals spread wider and wider, the red smooth line is not horizontal

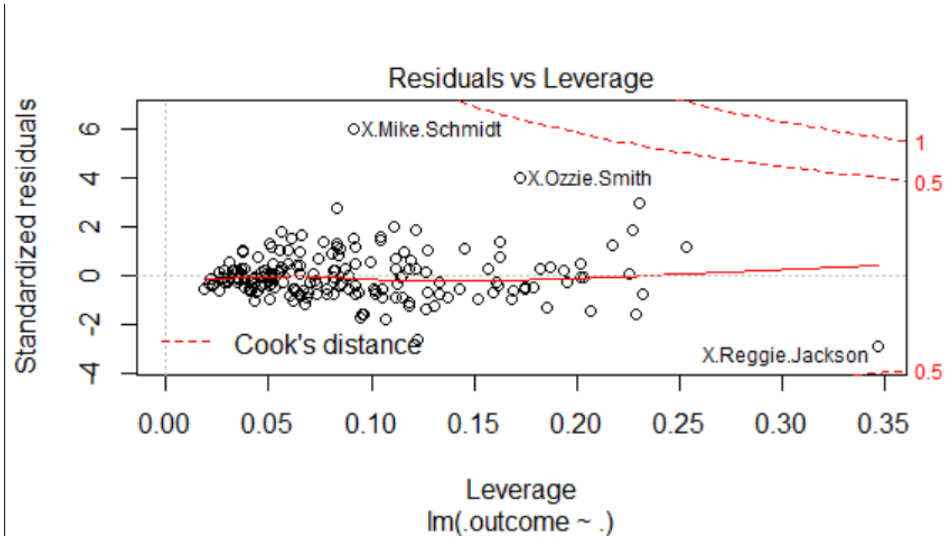

Figure 1.7 Standardizes residuals VS leverage

This graph helps to find the influential cases .Not all outliers are influential in linear regression analysis (whatever outliers mean). Even though data have extreme values, they might not be

influential to determine a regression line. That means, the results wouldn't be much different if we either include or exclude them from analysis.From figure we can see that we barely see Cook's distance lines. because all cases are well inside of the Cook's distance lines. This a typical look when there is no influential case, or cases

Ridge Regression

Code and Output:

```
set.seed(42)
ridge_reg <- train(Salary ~ ., train_data, method = 'glmnet',
                   tuneGrid = expand.grid(alpha = 0, lambda = seq(0.7,0.8,length = 5)),
                   trControl = custom_parameter)
```

```
options(repr.plot.width = 5, repr.plot.height = 4)
plot(ridge_reg)
```

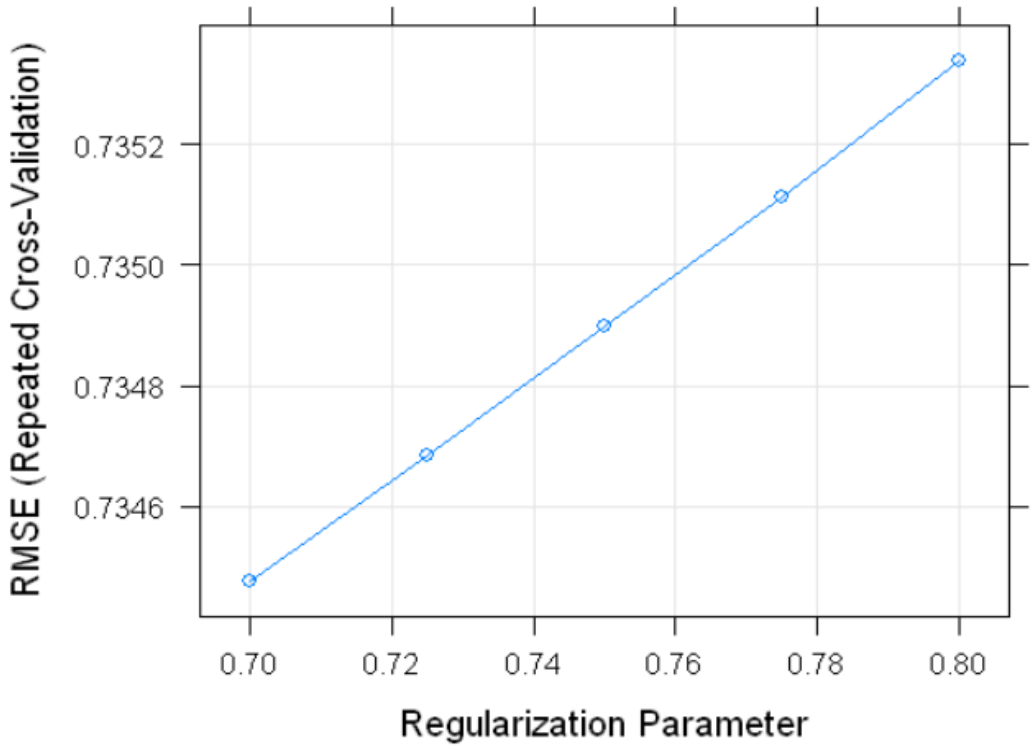

Fig 1.7 RMSE graph for regularization parameter with repeated cross validation

```
ridge_reg
```

```
glmnet

187 samples
 16 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 170, 167, 169, 168, 167, 168, ...
Resampling results across tuning parameters:

  lambda  RMSE       Rsquared   MAE
  0.700   0.7344775  0.5276390  0.6031542
  0.725   0.7346856  0.5275785  0.6034067
  0.750   0.7348993  0.5275172  0.6036587
  0.775   0.7351126  0.5274589  0.6039088
  0.800   0.7353367  0.5273940  0.6041614

Tuning parameter 'alpha' was held constant at a value of 0
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were alpha = 0 and lambda = 0.7.
```

```
ridge_reg$results
```

alpha	lambda	RMSE	Rsquared	MAE	RMSESD	RsquaredSD	MAESD
0	0.700	0.7344775	0.5276390	0.6031542	0.1356134	0.1808546	0.08282236
0	0.725	0.7346856	0.5275785	0.6034067	0.1350207	0.1807321	0.08257448
0	0.750	0.7348993	0.5275172	0.6036587	0.1344400	0.1806098	0.08233298
0	0.775	0.7351126	0.5274589	0.6039088	0.1338681	0.1804914	0.08209989
0	0.800	0.7353367	0.5273940	0.6041614	0.1333091	0.1803781	0.08187622

Tab 1.1 Ridge regression results on Hitters data

Observation

From the Fig 1.7 we can see that the Root mean squared error starts to grow with the increase in the value of regularization parameter. For lambda of 0.7 and alpha of 0 we have the lowest root mean squared error of 0.7345 and highest R^2 Squared value of 0.5276. The above table shows

the results of ridge regression for different values of lambda. The Table 1.1 shows the Root mean squared error, R squared, mean absolute error and standard deviation values for RMSE, R^2 and MAE.

```
options(repr.plot.width = 8, repr.plot.height = 5)
plot(ridge_reg$finalModel, xvar = 'lambda', label = T)
```

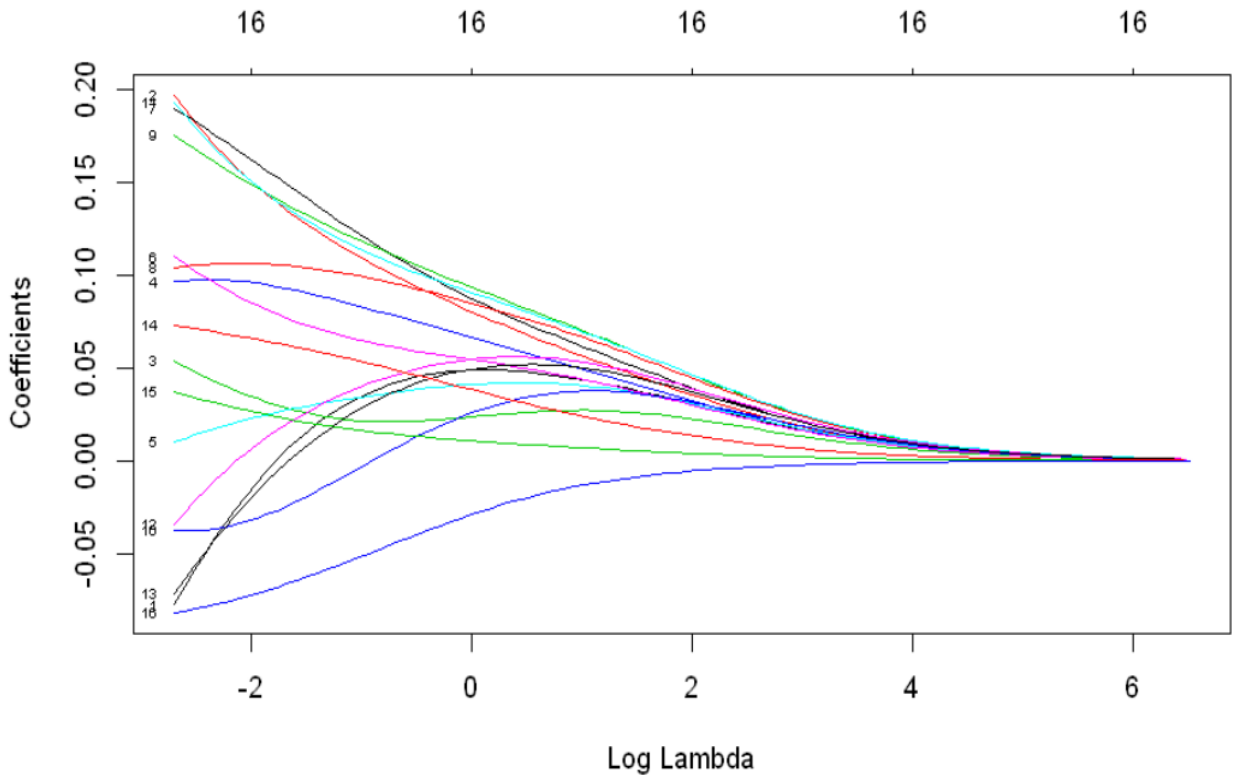

Fig 1.8 Coefficients graph for all the features for different lambda.

Observation

From the Fig 1.8 we can see that for increase in the value of lambda the coefficients for all the features approaches to zero and for the lower values of the lambda we can see some features in the data has larger coefficients that indicates those are importance variable that helps in predicting the salary. On the top of graph, we can see the value 16 which indicates the number of

variables that are contributing in the predicting for different values of lambda and in the case of ridge regression it makes the coefficients nearly zero but not exactly zero.

```
options(repr.plot.width = 8, repr.plot.height = 5)
plot(ridge_reg$finalModel, xvar = 'dev', label = T)
```

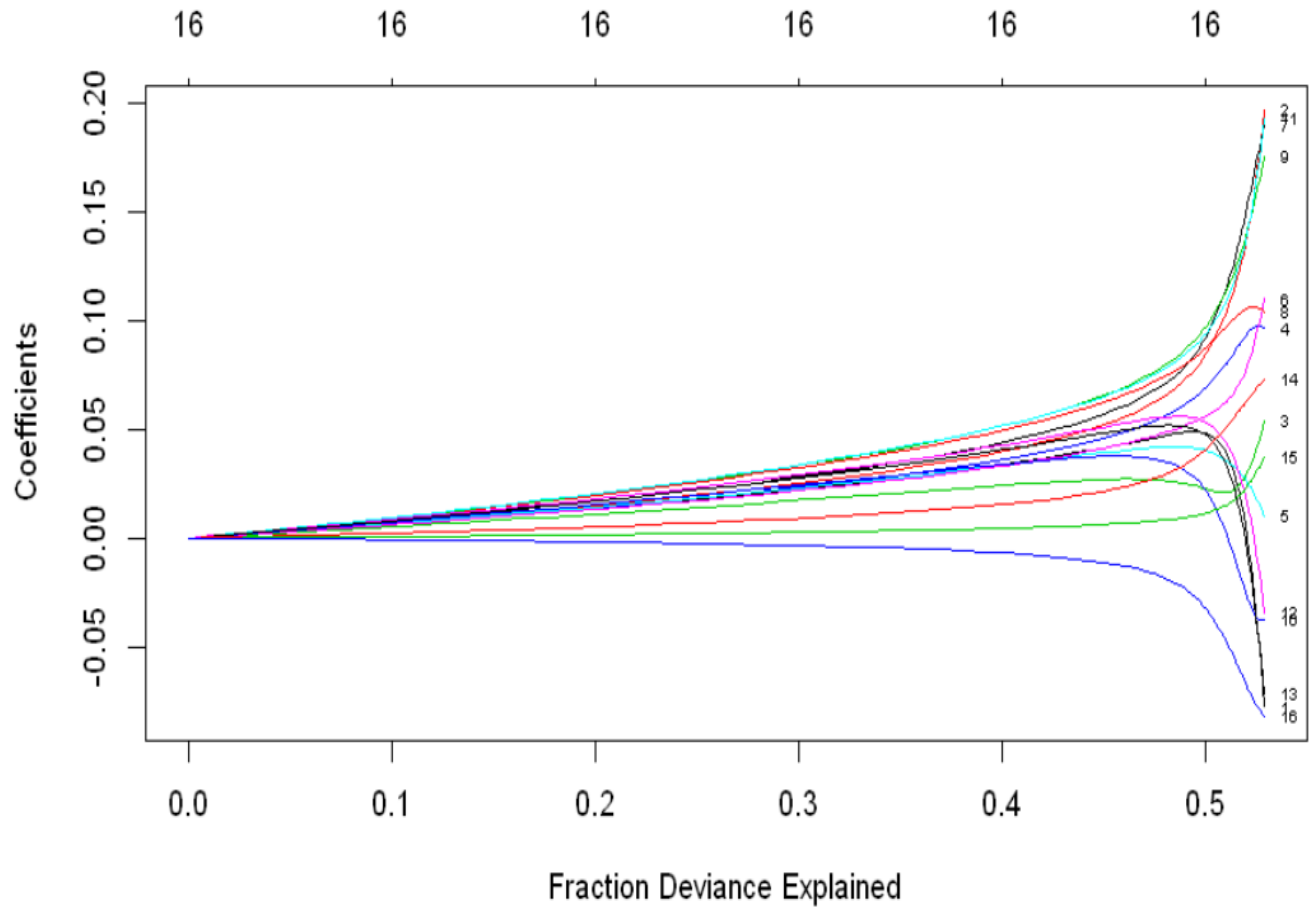

Fig 1.9 Coefficients graph for all the features for different fraction deviance.

Observation

From the Fig 1.9 we can see that for increase in the fraction deviance the coefficients for all the features increases till 0.5 and there is sudden increase in the value of coefficients indicates that the model starts to overfit the data. The 0.5 fraction deviance value indicates the 52.7% of

variation is explained by these features in the Hitters data and it also shows the accuracy of model in predicting the salary.

```
options(repr.plot.width = 7, repr.plot.height = 5)
plot(varImp(ridge_reg))
```

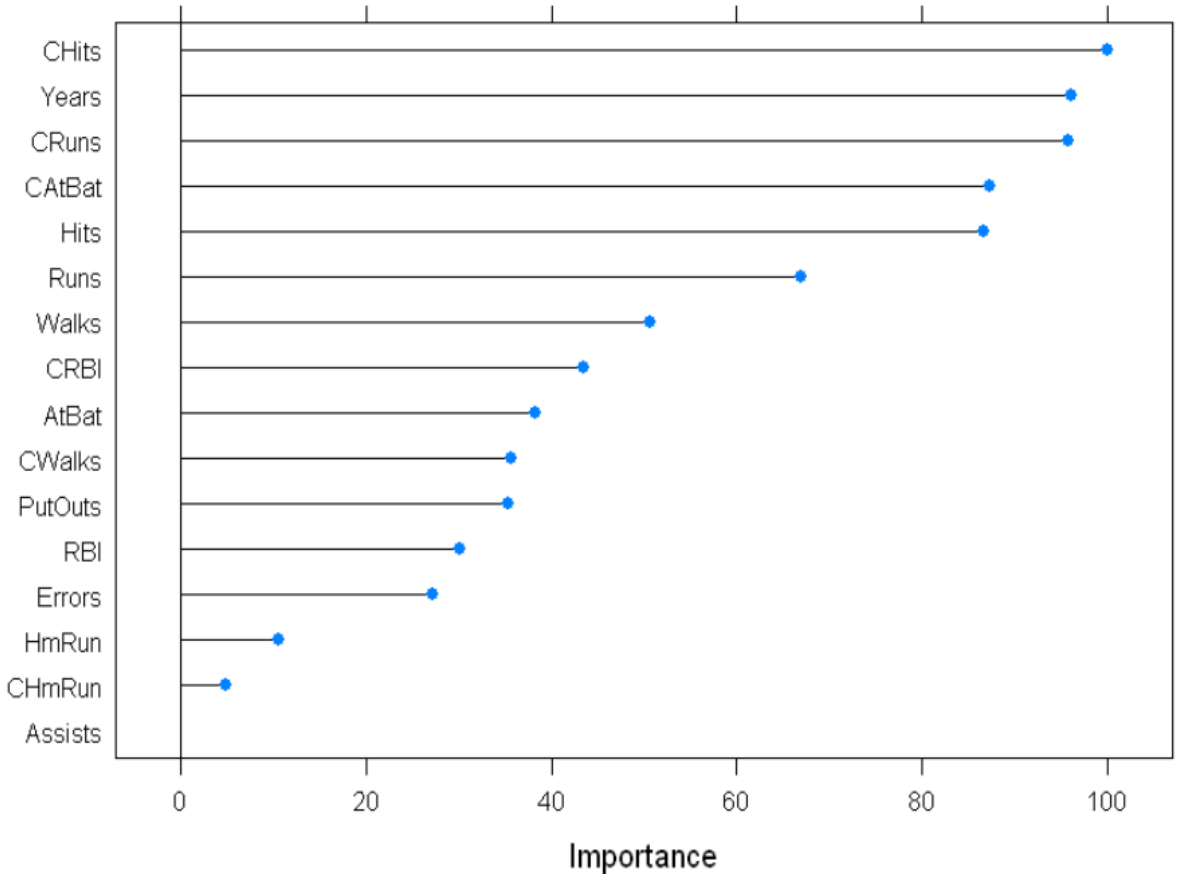

Fig 1.10 Variable Importance based on Ridge Regression

Observation

From the Fig 1.10 we can see the important variables that helps in predicting the salary of the player. **CHits** - Number of hits during his career, **Years** - Number of years in the major leagues and **Runs** - Number of runs in 1986 are top 3 features that has highest contribution for prediction.

Lasso Regression

Code and output:

```
set.seed(42)
lasso_reg <- train(Salary ~ ., train_data, method = 'glmnet',
                   tuneGrid = expand.grid(alpha = 1, lambda = seq(0.016,0.020, length = 10)),
                   trControl = custom_parameter)
```

```
lasso_reg
```

```
glmnet

187 samples
 16 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 170, 167, 169, 168, 167, 168, ...
Resampling results across tuning parameters:

  lambda       RMSE       Rsquared   MAE
  0.01600000   0.7329825  0.5280082  0.5973768
  0.01644444   0.7328430  0.5281511  0.5973205
  0.01688889   0.7327487  0.5282555  0.5973135
  0.01733333   0.7326976  0.5283055  0.5973410
  0.01777778   0.7326553  0.5283483  0.5973729
  0.01822222   0.7326334  0.5283666  0.5974285
  0.01866667   0.7326404  0.5283460  0.5975148
  0.01911111   0.7326571  0.5283125  0.5976113
  0.01955556   0.7326773  0.5282740  0.5977096
  0.02000000   0.7327073  0.5282216  0.5978123

Tuning parameter 'alpha' was held constant at a value of 1
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were alpha = 1 and lambda = 0.01822222.
```

```
options(repr.plot.width = 5, repr.plot.height = 3)
plot(lasso_reg)
```

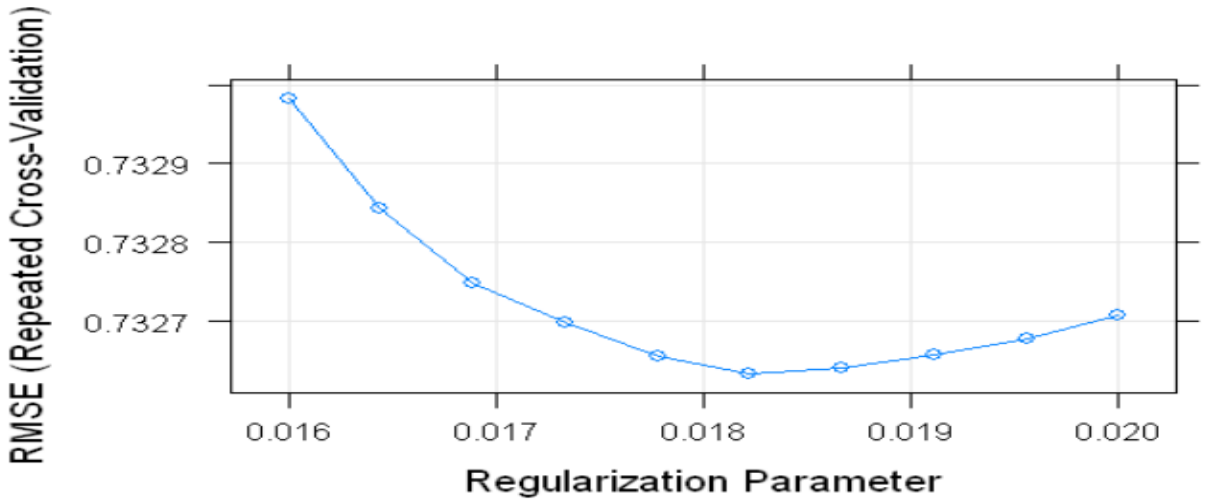

Fig 1.11 RMSE graph for regularization parameter with repeated cross validation

```
lasso_reg$results
```

alpha	lambda	RMSE	Rsquared	MAE	RMSESD	RsquaredSD	MAESD
1	0.01600000	0.7329825	0.5280082	0.5973768	0.1592608	0.1840529	0.09316926
1	0.01644444	0.7328430	0.5281511	0.5973205	0.1589905	0.1839311	0.09300552
1	0.01688889	0.7327487	0.5282555	0.5973135	0.1587399	0.1838222	0.09286401
1	0.01733333	0.7326976	0.5283055	0.5973410	0.1585110	0.1837569	0.09271871
1	0.01777778	0.7326553	0.5283483	0.5973729	0.1582883	0.1836858	0.09257477
1	0.01822222	0.7326334	0.5283666	0.5974285	0.1580613	0.1836013	0.09243601
1	0.01866667	0.7326404	0.5283460	0.5975148	0.1578490	0.1835376	0.09231560
1	0.01911111	0.7326571	0.5283125	0.5976113	0.1576450	0.1834914	0.09220890
1	0.01955556	0.7326773	0.5282740	0.5977096	0.1574412	0.1834434	0.09210378
1	0.02000000	0.7327073	0.5282216	0.5978123	0.1572344	0.1833889	0.09199907

Tab 1.2 Lasso regression results on Hitters data

Observation

From the Fig 1.11 we can see that the Root mean squared error starts to drop with the increase in the value of regularization parameter and starts to grow after the value of 0.0185. After tuning different ranges for lambda, we got optimum range 0.016 to 0.020. For lambda of 0.0182 and alpha of 1 we have the lowest root mean squared error of 0.73263 and highest R^2 Squared value of 0.52836. The Tab 1.2 shows the results of ridge regression for different values of lambda and shows the Root mean squared error, R squared, mean absolute error and standard deviation values for RMSE, R^2 and MAE.

```
options(repr.plot.width = 7, repr.plot.height = 5)
plot(lasso_reg$finalModel, xvar = 'lambda', label = T)
```

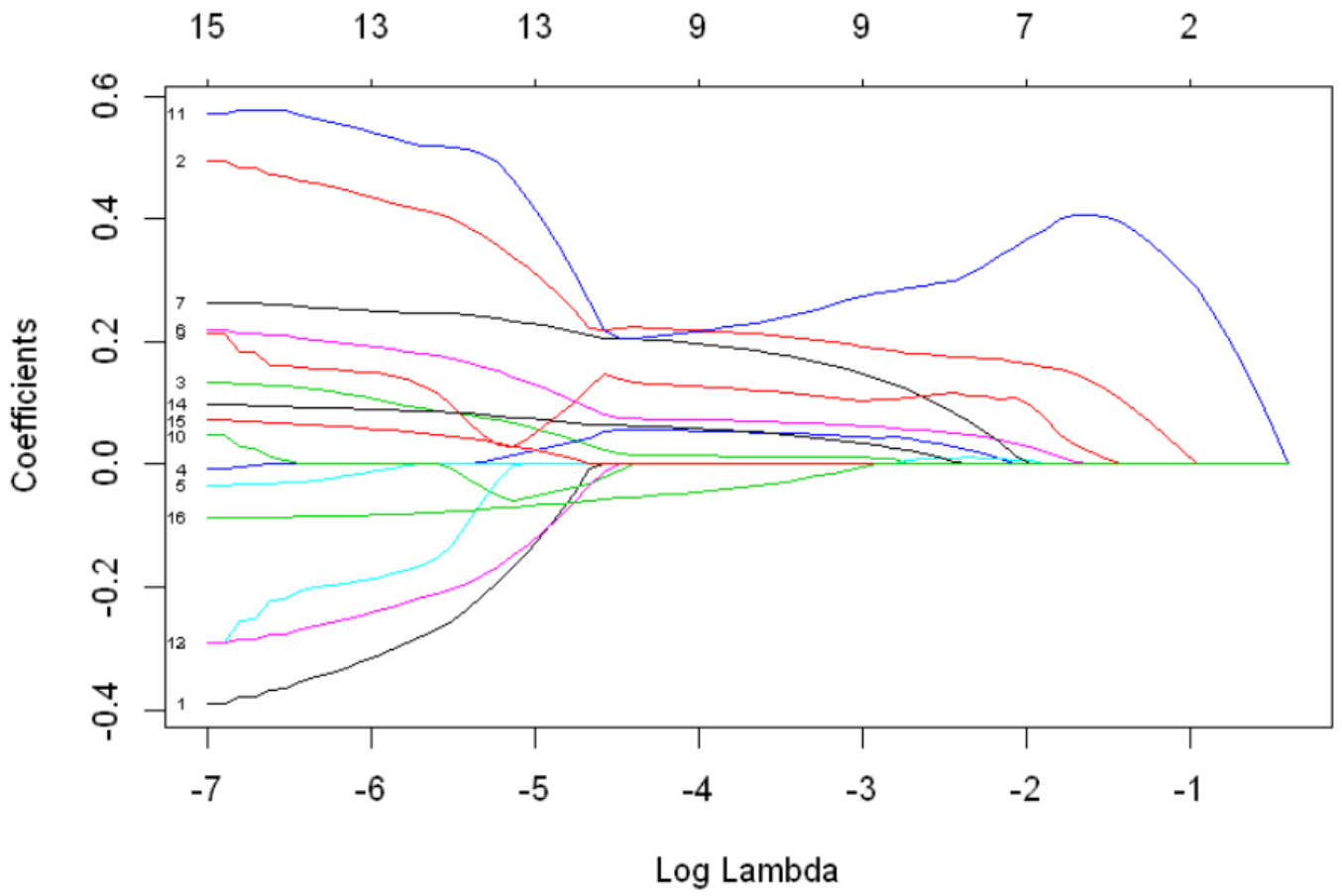

Fig 1.12 Coefficients graph for all the features for different lambda.

Observation

From the Fig 1.12 we can see that for increase in the value of lambda the coefficients for all the features reaches to zero for lasso regression which helps in removing the variables which has multi -collinearity. For lower values of the lambda we can see some features in the data has larger coefficients that indicates those are importance variable that helps in predicting the salary. On the top of graph, we can see the value 15 which indicates the number of variables that are

contributing in the predicting for different values of lambda and in the case of lasso regression it makes the useless features coefficients to zero.

```
options(repr.plot.width = 7, repr.plot.height = 5)
plot(lasso_reg$finalModel, xvar = 'dev', label = T)
```

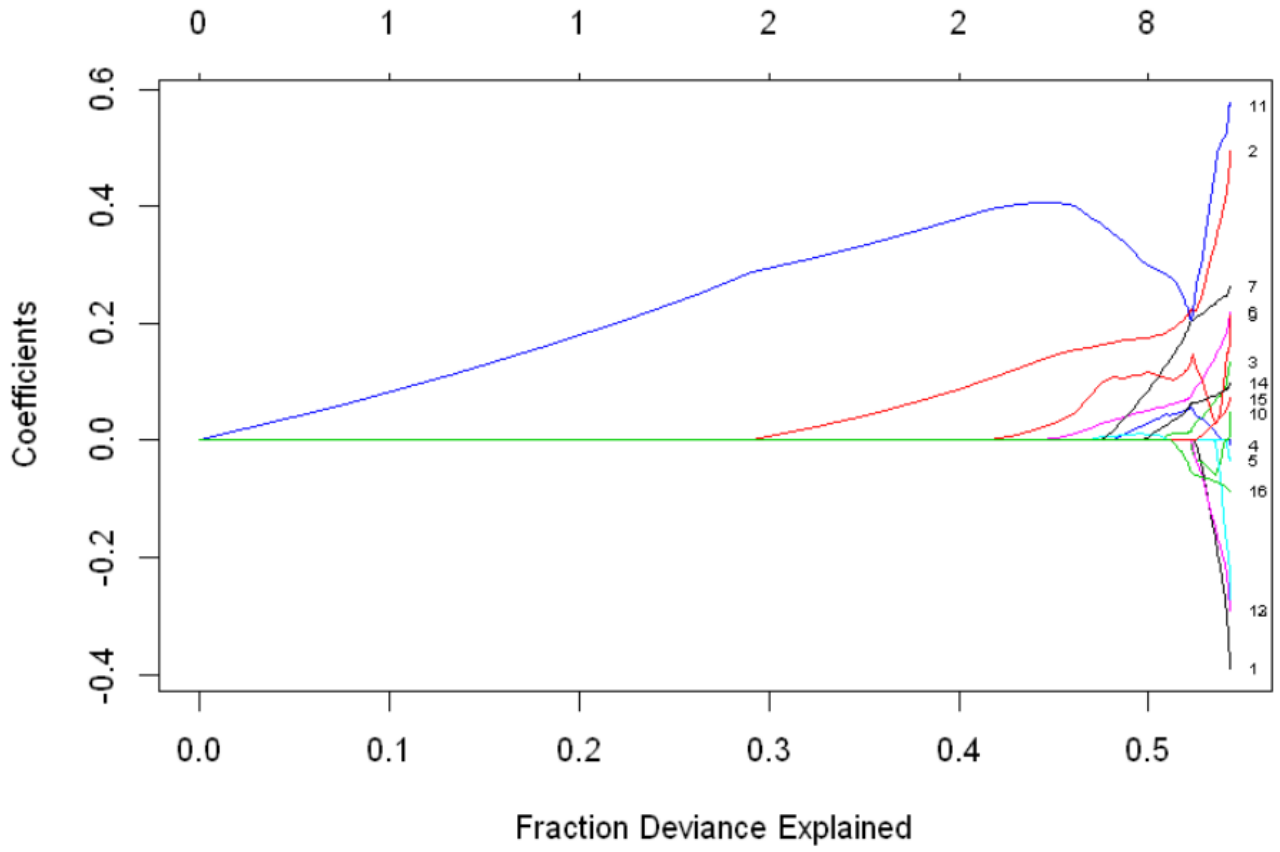

Fig 1.13 Coefficients graph for all the features for different fraction deviance.

Observation

From the Fig 1.13 we can see that for increase in the fraction deviance the coefficients for all the features increases till 0.5 and there is sudden increase in the value of coefficients indicates that the model starts to overfit the data. The 0.5 fraction deviance value indicates the 52.8% of

variation is explained by **eight** features in the Hitters data and other useless variables are not considered for prediction and it also shows the accuracy of model in predicting the salary.

```
options(repr.plot.width = 7, repr.plot.height = 4)
plot(varImp(lasso_reg))
```

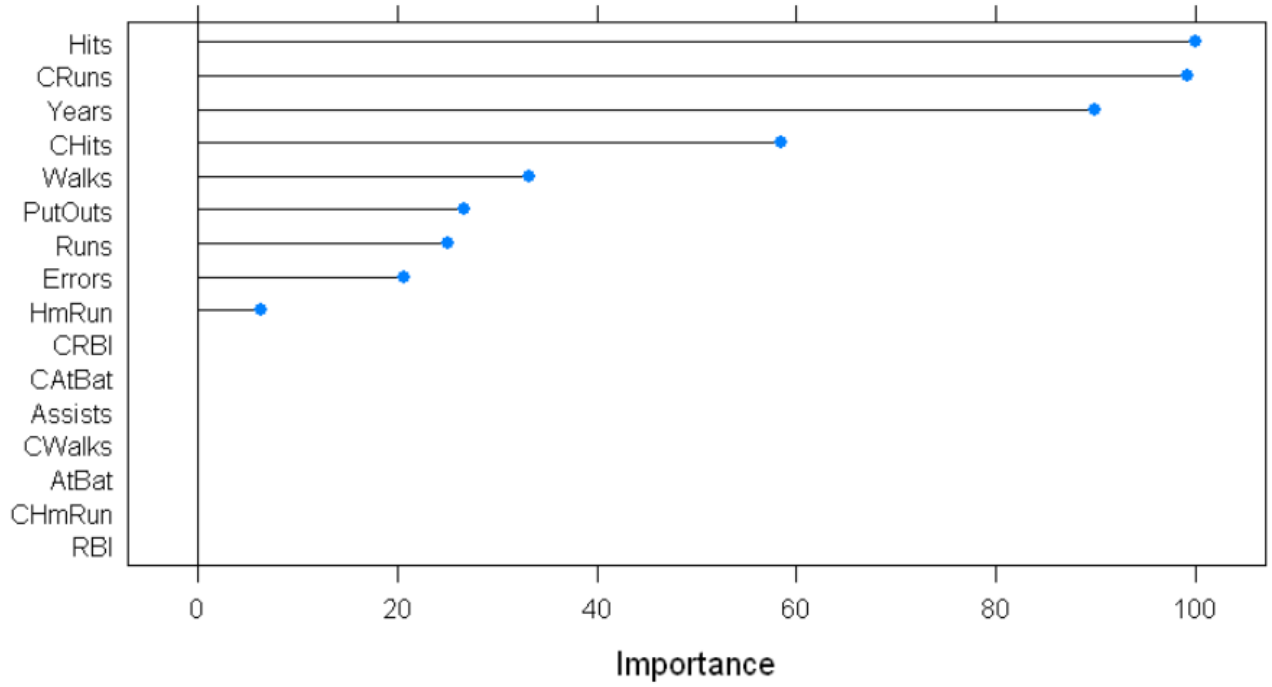

Fig 1.14 Variable Importance based on Lasso regression

From the Fig 1.14 we can see the important variables like Hits, CRuns, Years, CHits, Walks, Putouts, Runs, Errors and Hmrun that helps in predicting the salary of the player.

Elastic Net Regression

Code and output

```
set.seed(42)
enet_reg <- train(Salary ~ ., train_data, method = 'glmnet',
                  tuneGrid = expand.grid(alpha = seq(0,1,length = 10), lambda = seq(0.05,0.1, length = 10)),
                  trControl = custom_parameter)
```

```
Selecting tuning parameters
Fitting alpha = 0.222, lambda = 0.0778 on full training set
```

```
options(repr.plot.width = 10, repr.plot.height = 4)
plot(enet_reg)
```

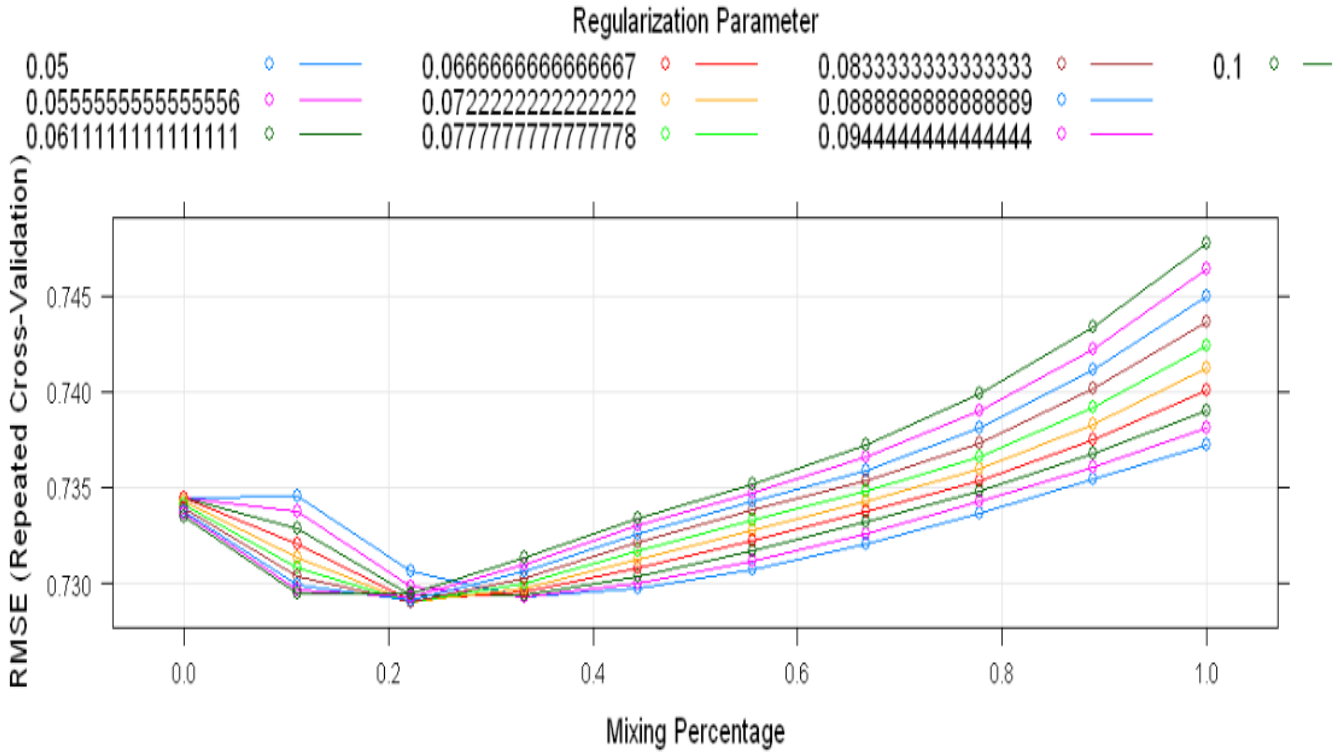

Fig 1.15 RMSE graph for regularization parameter with repeated cross validation

Observation

From the Fig 1.15 we can see that the Root mean squared error changes for different values of lambda and alpha. After tuning different ranges for lambda, we got optimum range 0.05 to 0.1. For lambda of 0.0778 and alpha of 0.222 we have the lowest root mean squared error of 0.7289 and highest R^2 Squared value of 0.5318.

```
options(repr.plot.width = 7, repr.plot.height = 5)
plot(enet_reg$finalModel, xvar = 'lambda', label = T)
```

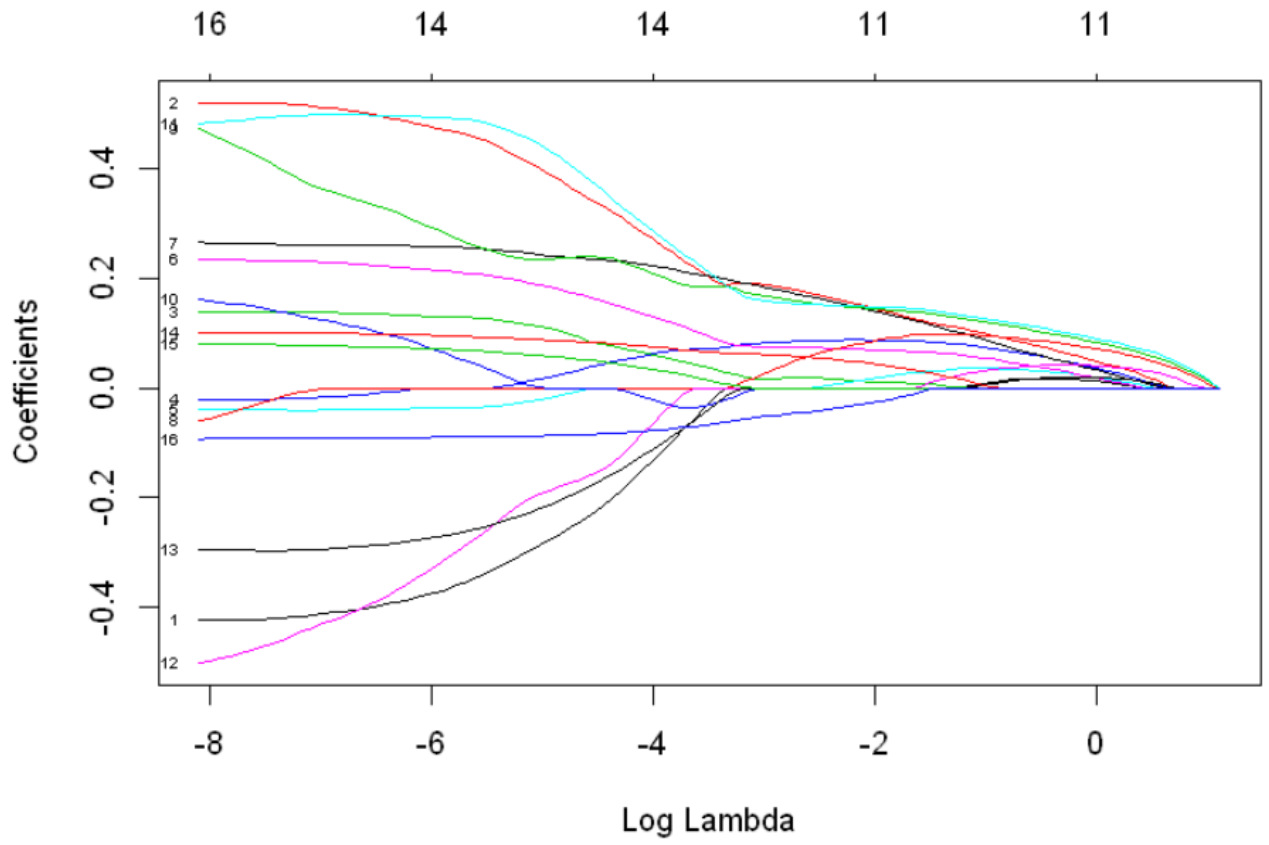

Fig 1.16 Coefficients graph for all the features for different lambda.

Observation

From the Fig 1.16 we can see that for increase in the value of lambda the coefficients for all the features reaches to zero for elastic net regression. For lower values of the lambda we can see some features in the data has larger coefficients that indicates those are importance variable that helps in predicting the salary. On the top of graph, we can see the value 11 which indicates the number of variables that are contributing in the predicting the salary.

```
options(repr.plot.width = 7, repr.plot.height = 5)
plot(enet_reg$finalModel, xvar = 'dev', label = T)
```

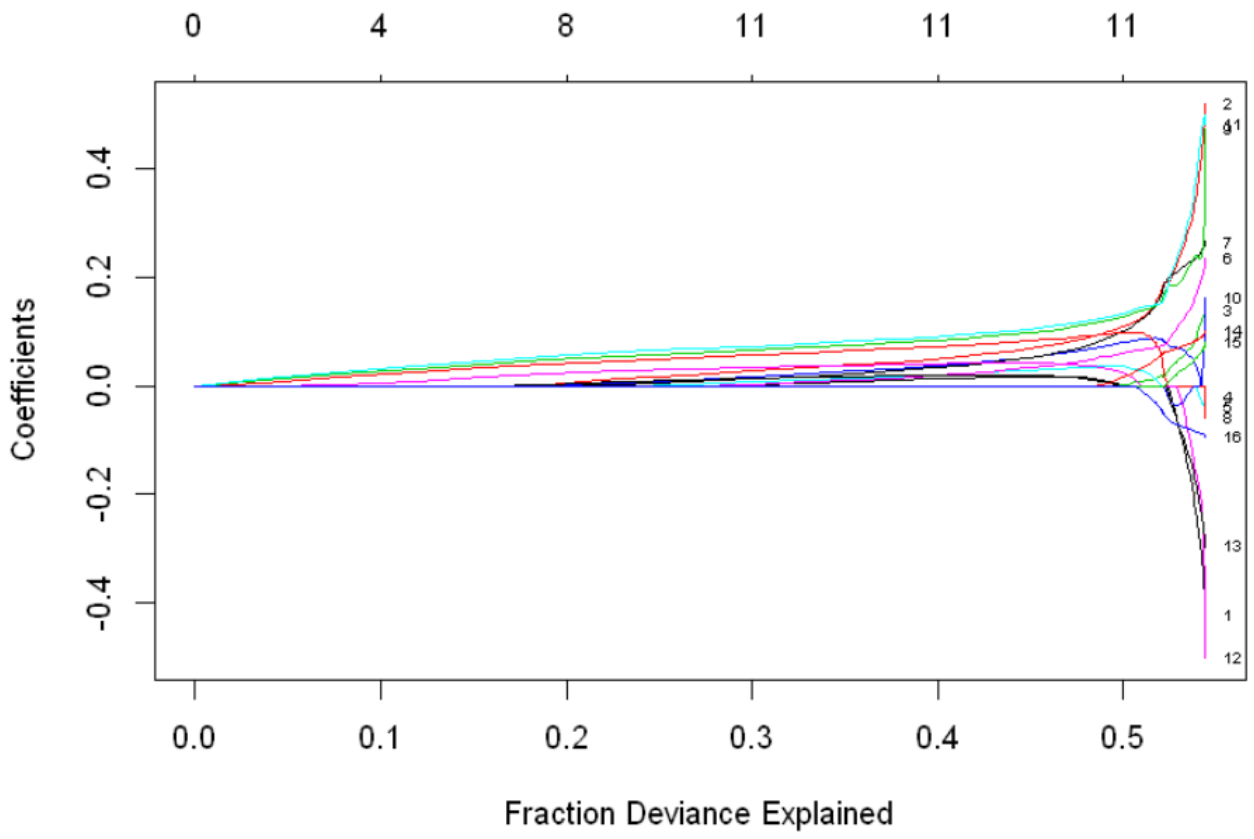

Fig 1.17 Coefficients graph for all the features for different fraction deviance.

Observation

From the Fig 1.17 we can see that for increase in the fraction deviance the coefficients for all the features increases till 0.5 and there is sudden increase in the value of coefficients indicates that the model starts to overfit the data. The 0.5 fraction deviance value indicates the 53% of variation is explained by **Eleven** features in the Hitters data and other useless variables are not considered for prediction and it also shows the accuracy of model in predicting the salary.

```
options(repr.plot.width = 7, repr.plot.height = 4)
plot(varImp(enet_reg))
```

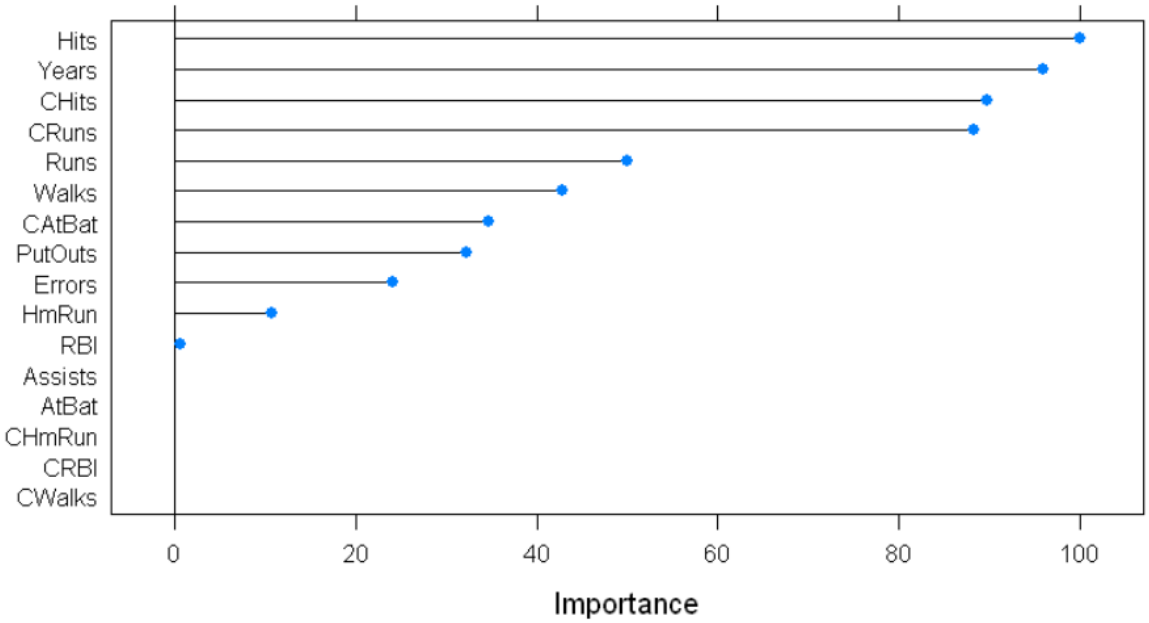

Fig 1.18 Variable Importance based on Elastic net Regression

Observation

From the Fig 1.18 we can see the important variables like Hits, CRuns, Years, CHits, Walks, Putouts, Runs, Errors, RBI, CAtBat and Hmrun that helps in predicting the salary of the player.

Conclusion

To analyse which model is the best for the dataset. we have performed model seclection which as follows .

```
models = list(LinearModel = lm, Ridge = ridge_reg, Lasso = lasso_reg, ElasticNet = enet_reg)
```

```
r <- resamples(models)
summary(r)
```

```
Call:
summary.resamples(object = r)

Models: LinearModel, Ridge, Lasso, ElasticNet
Number of resamples: 50

MAE
                Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
LinearModel 0.4715870 0.5439128 0.5984201 0.6139806 0.6447944 0.9573609    0
Ridge       0.4452550 0.5406212 0.6097848 0.6031542 0.6445205 0.8188163    0
Lasso       0.4378128 0.5341055 0.6029851 0.5974285 0.6433418 0.9041562    0
ElasticNet  0.4370602 0.5379024 0.6035362 0.5967160 0.6337799 0.8742625    0

RMSE
                Min.   1st Qu.    Median      Mean   3rd Qu.     Max. NA's
LinearModel 0.5636627 0.6275256 0.7024005 0.7510964 0.8366322 1.404070    0
Ridge       0.5404336 0.6440317 0.7166173 0.7344775 0.7773434 1.213015    0
Lasso       0.5167648 0.6276162 0.7039694 0.7326334 0.7848234 1.356663    0
ElasticNet  0.5180053 0.6322191 0.6963452 0.7289926 0.7793887 1.306211    0

Rsquared
                  Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
LinearModel 0.001078621 0.3741733 0.5441052 0.5061612 0.6438329 0.7899489    0
Ridge       0.019973661 0.4030044 0.5749445 0.5276390 0.6626265 0.8273094    0
Lasso       0.003203456 0.4031984 0.5783181 0.5283666 0.6506339 0.8100293    0
ElasticNet  0.007969084 0.4019734 0.5761262 0.5319031 0.6578558 0.8159962    0
```

To check the best model in Elastic Net

```
enet_reg$bestTune
```

	alpha	lambda
26	0.2222222	0.07777778

```
Best <- enet_reg$finalModel
```

```
coef(Best, s = enet_reg$bestTune$lambda)
```

```
17 x 1 sparse Matrix of class "dgCMatrix"
                      1
(Intercept) -0.024953306
AtBat        .
Hits         0.173056697
HmRun        0.018612597
Runs         0.086623726
RBI          0.000905772
Walks        0.074024382
Years        0.165988581
CAtBat       0.059986499
CHits        0.155277166
CHmRun       .
CRuns        0.152845266
CRBI         .
CWalks       .
PutOuts      0.055827476
Assists      .
Errors      -0.041706731
```

```
saveRDS(enet_reg, 'final_model53.rds')
```

```
fm <- readRDS("final_model53.rds")
```

```
# Prediction
p2 <- predict(fm, test_data)
```

```
cat('RMSE for test data is', sqrt(mean((test_data$Salary-p2)^2)))
```

```
RMSE for test data is 0.6604637
```

Reference

1. Machine Learning Explained: Regularization. (2020). Retrieved 26 January 2020, from https://www.r-bloggers.com/machine-learning-explained-regularization/

2. Trevor Hastie and Junyang Qian (2020).Glmnet Vignette. Retrieved 26 January 2020, from https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html

3. Cross-Validation Essentials in R - Articles - STHDA. (2020). Retrieved 26 January 2020, from http://www.sthda.com/english/articles/38-regression-model-validation/157-cross-validation-essentials-in-r/