lab_07

2023-11-05

Preliminaries

I've hidden the preliminaries as usual. You can see it in the `R` Markdown file.

```
##                    AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun
## -Andy Allanson      293   66     1   30  29    14     1    293    66      1
## -Alan Ashby         315   81     7   24  38    39    14   3449   835     69
## -Alvin Davis        479  130    18   66  72    76     3   1624   457     63
## -Andre Dawson       496  141    20   65  78    37    11   5628  1575    225
## -Andres Galarraga   321   87    10   39  42    30     2    396   101     12
## -Alfredo Griffin    594  169     4   74  51    35    11   4408  1133     19
##                    CRuns CRBI CWalks League Division PutOuts Assists Errors
## -Andy Allanson        30   29     14      A        E     446      33     20
## -Alan Ashby          321  414    375      N        W     632      43     10
## -Alvin Davis         224  266    263      A        W     880      82     14
## -Andre Dawson        828  838    354      N        E     200      11      3
## -Andres Galarraga     48   46     33      N        E     805      40      4
## -Alfredo Griffin     501  336    194      A        W     282     421     25
##                    Salary NewLeague
## -Andy Allanson         NA         A
## -Alan Ashby         475.0         N
## -Alvin Davis        480.0         A
## -Andre Dawson       500.0         N
## -Andres Galarraga    91.5         N
## -Alfredo Griffin    750.0         A
```

We have the first six rows of data above, and we already see an `NA` value in the first row. Looking at the whole data set, we see that all the `NA` values present are in the column `Salary`. This makes sense since maybe the players don't want to reveal their salaries/the salaries are not public knowledge.

Omitting the rows with missing data:

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

Splitting the data into training and test sets:

```
set.seed(333)
nrows <- nrow(hitters)
sample_rows <- sample(nrow(hitters), size=(0.3*nrows))
X <- model.matrix(Salary ~., data = hitters)
X <-X[,-1]
train_x <- X[-sample_rows, ]
test_x  <- X[sample_rows,]
train_y <- hitters[-sample_rows, c("Salary")]
test_y <- hitters[sample_rows, c("Salary")]
```

After visiting the peer tutor (Nihal) and consulting with some classmates, we realised that if we sampled the other way (i.e. choosing the size to be `0.7*nrows` and using the sampled rows as the training set instead of the test set), we do not get the expected result in the last question of this lab. (Basically, in the last question we ended up getting the result that the OLS model does second best, which isn't really what we wanted. If OLS is already second best then why did we go through all this trouble of learning lasso and ridge regressions?)

Anyway, we can just continue with the rest of the assignment.

Here, we perform the lasso regression:

```
fit.lasso <- glmnet(train_x, train_y, alpha=1, standardize=TRUE) #remember to standardize!!!
```

Here, we find the required values from Q2:

```
lambda_10 <- fit.lasso$lambda[10]
lambda_50 <- fit.lasso$lambda[50]
coef_10 <- coef(fit.lasso, s = lambda_10)
coef_50 <- coef(fit.lasso, s = lambda_50)
l1_10 <- sum(abs(coef_10[-1]))
l1_50 <- sum(abs(coef_50[-1]))
```

The sum of the coefficients for the 50th value of λ is larger. Since the 50th value of $\lambda = 2.998$, which is smaller than the 10th value of λ, the L1 norm can be larger, while still minimising the value of the objective function.

Optimal λ

In general, to find the optimal λ, we want to find the λ that would give us the lowest possible value of the objective function for all potential values of λ. We can use cross-validation to help us with this task. If we only trained our model against one training set, we risk overfitting (i.e. capturing too much of the noise from the training set). Using cross-validation allows us to train our model against different samples (even though they all come from the same training set), so that we can produce a more accurate model (I think it's by taking the average of the errors produced by each model? But I am slightly confused so I hope my understanding is ok). So, by performing a cross-validated regression, we can find the optimal λ which gives the minimum MSE.

Cross-Validation Time

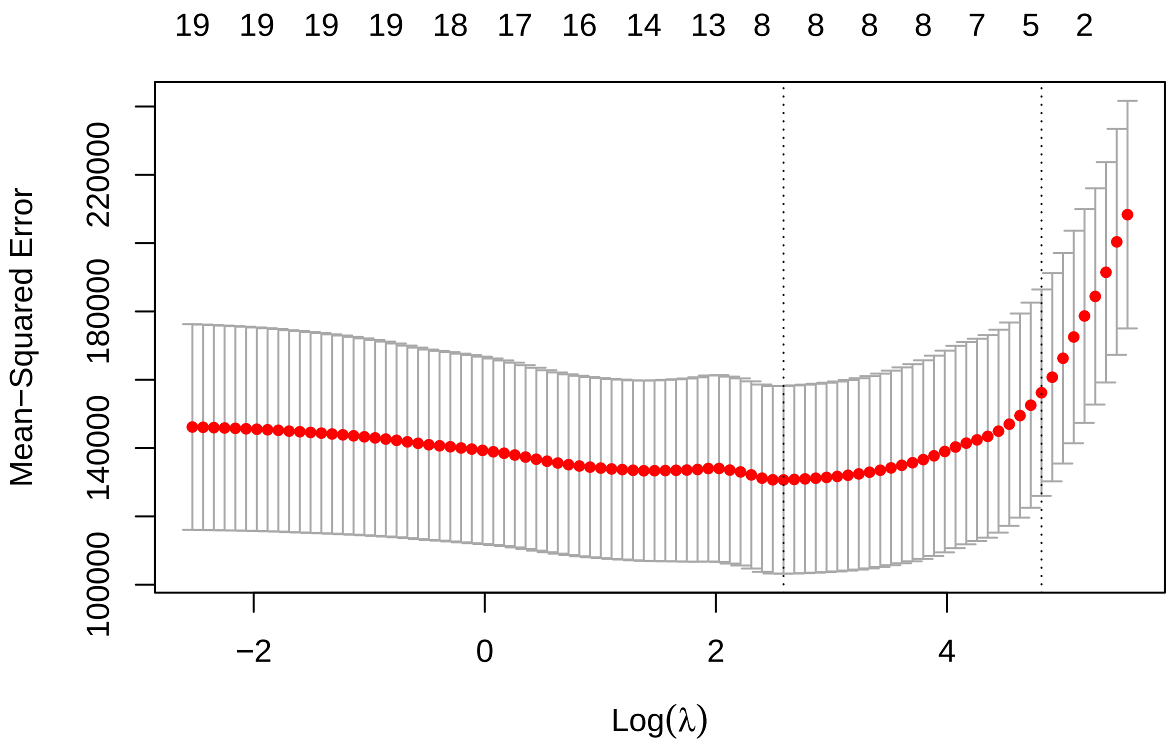
Here, we perform the required cross-validated lasso regression:

```
set.seed(1)
cv.lasso <- cv.glmnet(train_x, train_y,
                    alpha = 1,
                    nfolds=20,
                    standardize = TRUE)
best_lambda <- cv.lasso$lambda.min
lambda_1se <- cv.lasso$lambda.1se
```

```
## [1] "The best lambda is 13.2724232755777."
```

```
## [1] "The lambda within one standard error of the lowest MSE is 123.779063680546."
```

Here is the plot of the model performance vs the value of λ:



Basically, we want the λ that gives us the global minimum in the plot above, which is the λ that we stated above.

Here, we check the coefficients for each of the λs as requested:

```r
cv_coef_best <- coef(cv.lasso, s = best_lambda)
cv_coef_1se <- coef(cv.lasso, s = lambda_1se)
```

Now, looking at the coefficients:

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept)   27.6486139
## AtBat          .
## Hits           1.1882911
## HmRun          .
## Runs           1.2591490
## RBI            0.6056960
## Walks          1.4834147
## Years          .
## CAtBat         .
## CHits          0.2811829
## CHmRun         .
## CRuns          .
## CRBI           0.0756584
## CWalks         .
## LeagueN        .
## DivisionW   -116.5455275
## PutOuts        0.1672320
## Assists        .
## Errors         .
## NewLeagueN     .
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept) 300.4982347
## AtBat         .
## Hits          0.2177353
## HmRun         .
## Runs          1.1310268
## RBI           0.4084429
## Walks         .
## Years         .
## CAtBat        .
## CHits         0.1884307
## CHmRun        .
## CRuns         .
## CRBI          .
## CWalks        .
## LeagueN       .
## DivisionW     .
## PutOuts       .
## Assists       .
## Errors        .
## NewLeagueN    .
```

As we can see, there are coefficients which are `0`. The variables which are selected would be the ones that are not `0`.

The variables which are selected for the best λ are "Hits", "Runs", "RBI", "Walks", "CHits", "CRBI", "DivisionW", "PutOuts".

The variables which are selected for the λ within one standard error of the lowest MSE are "Hits", "Runs", "RBI", "CHits".

Now, we do the ridge regression:

```
cv.ridge <- cv.glmnet(train_x, train_y,
                      alpha = 0,
                      nfolds=20,
                      standardize = TRUE)
cv_ridge_lambda_1se <- cv.ridge$lambda.1se
cv_coef_ridge_1se <- coef(cv.ridge, s = cv_ridge_lambda_1se)
```

Looking at the list of coefficients,

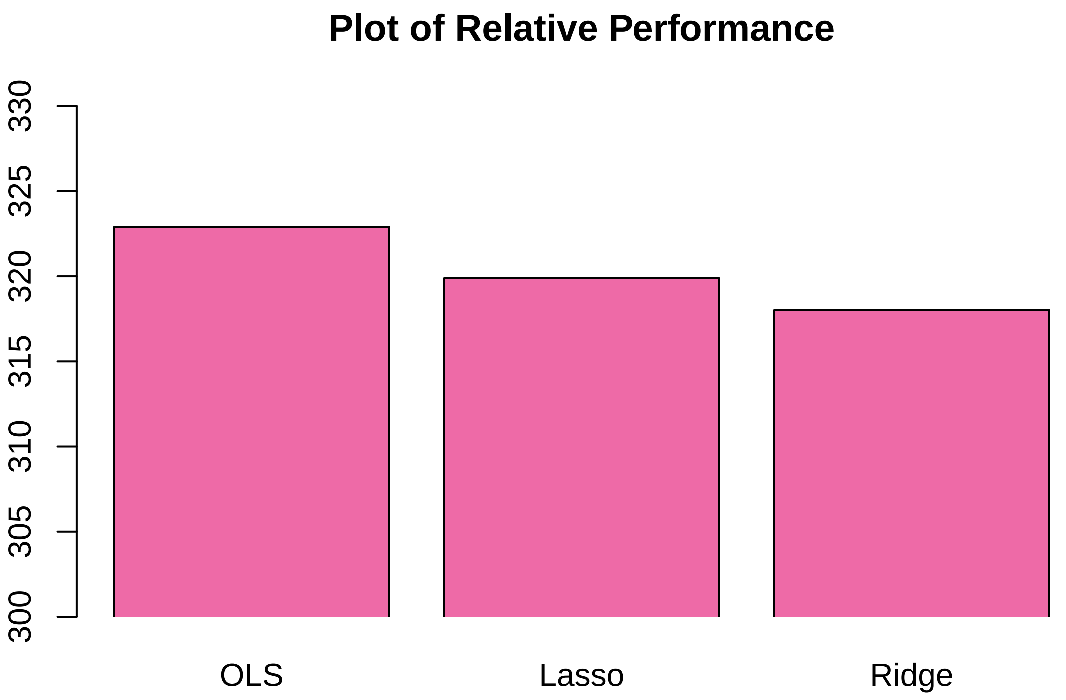
```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept) 308.193019743
## AtBat         0.068377092
## Hits          0.262852631
## HmRun         0.769737850
## Runs          0.461847635
## RBI           0.440885652
## Walks         0.517496605
## Years         2.356156656
## CAtBat        0.006012658
## CHits         0.022637904
## CHmRun        0.128577432
## CRuns         0.042919497
## CRBI          0.043055336
## CWalks        0.042082562
## LeagueN      -1.611722217
## DivisionW   -12.907797500
## PutOuts       0.027748668
## Assists       0.010075513
## Errors        0.003394876
## NewLeagueN    0.010305857
```

none of the coefficients are exactly zero.

Here, we do the OLS fitting, as well as evaluate the performances of all of our models:

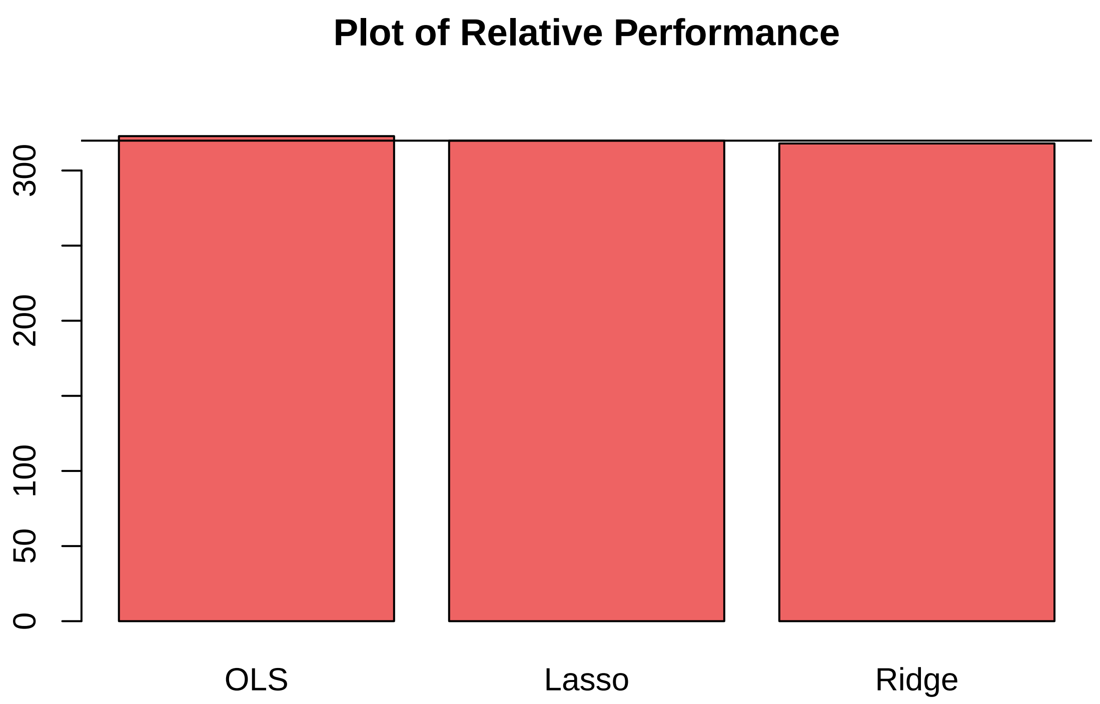
```r
lm_test <- hitters[sample_rows, ]
fit.lm <- lm(Salary ~ . , data = hitters[-sample_rows,])
predict_lm_y <- predict(fit.lm, new = hitters[sample_rows,])
rmse_lm <- sqrt(mean((predict_lm_y - test_y)^2))
predict_lasso_y <- predict(cv.lasso, newx = test_x, s = "lambda.min")
rmse_lasso <- sqrt(mean((predict_lasso_y - test_y)^2))
predict_ridge_y <- predict(cv.ridge, newx = test_x, s = "lambda.min")
rmse_ridge <- sqrt(mean((predict_ridge_y - test_y)^2))
```

Now, our barplot:



As expected, the linear model has the highest value of RMSE. OLS tends to overfit the data and not performing well. Ridge and lasso have better performances, as they are used for avoiding overfitting, which helps them to make more accurate predictions.

Below, I included another plot which doesn't cut off the bottom of the barplots, as I'm not sure if the above one is technically considered visual manipulation. I just wanted the differences to show up more clearly, but maybe the plot below is a better way of doing it:



Or maybe not, since you can barely see that the ridge barplot ends below the line.