NBA Basketball and Rookies – Regularised Regression

Valérie BLANCH
2614867B

1. INTRODUCTION

The dataset analysed here contains statistics on 600 basketball players during their rookie year at the NBA: the question of interest that justifies this project is whether it is possible to predict the career length of the players from their first year's performance, and which variable, or combination of variables will be the most useful during this analysis.

The dataset is structured as follows:

```
> str(basket)
'data.frame':   600 obs. of  23 variables:
 $ Name        : chr  "Mustafa Shakur" "Alec Burks" "Charles Thomas" "J.R. Henderson"
 ...
 $ Year_drafted: int  2010 2011 1991 1998 1991 1987 1989 2000 2009 1989 ...
 $ GP          : int  22 59 36 30 51 40 50 50 25 55 ...
 $ MIN         : num  7.2 15.9 4.3 11 20.7 9.5 9.8 18.7 5.2 18 ...
 $ PTS         : num  2.3 7.2 1.3 3.2 8.4 3.8 5.2 6.3 2.3 6.1 ...
 $ FG_made     : num  1 2.6 0.5 1.2 3.2 1.6 1.9 2.3 0.8 2.5 ...
 $ FGA         : num  2.7 6.1 1.4 3.2 6.4 3.7 4.1 6.2 2.1 5.2 ...
 $ FG_percent  : num  35.6 42.9 35.3 36.5 49.7 42.2 47.3 37 39.6 47.9 ...
 $ TP_made     : num  0 0.3 0.1 0.1 0 0.1 0.1 0.6 0.2 0 ...
 $ TPA         : num  0.5 0.8 0.5 0.2 0 0.3 0.3 1.7 0.6 0 ...
 $ TP_percent  : num  10 33.3 11.8 40 0 20 35.7 33.3 31.3 0 ...
 $ FT_made     : num  0.4 1.8 0.3 0.8 2 0.7 1.2 1.1 0.4 1.2 ...
 $ FTA         : num  0.7 2.4 0.4 1.5 2.9 0.7 1.8 1.4 0.5 1.7 ...
 $ FT_percent  : num  53.3 72.7 66.7 55.6 66.7 89.7 67 77.8 84.6 66.7 ...
 $ OREB        : num  0.3 1 0.2 0.7 2.3 0.1 0.6 0.4 0.3 1.5 ...
 $ DREB        : num  0.7 1.3 0.4 0.9 3.1 0.6 0.9 1.2 0.4 1.9 ...
 $ REB         : num  1 2.2 0.6 1.6 5.4 0.7 1.5 1.6 0.7 3.4 ...
 $ AST         : num  1.1 0.9 0.6 0.7 0.8 1.5 0.9 2.9 0.5 1 ...
 $ STL         : num  0.2 0.5 0.1 0.3 0.5 0.1 0.6 1 0.3 0.4 ...
 $ BLK         : num  0.1 0.1 0 0.1 0.5 0.1 0 0 0.1 0.2 ...
 $ TOV         : num  0.8 0.9 0.5 0.6 1.5 0.6 0.8 1.1 0.6 1.1 ...
 $ Yrs         : int  2 5 1 1 5 2 3 1 4 3 ...
 $ Target      : int  0 0 0 0 0 0 0 0 0 0 ...
```

- 600 rows corresponding to the 600 players.
- The last column of the data frame is the target variable, which is a binary outcome: "0" if the player's career length was less than or equal to 5 years, "1" otherwise.
- 22 variables containing various statistics for each player, which will be the predictors of the models:
    - **Categorical variables**: *Name* (the name of the player), *Year_drafted* (the year the player was drafted).
    - **Numeric variables**: *GP* (Games Played - out of 82), *PTS* (Points per game), *FG_made* (Field goals made - per game), *FGA* (Field goal attempts - per game), *TP_made* (Three points made - per game), *TPA* (Three point attempts - per game), *FT_made* (Free throws made - per game), *FTA* (Free throws attempts - per game), *OREB* (Offensive rebounds - per game), *DREB* (Defensive rebounds - per game), *REB* (Total rebounds - per game), *AST* (Assists - per game), *STL* (Steals - per game), *BLK* (Blocks - per game), *TOV* (Turnovers - per game), *MIN* (Minutes per game - out of 48), *FG_percent* (Field goal percentage), *TP_percent* (Three point percentage), *FT_percent* (Free throws percentage), *Yrs* (Career length - in years).

2. **EXPLORATORY ANALYSIS**

Pre-processing:

- Both the columns *Name* and *Yrs* have been dropped: the former because this variable is not relevant for this regression analysis, and the latter because it duplicates the target variable and thus produces perfect separation.
- The variables *Target* and *Year_drafted* have been converted to factors.
- The dataset has been tested for missing values, but there were none.

```
> anyNA(basket)
[1] FALSE
```

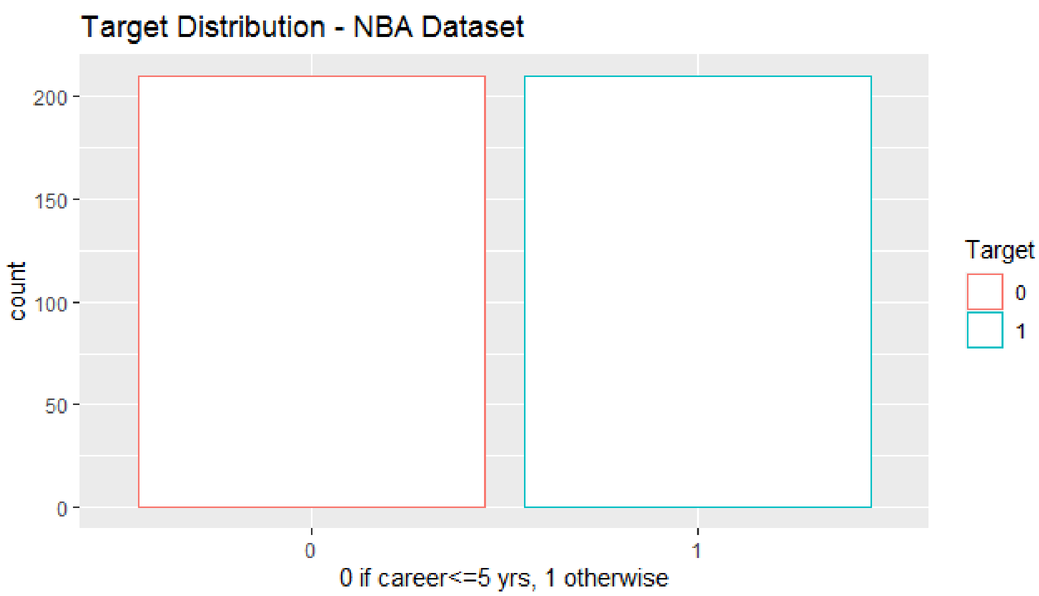
- The dataset has been separated into a training set (70%) and a test set (30%) using the createDataPartition() function from the *caret* package: that way both categories of the target variable are evenly distributed in each set. The training set contains 21 variables and 420 observations, while the test set contains 21 variables for 180 observations.

```
> dim(train.set)
[1] 420  21

> dim(test.set)
[1] 180  21
```

- The values have been scaled since they were recorded in different units.

EDA of the training set:

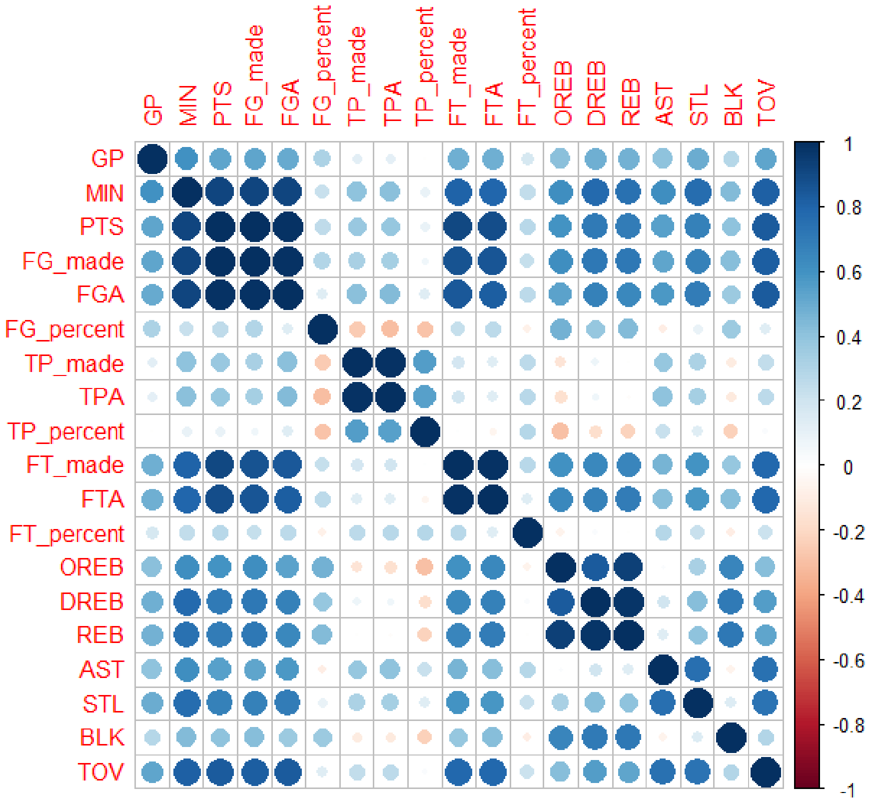
As mentioned above, the two categories of the target variable are evenly distributed in the training set.



210 players have a career less than or equal to 5 years and 210 have a career greater than 5 years, as shown in this proportion table:

```
> table(train.set$Target)

  0   1
210 210
```

After calculating the correlation between the numeric variables, a plot has been drawn and shows obvious evidence of multicollinearity.



The multicollinearity between the variables of the training set justifies the use of regularised logistic models. Three different techniques have been used:

- Lasso regression, which sets the irrelevant or correlated variables to zero, thus provides a strict variable selection and highlights which are the most influential.
- Ridge regression, which weights down irrelevant or correlated variables, thus keeps all of the variables and avoids any potential loss of information.
- Elastic net regression, which incorporates both algorithms.

3. <u>RESULTS</u>

All of the models in this study have been fitted using the *glmnetUtils* package, which provides the same functions as the *glmnet* one, plus the cva.glmnet() function which calculates the optimal alpha parameter for elastic net models.

Lasso Regression:

First, a lasso model has been computed using the following function and arguments:

```
> cv.lasso <- cv.glmnet(x, y, alpha=1, type.measure='class',
+                        family='binomial')
```

The cv.glmnet() function computes a 10-folds cross-validation by default. The alpha parameter has been set to 1 for lasso regression, the *family* option has been set to 'binomial' for the algorithm to recognize the target as binary. The *type.measure* argument is set to 'class': the function will find the best lambda (shrinkage parameter) according to the lowest misclassification error.

```
Measure: Misclassification Error

     Lambda Index Measure     SE Nonzero
min 0.03587    19  0.3095 0.0207       8
1se 0.13196     5  0.3238 0.0249       3
```

The function returns two lambda parameters: *lambda.min*, which is the exact value of lambda, and *lambda.1se*, which simplifies the model further within 1 standard error. Thus, the model using *lambda.min* has a lower misclassification error rate of 0.31 and 8 non-zero variables, and the one using *lambda.1se* has a higher misclassification error rate but has kept only 3 variables, *GP, MIN* and *REB*.

```
GP                0.1283905650
MIN               0.0140638773
PTS               .
FG_made           .
FGA               .
FG_percent        .
TP_made           .
TPA               .
TP_percent        .
FT_made           .
FTA               .
FT_percent        .
OREB              .
DREB              .
REB               0.1715358220
```

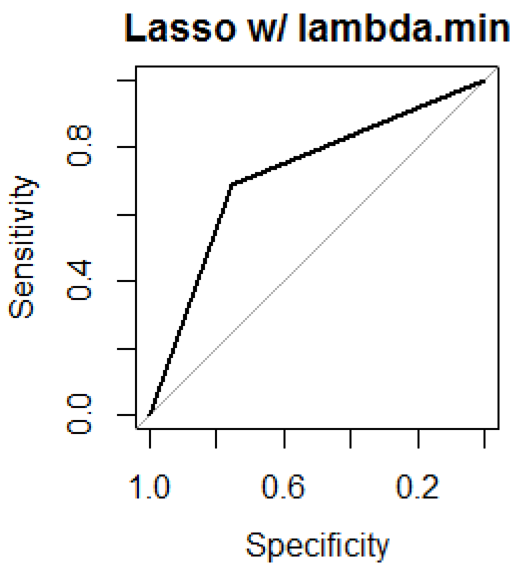
Figure 1 : coef(cv.lasso) output (cropped)

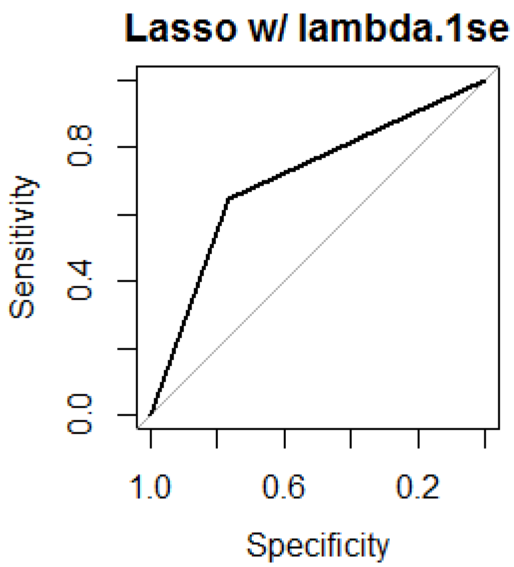
Predictions:

```
> #Lasso Model with lambda.min
> confusionMatrix(as.factor(pred.lasso.min)
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 68 28
         1 22 62

               Accuracy : 0.7222
                 95% CI : (0.6507, 0.7863)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 1.067e-09

                  Kappa : 0.4444

 Mcnemar's Test P-Value : 0.4795

            Sensitivity : 0.6889
            Specificity : 0.7556
         Pos Pred Value : 0.7381
         Neg Pred Value : 0.7083
             Prevalence : 0.5000
         Detection Rate : 0.3444
   Detection Prevalence : 0.4667
      Balanced Accuracy : 0.7222

       'Positive' Class : 1
```

```
> #Lasso Model with lambda.1se
> confusionMatrix(as.factor(pred.lasso.1se)
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 69 32
         1 21 58

               Accuracy : 0.7056
                 95% CI : (0.6332, 0.771)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 1.716e-08

                  Kappa : 0.4111

 Mcnemar's Test P-Value : 0.1696

            Sensitivity : 0.6444
            Specificity : 0.7667
         Pos Pred Value : 0.7342
         Neg Pred Value : 0.6832
             Prevalence : 0.5000
         Detection Rate : 0.3222
   Detection Prevalence : 0.4389
      Balanced Accuracy : 0.7056

       'Positive' Class : 1
```

Predictions have been computed using the predict() function, the test set, and both *lambda.min* and *lambda.1se.* The outputs of the confusionMatrix() function (*caret* package) are shown above. The overall accuracy of the lasso model with shrinkage parameter *lambda.min* is greater than the one with *lambda.1se* (0.72 vs 0.71), although the difference is not significant. Both models have a specificity greater than the sensitivity, meaning they both have trouble detecting positive outcomes (false negatives or type II errors). This slight imbalance between type I and type II errors can be seen with ROC curves (*pROC* package):





Therefore, considering the outputs above, the lasso model with *lambda.min* as a shrinkage parameter is the most performant one, although *lambda.1se* could be used as well, since its metrics are not much different, and its model's simplicity might be preferable for inference.

Ridge Regression:

Two ridge regression models have been fitted, using the same function and same arguments, apart from alpha, which has been set to 0. The cv.glmnet() function has calculated the following lambda parameters:

```
Measure: Misclassification Error

    Lambda Index Measure      SE Nonzero
min  2.201    49  0.3167 0.02536      51
1se 14.150    29  0.3310 0.02805      51
```

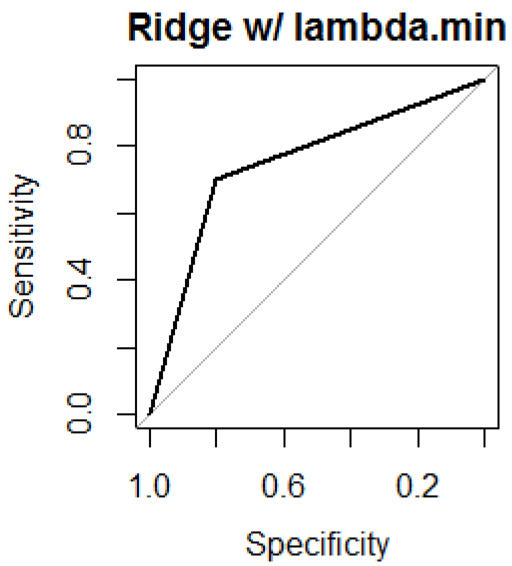
The misclassification errors are similar to the lasso models', slightly above 0.30. No variable has been set to zero, as expected from a ridge model.

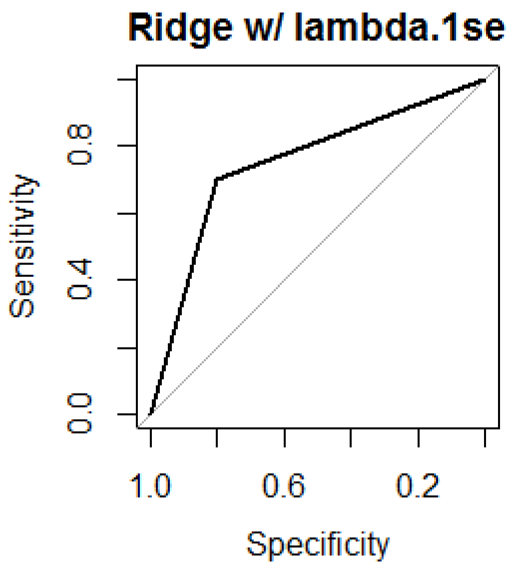
The same functions and arguments have been used to compute the following predictions and metrics:

```
> #Ridge model w/ lambda.min
> confusionMatrix(as.factor(pred.ridge.min)
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 72 27
         1 18 63

               Accuracy : 0.75
                 95% CI : (0.6801, 0.8114)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 6.046e-12

                  Kappa : 0.5

 Mcnemar's Test P-Value : 0.233

            Sensitivity : 0.7000
            Specificity : 0.8000
         Pos Pred Value : 0.7778
         Neg Pred Value : 0.7273
             Prevalence : 0.5000
         Detection Rate : 0.3500
   Detection Prevalence : 0.4500
      Balanced Accuracy : 0.7500

       'Positive' Class : 1
```

```
> #Ridge model w/lambda.1se
> confusionMatrix(as.factor(pred.ridge.1se)
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 72 27
         1 18 63

               Accuracy : 0.75
                 95% CI : (0.6801, 0.8114)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 6.046e-12

                  Kappa : 0.5

 Mcnemar's Test P-Value : 0.233

            Sensitivity : 0.7000
            Specificity : 0.8000
         Pos Pred Value : 0.7778
         Neg Pred Value : 0.7273
             Prevalence : 0.5000
         Detection Rate : 0.3500
   Detection Prevalence : 0.4500
      Balanced Accuracy : 0.7500

       'Positive' Class : 1
```

Although the two lambda parameters are different, they produced the same results. Both models score an overall accuracy of 0.75, and it can be noted again that the sensitivity is lower than the specificity. The ROC curves display similar imbalance, but a larger AUC:





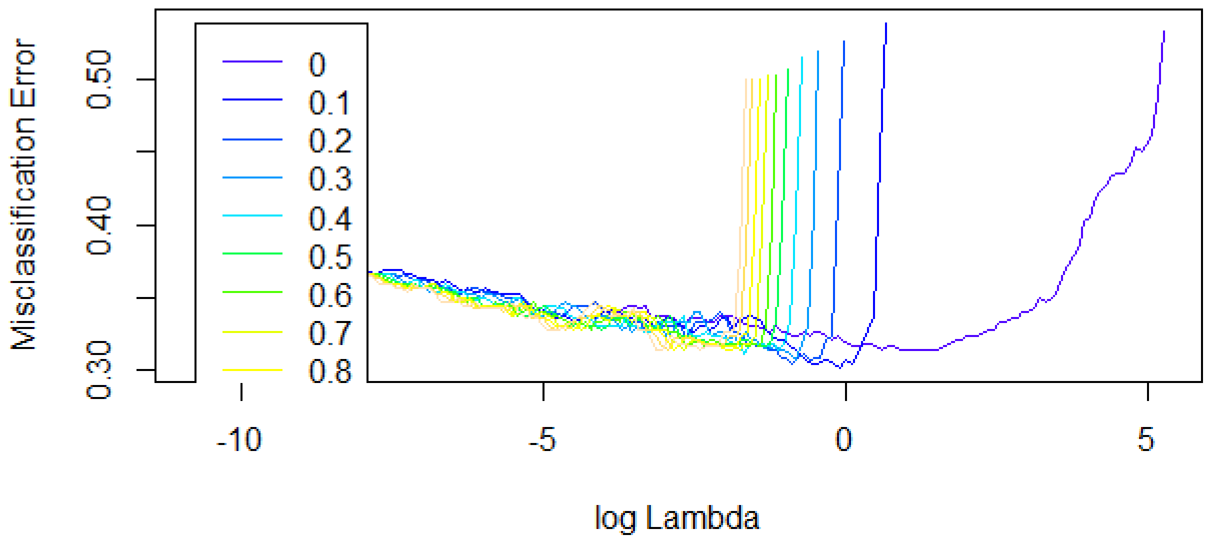
Therefore, the ridge models are more performant than the lasso models, at least from a predictive point of view. However, more testing should be performed to distinct between the two ridge models.

Elastic Net:

To fit the elastic net model, both alpha and lambda have to be determined. First, the cva.glmnet() function has been used to find an optimal alpha value, from 0 (ridge) to 1 (lasso):

```
> cv.net <- cva.glmnet(x, y, type.measure='class',
+                      family='binomial', alpha=seq(0,1,0.1))
```

Plotting the glmnet object gives the following output:

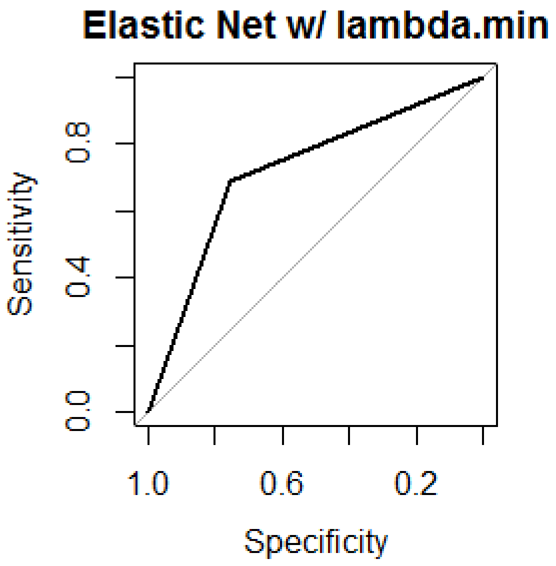


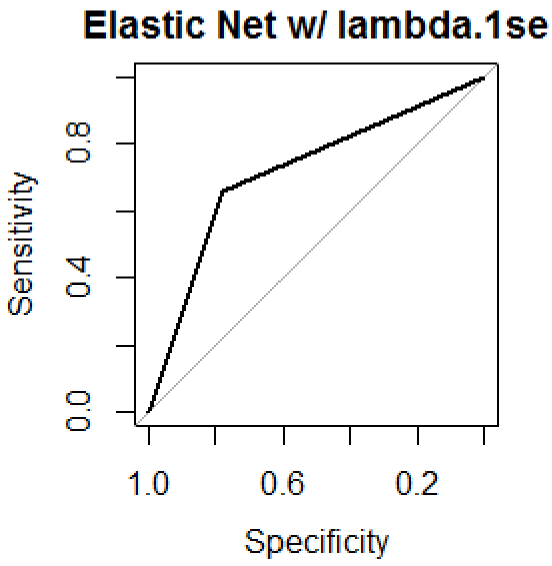
According to this graph, it seems that α=0.1 is an optimum, as it has the lowest misclassification error rate around 0.30 when log(λ) is around zero.

Thus, cv.glmnet() and α=0.1 have been used to fit the elastic net model. Here are the performance metrics and ROC curves that have been computed:

```
> #Elastic Net w/ lambda.min
> confusionMatrix(as.factor(pred.net.min),
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 68 28
         1 22 62

               Accuracy : 0.7222
                 95% CI : (0.6507, 0.7863)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 1.067e-09

                  Kappa : 0.4444

 Mcnemar's Test P-Value : 0.4795

            Sensitivity : 0.6889
            Specificity : 0.7556
         Pos Pred Value : 0.7381
         Neg Pred Value : 0.7083
             Prevalence : 0.5000
         Detection Rate : 0.3444
   Detection Prevalence : 0.4667
      Balanced Accuracy : 0.7222

       'Positive' Class : 1
```

```
> #Elastic Net w/ lambda.1se
> confusionMatrix(as.factor(pred.net.1se),
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 70 31
         1 20 59

               Accuracy : 0.7167
                 95% CI : (0.6448, 0.7812)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 2.766e-09

                  Kappa : 0.4333

 Mcnemar's Test P-Value : 0.1614

            Sensitivity : 0.6556
            Specificity : 0.7778
         Pos Pred Value : 0.7468
         Neg Pred Value : 0.6931
             Prevalence : 0.5000
         Detection Rate : 0.3278
   Detection Prevalence : 0.4389
      Balanced Accuracy : 0.7167

       'Positive' Class : 1
```





4. <u>DISCUSSION</u>

	Accuracy	Sensitivity	Specificity
Lasso w/ lambda.min	0.7222	0.6889	0.7556
Lasso w/ lambda.1se	0.7056	0.6444	0.7667
Ridge w/ lambda.min	0.75	0.7	0.8
Ridge w/ lambda.1se	0.75	0.7	0.8
Elastic Net w/ lambda.min	0.7222	0.6889	0.7556
Elastic Net w/ lambda.1se	0.7167	0.6556	0.7778

According to the results above, ridge regression (with *lambda.min* or *lambda.1se*) seems to be the most appropriate algorithm for predicting which NBA player will have a career length greater than 5 years – or not – with an overall accuracy of 0.75.

Several ameliorations could be brought to the model:

- Fine-tuning lambda between 0 and 0.1 since the elastic net model did not outperform the ridge one. The optimal lambda may lie between those two values.
- All of the models seem to underestimate the players' career length (low sensitivity). It could be interesting to find out why.
- Adding more observations to stabilise the classification rates and therefore obtaining more consistent models.