# Testing NHL Advanced statistics

Joseph Mazza

Analytics and Professional sports are becoming more intertwined. Analytics are mentioned in every broadcast of a sporting event. Professional clubs around the world are taking advantage of predictive analysis for processes like player development all the way to the contract negotiations. A sports conversation comparing athletes will always come down in numbers. This recent revelation holds true in the world of professional Ice Hockey as well. Overall, I wanted to investigate predictive modeling and analysis within the world's most popular hockey league, the National Hockey League (NHL). My interest specifically was looking into the statistics of the league over the past decade. This is because the NHL has continuously evolved over its 105-year history, and the past 10 – 15 years has often been referred to as the "modern era" of hockey. We can look at the statistics from this period and compute some predictive analysis. I will be putting forward 3 objectives that I want to meet with statistical analysis using a quantitative response variable, a qualitative response variable, and another variable for Principal Component Regression.

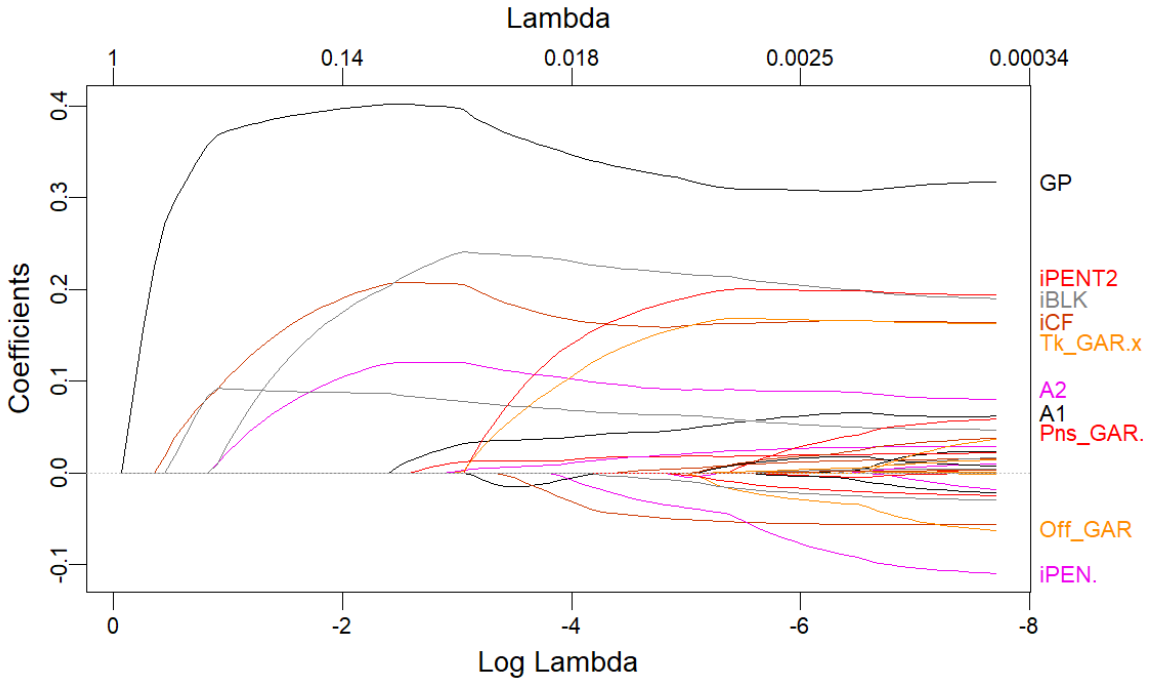
Of the three objectives I want to complete the first is taking a detailed look at the statistic "Points." Simply put, points in the NHL are a total of one player's Goals, Primary Assists, and Secondary Assists. The best way to explain this is a Secondary Assist is when Player C passes the puck to Player B, then player B passes to Player A and Player A scores the Goal. The goal goes to Player A, Primary assist to player B, and Secondary Assist to player C. Points have always been seen as one of the best ways to measure how effective a player is being offensive. I want to figure out what other statistics may be able to predict points besides Goals and Assists. This information is useful to many groups, like an NHL General Manager. For instance, if we wanted to try and find a good point producing player, we can look at the predictors related to Points, we do not just have to look at a prediction of points over time. We can try to look at predictors that influence points. Additionally, we can try and find a player that a team undervalues based on this theory. If we see that a player has positive predictor stats for points, but the points are not as high as expected, we can take advantage of that knowledge.

Secondly, I would like to dive into a statistic that I created. The stat is "Above Average" or "Below Average" Corsi difference per 60 minutes(C+-/60). This is a new and trendy stat that has been brought up in recent analytical talks. The best way to describe C+-/60 is a player with an above average C+-/60 can considered a good "two-way player." This means they thrive offensively AND defensively. I think most hockey folk would agree that any NHL player needs to be good at both offence and defense. Yet, to consistently be great at offence and defense is truly an art form, having a player or players that are fantastic both offensively and defensively is usually the difference in winning from a classical perspective. Looking deeper into this stat and seeing what other predictors of an above average two-way player may serve us well. In my mind observing the predictive analysis of this stat is particularly interesting for two reasons. There are elite players that deserve praise, but they have low or negative C+-/60. This can break off into another whole conversation of breaking down players by types, like "goal scorers" or "Stay at Home Defensemen." Then we can even build a hockey club that has all of the different types of players you think would help you win. The focus is on a player-type that I personally believe has the most value and is essential in an offensively focused league like the NHL.

Third, we investigate how the Time on Ice (TOI) for response variable works in a dimension reduction method. The method we will be using is Principal Components Analysis (PCA). PCA can tell us what "components" lead to most of the variation in the data. TOI is the most interesting variable to use because TOI can drastically vary between players that have similar statistics. The best players get the most TOI, but there are 18 skaters on a team and if you do not have a good balance of playing time amongst your team your best players will get too tired to be great. This should give us a better look to trends on why a player will have TOI. In other words, looking at TOI we can measure what groups of variables are related to TOI and we can interpret them.

Objective 1 technical analysis: "Points"

We start with hypothesis testing to see if Points as a response variable is correlated with other predictors in the data set. As I mentioned earlier, we will not be looking at Goals, Primary, or Secondary Assists for this. We then moved on to run linear regression, forward stepwise, backward stepwise, Ridge and Lasso regression. All the testing was completed with a training and test set to ensure the best accuracy rates. The best model turned out to be lasso model with an MSE of .0485. The lasso reduced some coefficients down to 0, so we can just focus on the ones that are left. With this information, we can take a more detailed look at Points when doing player analysis. For example, if a player has positive numbers for some of the coefficients that lasso wanted to keep like xPPO_GAR,TOI, ect. If the points are just not there as high as we think they should be based on looking at the predictors in this model, we can take a risk and try and acquire that player in hopes that our projection of his below average points will become a reality.




Plot of Lasso Regression with Points as the predictor

Objective 2 technical analysis: “Above or below average C+-/60”

Next, we start with hypothesis testing for a players C+-/60, we can find that there are some predictors that are related to C+-/60. We move on to run statistical tests that you would use on a categorical response variable, we judge these tests on accuracy rates rather than MSE. In the logistic regression test, we can predict an above average or below average C+-/60 67.91% of the time. For Linear Discriminant analysis our prediction rate drops to 50% of the time. The Quadratic Discriminant Analysis gives us an accuracy rate of 65%. Finally, we use K-Nearest Neighbors, this is where we get our best prediction rate. With K = 35 we can accurately predict an above or below average “two-way player” 70% of the time. KNN is a nonparametric classifier method, in this case we chose K = 35 because it is a good balance between a flexible model and a model with bias.

Objective 3 technical analysis: “TOI and PCR”

Our hypotheses testing for TOI proves that we should be able to test with TOI as the response variable. We start by running the PCR method and find that the MSE is around .40. This is a promising find, and we can look further into this test. When we ran other tests to check if PCR was better than other models, we found that it tested about the same as linear regression, but Lasso and Ridge were far superior when it came to this. The conclusion here is that we would not rely on a PCR test when looking at TOI as the response variable. Instead, we would lean on the Lasso or the Ridge regression for predictive analysis of TOI.

Overview of findings

Running these tests resulted in some interesting finds. We can look back at these three response variables and know that if we are looking in the right areas, we can start to predict how a player should be preforming. Specifically, in the lasso regression of Points I was shocked to see iPent2 and GIVE with positive coefficients. iPENT2 is the number of individual penalties taken and GIVE is the amount times the player gave the puck away. These two stats are seen as negative things, penalties put you down a player for 2 minutes and give aways mean you gave the puck to the other team. This is thought of as a bad idea. If we think about this though it makes a bit of sense, if a player is on the ice more, they will end up getting penalties and giving the puck away. It is the “price of doing business” in hockey and sports. We can even trust this analysis to take bets on players who may not be performing well in the “big” categories like Points or Goals. While sports in general are notoriously hard to predict, this analysis gives us a unique perspective and an advantage. In competitive sports that is sometimes the only difference between winning and losing.

Additionally, a final a note on early renderings of Corsi FOR and Goals had some intriguing signs. I ran quick tests on these before diving into Points, C+-/60, and Corsi For. The next steps for evolving this project could involve looking with more detail at Corsi For and Goals, while also continuing to test the response variables previously mentioned.

*Appendix:*

*Page 4: Organizing Data*

*Page 7: Objective 1 Technical Analysis*

*Page 19: Objective 2 Technical Analysis*

*Page 23: Objective 3 Technical Analysis*

*Organizing Data:*

**## Organizing Data:**

*library(dplyr)*

```
#TEST_NHL <- merge(Standard_Skaters_Table,GAR_Skater_Table, by = c("Player", "Season", "Team", "Position", "GP"))

#TEST_NHL <- merge(TEST_NHL,XGAR_Skaters_Table, by = c("Player", "Season", "Team", "Position", "GP"), all.x = TRUE, all.y = TRUE)

#TEST_NHL <- merge(TEST_NHL,RPAM_Skaters_Table, by = c("Player", "Season","Team", "Position", "GP"))

#TEST_NHL <- subset(TEST_NHL, select = -c(TakeX, DrawX, PensX))

#TEST_NHL <- subset(TEST_NHL, select = -c(points.BIN,points.AB) )

# Test area attempt 1 at Standardizing Data
#attach(TEST_NHL)

#og.G <- as.data.frame(log(G))

# Using Log does not seem to work, I belive it gives us "-inf" for 0s.
```

```
# Test 2

library(dplyr)


#SD.PG <- TEST_NHL %>% mutate_at(c("GP",  "TOI","G","A1","A2",
"Points","iSF","iFF","iCF","ixG", "Sh%","FSh%","xFSh%","iBLK","GIVE",
"TAKE","iHF","iHA","iPENT2","iPEND2","iPENT5","iPEND5","iPEN±","FOW","FOL","FO
±","EVO_GAR","EVD_GAR","PPO_GAR","SHD_GAR","Take_GAR.x","Draw_GAR.x",
"Off_GAR","Def_GAR","Pens_GAR.x","GAR","WAR","SPAR","xEVO_GAR","xEVD_GAR
","xPPO_GAR","xSHD_GAR","xOff_GAR","xDef_GAR","xGAR","xWAR","xSPAR","G±-
60","xG±-60","C±-60","GF-60","GA-60","xGF-60","xGA-60","CF-60","CA-60"), ~(scale(.)
%>% as.vector))

# No shot that is the most efficient way to do it but its done

#New_NHL <- SD.PG

#write.csv(New_NHL, "C:\\Users\\jm351\\OneDrive\\Documents\\School\\Predictive
Modeling\\Final_NHL.csv")

Objective 1 Technical Analysis

library(leaps)

library(caTools)

library(car)

library(quantmod)

library(MASS)
library(corrplot)

library(ISLR)
```

```
library(ISLR2)

library(dplyr)

library(olsrr)

library(glmnet)

Final_NHL <- read.csv("C:\\Users\\jm351\\OneDrive\\Documents\\School\\Predictive Modeling\\Final_NHL.csv")


Final_NHL <- na.omit(Final_NHL)



# Correlation analysis
Final_NHL %>% select(-c(Player,Season,Team,Position,G,A1,A2)) %>%
cor(Final_NHL$Points)

##                [,1]
## X          0.02886437
## ...1       0.02886437
## GP         0.71060816
## TOI        0.76934088
## Points     1.00000000
## iSF        0.90879521
## iFF        0.90611331
## iCF        0.88870422
## ixG        0.89211672
## Sh.        0.39942728
## FSh.       0.40408200
## xFSh.      0.34071013
## iBLK       0.24166523
## GIVE       0.70353128
```

```
## TAKE        0.80438812
## iHF         0.25835827
## iHA         0.52000767
## iPENT2      0.52552648
## iPEND2      0.70045289
## iPENT5     -0.07502826
## iPEND5     -0.07286899
## iPEN.       0.23690581
## FOW         0.44531361
## FOL         0.46004445
## FO.         0.05791853
## EVO_GAR     0.71431884
## EVD_GAR    -0.07182119
## PPO_GAR     0.59094828
## SHD_GAR     0.01366262
## Take_GAR.x  0.33554374
## Draw_GAR.x  0.13062802
## Off_GAR     0.79326456
## Def_GAR    -0.05972040
## Pens_GAR.x  0.35483045
## GAR         0.71277397
## WAR         0.71224728
## SPAR        0.71270158
## xEVO_GAR    0.73612051
## xEVD_GAR   -0.06183935
## xPPO_GAR    0.58410138
## xSHD_GAR    0.02074267
## xOff_GAR    0.79783853
## xDef_GAR   -0.04798711
## xGAR        0.71218983
## xWAR        0.71181745
```

```
## xSPAR       0.71250602
## G..60       0.39059146
## xG..60      0.31136955
## C..60       0.35690407
## GF.60       0.54777372
## GA.60       0.07157827
## xGF.60      0.47895196
## xGA.60      0.08468116
## CF.60       0.46061597
## CA.60      -0.08225616
```

# ( Use the lm() function to perform a multiple linear regression with Points as the response and all other variables except Player,Season,Team,Position,G,A1,A2 as the predictors. Use the summary() function to print the results. Comment on the output. For instance:

*Objective 1 Technical Analysis:*

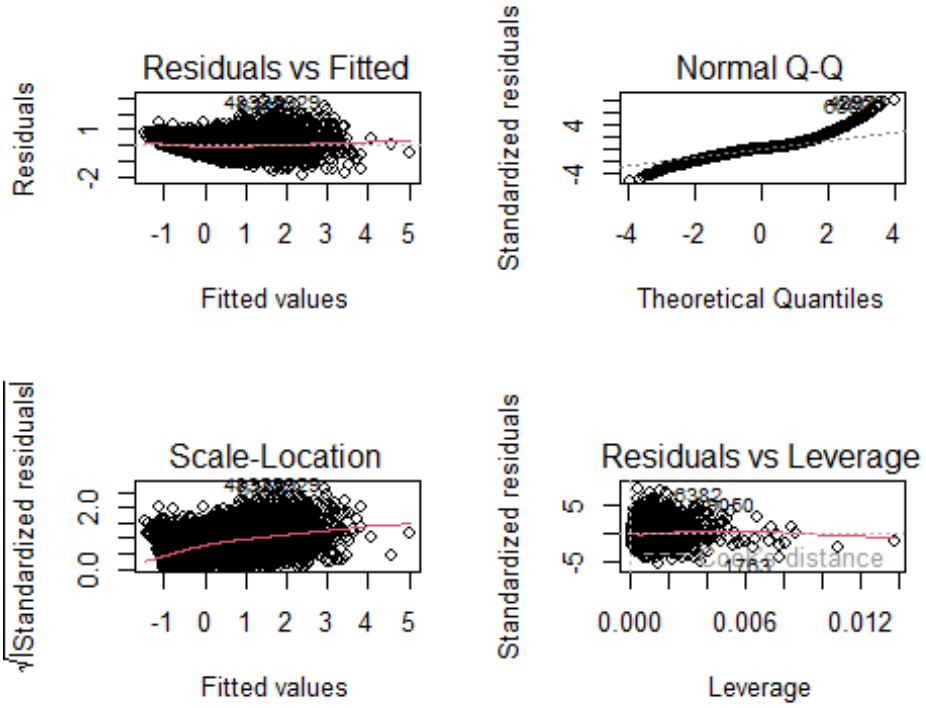
```
lm.fit1 <- lm(Points ~. -Player - Season - Team - Position - G - A1 - A2 - Final_NHL$...1-
Final_NHL$X, data = Final_NHL)

summary(lm.fit1)

##
## Call:
## lm(formula = Points ~ . - Player - Season - Team - Position -
##     G - A1 - A2 - Final_NHL$...1 - Final_NHL$X, data = Final_NHL)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.23175 -0.10327 -0.00396  0.09384  1.89572
```

```
##
## Coefficients: (1 not defined because of singularities)
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.262e-03  3.757e-03  -0.868  0.38531
## X            5.288e-07  4.888e-07   1.082  0.27938
## ...1               NA         NA      NA       NA
## GP          -1.222e-01  7.342e-03 -16.638  < 2e-16 ***
## TOI          4.972e-01  1.241e-02  40.054  < 2e-16 ***
## iSF          1.537e-01  2.286e-02   6.724 1.84e-11 ***
## iFF         -2.180e-01  3.515e-02  -6.202 5.74e-10 ***
## iCF          1.715e-01  2.265e-02   7.575 3.83e-14 ***
## ixG          2.870e-01  9.158e-03  31.334  < 2e-16 ***
## Sh.          3.790e-03  9.154e-03   0.414  0.67884
## FSh.         2.593e-04  9.346e-03   0.028  0.97787
## xFSh.        1.058e-02  3.323e-03   3.185  0.00145 **
## iBLK        -1.167e-01  4.940e-03 -23.622  < 2e-16 ***
## GIVE         8.545e-02  3.794e-03  22.524  < 2e-16 ***
## TAKE         3.214e-02  3.875e-03   8.294  < 2e-16 ***
## iHF         -5.229e-02  2.906e-03 -17.996  < 2e-16 ***
## iHA         -1.940e-02  3.697e-03  -5.247 1.57e-07 ***
## iPENT2       2.199e+04  1.146e+04   1.919  0.05497 .
## iPEND2      -2.265e+04  1.180e+04  -1.919  0.05497 .
## iPENT5       6.439e+03  3.355e+03   1.919  0.05497 .
## iPEND5      -6.354e+03  3.311e+03  -1.919  0.05497 .
## iPEN.        1.828e+04  9.525e+03   1.919  0.05497 .
## FOW          5.901e+03  1.858e+04   0.318  0.75074
## FOL         -5.567e+03  1.752e+04  -0.318  0.75074

## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.2164 on 13393 degrees of freedom
## Multiple R-squared:  0.9534, Adjusted R-squared:  0.9532
## F-statistic:  5165 on 53 and 13393 DF,  p-value: < 2.2e-16

attach(Final_NHL)
lm.fit1 <- lm(Points~GP + TOI + iSF + iFF + iCF + ixG + iBLK + GIVE + TAKE + iHF +
iHA, data = Final_NHL)

ols_vif_tol(lm.fit1)

##    Variables   Tolerance        VIF
## 1         GP 0.098204055   10.182879
## 2        TOI 0.039738750   25.164355
## 3        iSF 0.007871593  127.039089
## 4        iFF 0.003139205  318.551979
## 5        iCF 0.007444812  134.321729
## 6        ixG 0.067589008   14.795305
## 7       iBLK 0.183984050    5.435254
## 8       GIVE 0.256350926    3.900903
## 9       TAKE 0.268509342    3.724265
## 10       iHF 0.518148826    1.929947
## 11       iHA 0.293622296    3.405736

# Warning: essentially perfect fit: summary may be unreliableWarning: essentially
perfect fit: summary may be unreliable

# We take out GP, iSF,iFF, and iXG
lm.fit1 <- lm(Points~ TOI + iCF + iBLK + GIVE + TAKE + iHF + iHA, data = Final_NHL)

ols_vif_tol(lm.fit1)

##   Variables  Tolerance       VIF
## 1       TOI 0.06955491 14.377130
```

```
## 2       iCF 0.15855399  6.307000
## 3      iBLK 0.27911270  3.582782
## 4      GIVE 0.27785555  3.598992
## 5      TAKE 0.31009540  3.224814
## 6       iHF 0.59463157  1.681714
## 7       iHA 0.31282238  3.196702
```

```
# Much better VIF numbers


par(mfrow=c(2,2))
plot(lm.fit1)
```




```
# The residual tests look ok


# Just curious who the outlines pointed out in the graphs were.
Final_NHL[Final_NHL$...1 == "1763",]
```

```
## Brent Burns

Final_NHL[Final_NHL$...1 == "9050",]

## Mike Ribeiro

Final_NHL[Final_NHL$...1 == "6382",]

## Jonathan Huberdeau

# Fit test and training Data

set.seed(1)
train <- sample(c(TRUE, FALSE), nrow(Final_NHL), rep = TRUE)
test <- (!train)
Final.train <- Final_NHL[train,]
Final.test <- Final_NHL[test, ]

# Testing LM model
set.seed(1)
glm.fit <- lm(Points~ TOI + iCF + iBLK + GIVE + TAKE + iHF + iHA, data = Final.test,
subset = train)

lm.pred <- predict(glm.fit, Final.test, type = "response")

mean((lm.pred - Final.test$Points)^2)

## [1] 0.1353613

# MSE = .1353

# Try again with different seed to be sure
set.seed(5)
train <- sample(c(TRUE, FALSE), nrow(Final_NHL), rep = TRUE)
```

```
test <- (!train)
Final.train <- Final_NHL[train,]
Final.test <- Final_NHL[test, ]

# Testing LM model
set.seed(1)
glm.fit <- lm(Points~ TOI + iCF + iBLK + GIVE + TAKE + iHF + iHA, data = Final_NHL)

lm.pred <- predict(glm.fit, Final.test, type = "response")

mean((lm.pred - Final.test$Points)^2)

## [1] 0.1325317

# Leaving out Assists1, Assists2, and Goals
regfit.full <- regsubsets(Points ~. -Player - Season - Team - Position-G-A1-A2,
Final_NHL, really.big =T)



reg.summary <- summary(regfit.full)
reg.summary

## Subset selection object
## Call: regsubsets.formula(Points ~ . - Player - Season - Team - Position -
##     G - A1 - A2, Final_NHL, really.big = T)



# Plotting all of the functions at once will help us decide which model to select, (NOTE
type=L TELLS R TO CONNECT THE PLOTTED POINTS WITH LINES)

par(mfrow = c(2,2))
```

```
plot(reg.summary$rss, xlab = "Number of Variables", ylab = "RSS", type = "l")
which.min(reg.summary$rss)

## [1] 9

points(9,reg.summary$rss[9], col = "red", cex = 2, pch = 20)

plot(reg.summary$adjr2, xlab = "Number of Variables", ylab = "Adjusted RSq", type =
"l")
which.max(reg.summary$adjr2)

## [1] 9

points(9,reg.summary$adjr2[9], col = "red", cex = 2, pch = 20)


plot(reg.summary$cp, xlab = "Number of Variables", ylab = "Cp", type = "l")
which.min(reg.summary$cp)

## [1] 9

points(9,reg.summary$cp[9], col = "red", cex = 2, pch = 20)

which.min(reg.summary$bic)

## [1] 9

points(9,reg.summary$bic[9], col = "red", cex = 2, pch = 20)

# All tests point to the best model having 9 predictors

# Next we run the model though Froward and backward stepwise selection

#Forward and Backward step wise selection
```

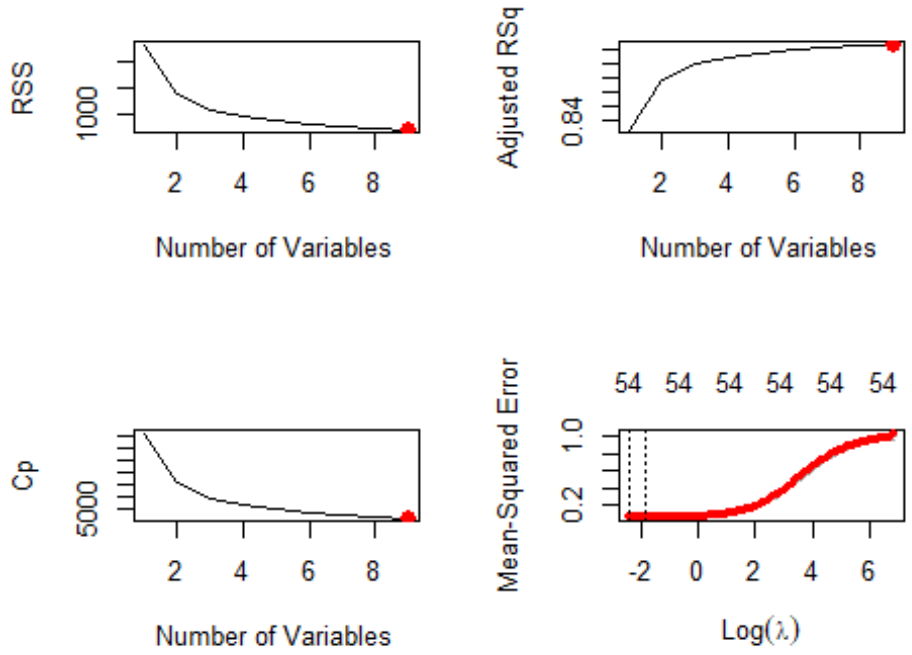
```
# We can also use the regsubsets function to preform forward or backward step wise section

regfit.FW <- regsubsets(Points ~.  -Player - Season - Team - Position-G-A1-A2, data = Final_NHL, method = "forward")

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax, force.in =
## force.in, : 1 linear dependencies found

## Reordering variables and trying again:

summary(regfit.FW)

## Subset selection object
## Call: regsubsets.formula(Points ~ . - Player - Season - Team - Position -
##     G - A1 - A2, data = Final_NHL, method = "forward")
## 54 Variables  (and intercept)
##           Forced in Forced out



# Testing model with forward stepwise 9 predictors
set.seed(1)
glm.fit <- lm(Points~ TOI + iHF + iSF + ixG + GIVE + FOL + Off_GAR + xOff_GAR + xGF.60, data = Final_NHL)

lm.pred <- predict(glm.fit, Final.test, type = "response")

mean((lm.pred - Final.test$Points)^2)

## [1] 0.05604322

# MSE for forward selection with 9 predictors .0560
ols_vif_tol(glm.fit)
```

```
##   Variables Tolerance       VIF
## 1       TOI 0.1429945  6.993274
## 2       iHF 0.6590648  1.517302
## 3       iSF 0.0650498 15.372836
## 4       ixG 0.1009976  9.901224
## 5      GIVE 0.2805022  3.565034
## 6       FOL 0.7018934  1.424718
## 7   Off_GAR 0.3472267  2.879963
## 8  xOff_GAR 0.2883517  3.467988
## 9    xGF.60 0.5660869  1.766513

# The VIF test tells us to take out iSF for multicolinarity

set.seed(1)
glm.fit <- lm(Points~ TOI + iHF + ixG + GIVE + FOL + Off_GAR + xOff_GAR + xGF.60,
data = Final_NHL)

lm.pred <- predict(glm.fit, Final.test, type = "response")

mean((lm.pred - Final.test$Points)^2)

## [1] 0.05686734

# MSE for forward selection using 8 predictors is .0567
ols_vif_tol(glm.fit)

##   Variables Tolerance      VIF
## 1       TOI 0.2043137 4.894434
## 2       iHF 0.6603184 1.514421
## 3       ixG 0.3269219 3.058835
## 4      GIVE 0.2826748 3.537634
## 5       FOL 0.7432097 1.345515
## 6   Off_GAR 0.3472813 2.879510
```

```
## 7  xOff_GAR 0.2898350 3.450239
## 8    xGF.60 0.5667971 1.764300

regfit.BW <- regsubsets(Points ~.  -Player - Season - Team - Position-G-A1-A2, data =
Final_NHL,, method = "backward")

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax, force.in =
## force.in, : 1 linear dependencies found

## Reordering variables and trying again:

summary(regfit.BW)

# Testing model with Backward stepwise 7 predictors
set.seed(1)
glm.fit <- lm(Points~ TOI + EVO_GAR + iBLK + ixG + GIVE + Off_GAR + xG..60 +
xGAR + iHF, data = Final_NHL)


lm.pred <- predict(glm.fit, Final.test, type = "response")


mean((lm.pred - Final.test$Points)^2)

## [1] 0.05559715

# MSE for forward selection wiht 9 predictors is  .0556
ols_vif_tol(glm.fit)

##   Variables  Tolerance       VIF
## 1       TOI 0.09379140 10.661958
## 2   EVO_GAR 0.08759695 11.415923
## 3      iBLK 0.20199855  4.950531
## 4       ixG 0.21225151  4.711392
## 5      GIVE 0.28698491  3.484504
## 6   Off_GAR 0.07158272 13.969852
## 7    xG..60 0.54110341  1.848076
```

```
## 8      xGAR 0.30507011  3.277935
## 9       iHF 0.65528224  1.526060

# If we take out Off_GAR
set.seed(1)
glm.fit <- lm(Points~ TOI + EVO_GAR + iBLK + ixG + GIVE  + xG..60 + xGAR + iHF,
data = Final_NHL)

lm.pred <- predict(glm.fit, Final.test, type = "response")

mean((lm.pred - Final.test$Points)^2)

## [1] 0.06648192

ols_vif_tol(glm.fit)

##   Variables  Tolerance       VIF
## 1       TOI 0.09379207 10.661882
## 2   EVO_GAR 0.50465929  1.981535
## 3      iBLK 0.20200469  4.950380
## 4       ixG 0.21852828  4.576067
## 5      GIVE 0.28851260  3.466053
## 6    xG..60 0.55404368  1.804912
## 7      xGAR 0.32770893  3.051488
## 8       iHF 0.66115149  1.512513

# So far the fowrward selection method with while using best subset selection has the
best MSE and VIF.

# we use glmnet package to preform ridge regression and the lasso, with λ chosen by
cross validation.

# Starting with Ridge
```

```
x=model.matrix(Points ~.-Player - Season - Team - Position-G-A1-A2,Final_NHL)
y=Final_NHL$Points


set.seed(1)
y.test<-y[test]

set.seed(1)
cv.out <- cv.glmnet(x[train,], y[train], alpha = 0)
plot(cv.out)
```




```
bestlam = cv.out$lambda.min
bestlam

## [1] 0.09204818

# Therefore, we see that the value of λ that results in the smallest cross validation error is .0942
```

```
ridge.mod=glmnet(x[train,],y[train],alpha=0)
ridge.pred <- predict(ridge.mod, s = bestlam, newx = x[test,])
mean((ridge.pred - y.test)^2)

## [1] 0.05321039

# Using the cross-validation we get a MSE of .053
out <- glmnet(x,y,alpha=0)

predict(out,type="coefficients",s=bestlam)

## 56 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept) -3.133130e-03
## (Intercept)  .
## X            2.327871e-07
## ...1         2.366840e-07
## GP           1.540353e-02
## TOI          1.274578e-01
## iSF          9.387581e-02
## iFF          8.501857e-02
## iCF          8.336939e-02
## ixG          1.504712e-01
## Sh.          2.697187e-03
## FSh.         8.377292e-03
## xFSh.        3.009881e-02
## iBLK        -4.971674e-02
## GIVE         9.209507e-02
## TAKE         5.253300e-02
## iHF         -4.901158e-02
## iHA         -1.055925e-02
## iPENT2       3.356309e-02
```

```
## iPEND2       4.609572e-02
## iPENT5      -3.664091e-03
## iPEND5      -5.262767e-03
## iPEN.        1.535642e-02
## FOW          2.662763e-02
## FOL          2.922896e-02
## FO.         -3.894063e-03
## EVO_GAR      2.156320e-02
## EVD_GAR     -1.828644e-02
## PPO_GAR      7.705095e-02
## SHD_GAR     -8.763431e-03
## Take_GAR.x   1.450961e-02
## Draw_GAR.x  -3.444012e-02
## Off_GAR      4.346269e-02
## Def_GAR     -2.048740e-02
## Pens_GAR.x  -1.234087e-02
## GAR          2.534778e-02
## WAR          1.870209e-02
## SPAR         1.937735e-02
## xEVO_GAR     4.242247e-02
## xEVD_GAR    -1.624386e-02
## xPPO_GAR     3.583457e-02
## xSHD_GAR     3.772788e-03
## xOff_GAR     4.681051e-02
## xDef_GAR    -1.289140e-02
## xGAR         3.081607e-02
## xWAR         2.414582e-02
## xSPAR        2.337784e-02
## G..60        2.328650e-02
## xG..60      -1.464534e-02
## C..60       -9.716793e-03
```

```
## GF.60        4.221349e-02
## GA.60        1.721002e-02
## xGF.60      -2.761638e-02
## xGA.60      -9.278530e-03
## CF.60       -2.323474e-02
## CA.60       -9.945199e-03
```

# none of the coefficients are exactly zero, but they are shrunken towards 0. This is because ridge regression does not preform variable selection.

# Next we Fit a lasso model on the training set, with λ chosen by cross validation. Report the test error obtained, along with the number of non-zero coefficient estimates.

```
lasso.mod=glmnet(x[train,],y[train],alpha=1)
plot(lasso.mod)
```

# We can see from the coefficient plot that depending on the choice of tuning parameter, some of the coefficients will be exactly equal to zero. We now perform cross-validation and compute the associated test error

```
set.seed(1)
cv.out=cv.glmnet(x[train,],y[train],alpha=1)
plot(cv.out)
bestlam=cv.out$lambda.min
lasso.pred=predict(lasso.mod,s=bestlam,newx=x[test,])
mean((lasso.pred-y.test)^2)
```

```
## [1] 0.04853196
```

# The MSE is .0485, this is  lower than the test set MSE of the  of least squares and ridge regression with λ chosen by cross-validation. However, the lasso has a substantial advantage over ridge regression in that the resulting coefficient estimates are sparse.

Here we see that 9 of the 16 coefficient estimates are exactly zero. So the lasso model with λ chosen by cross-validation contains only 9 variables.

```
out=glmnet(x,y,alpha=1)
lasso.coef=predict(out,type="coefficients",s=bestlam)
lasso.coef

## 56 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept) -1.737110e-03
## (Intercept)  .
## X            2.591801e-07
## ...1         6.492947e-23
## GP          -1.057497e-01
## TOI          4.642951e-01
## iSF          7.398055e-02
## iFF          .
## iCF          5.316758e-02
## ixG          2.678066e-01
## Sh.          .
## FSh.         5.089170e-03
## xFSh.        1.205995e-02
## iBLK        -1.139719e-01
## GIVE         8.756510e-02
## TAKE         3.340594e-02
## iHF         -5.311498e-02
## iHA         -1.872697e-02
## iPENT2       .
## iPEND2       1.103000e-02
## iPENT5       6.764599e-03
## iPEND5       7.614797e-03
```

```
## iPEN.        7.844107e-02
## FOW          .
## FOL          3.912678e-02
## FO.         -9.919417e-04
## EVO_GAR      .
## EVD_GAR      .
## PPO_GAR      8.673305e-02
## SHD_GAR     -5.238055e-03
## Take_GAR.x   .
## Draw_GAR.x  -2.369545e-04
## Off_GAR      6.760463e-02
## Def_GAR     -2.204783e-02
## Pens_GAR.x  -6.003184e-02
## GAR          .
## WAR          .
## SPAR         .
## xEVO_GAR     .
## xEVD_GAR     .
## xPPO_GAR     6.600906e-03
## xSHD_GAR     1.749001e-02
## xOff_GAR     1.603049e-01
## xDef_GAR     .
## xGAR         .
## xWAR         .
## xSPAR        .
## G..60        .
## xG..60       .
## C..60       -8.867120e-03
## GF.60        9.797916e-02
## GA.60        6.488749e-04
## xGF.60      -5.789051e-02
```

```
## xGA.60      -2.003489e-02
## CF.60       -1.790986e-02
## CA.60        .

lasso.coef[lasso.coef!=0]
```

```
Objective 2 Technical Analysis:

library(leaps)

Final_NHL <- read.csv("C:||Users||jm351||OneDrive||Documents||School||Predictive Modeling||Final_NHL.csv")

Final_NHL <- na.omit(Final_NHL)
attach(Final_NHL)
```

# We are going to try and predict if a player is above average defencive player or below average defensive player based on `C±-60`

#Looking at this correlation matrix there seems to be a couple of variables that are correlated with crim. indus, rad, tax, nox, and age are the best.

**## We will start by looking at logistic regression**

# We need to make crim a binary variable with 1 being above the median crim rate

```
Corsi.BIN <- rep(0, length(Final_NHL$C..60))
Corsi.BIN[Final_NHL$C..60 > median(Final_NHL$C..60)] <- 1
Final_NHL <- data.frame(Final_NHL, Corsi.BIN)
```

#Now creating the training data and test data

```
train <- 1:(dim(Final_NHL)[1]/2)
```

```
test <- (dim(Final_NHL)[1]/2 + 1):dim(Final_NHL)[1]
Final_NHL.train <- Final_NHL[train,]
Final_NHL.test <- Final_NHL[test,]
Corsi.test <- Corsi.BIN[test]


glm.fit <- glm(Corsi.BIN ~G + A1 + A2 + iBLK + GIVE + TAKE + iPENT2 + iPEND2 +
EVO_GAR + EVD_GAR, data = Final_NHL.train, family = binomial)
summary(glm.fit)

glm.fit <- glm(Corsi.BIN ~ A2 + iBLK + GIVE + TAKE + EVO_GAR + EVD_GAR, data =
Final_NHL.train, family = binomial)
summary(glm.fit)

##
## Call:
## glm(formula = Corsi.BIN ~ A2 + iBLK + GIVE + TAKE + EVO_GAR +
##     EVD_GAR, family = binomial, data = Final_NHL.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2288  -0.9356  -0.1515   0.9803   2.7940
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.09221    0.02884   3.198  0.00139 **
## A2           0.33416    0.05653   5.911 3.40e-09 ***
## iBLK        -0.78956    0.04527 -17.441  < 2e-16 ***
## GIVE         0.38263    0.05681   6.735 1.64e-11 ***
## TAKE         0.29566    0.04849   6.097 1.08e-09 ***
## EVO_GAR      0.45341    0.04004  11.324  < 2e-16 ***
## EVD_GAR      1.05952    0.03778  28.044  < 2e-16 ***
```

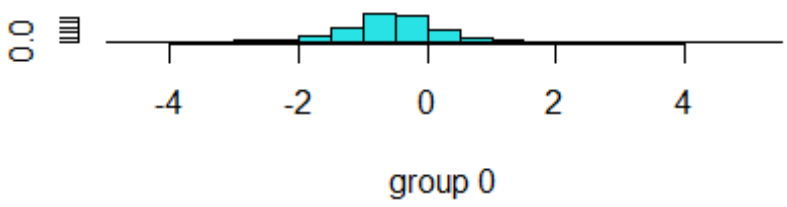
```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 9320.0  on 6722  degrees of freedom
## Residual deviance: 7470.6  on 6716  degrees of freedom
## AIC: 7484.6
##
## Number of Fisher Scoring iterations: 5

glm.probs <- predict(glm.fit, Final_NHL.test)
glm.pred <- rep(0, length(glm.probs))
#Producing a confusion matrix
glm.pred[glm.probs > .5] = 1

#Producing a confusion matrix
table(glm.pred,Corsi.test)

##         Corsi.test
## glm.pred    0    1
##        0 2919 1648
##        1  442 1714

mean(glm.pred == Corsi.test)

## [1] 0.6891269
```

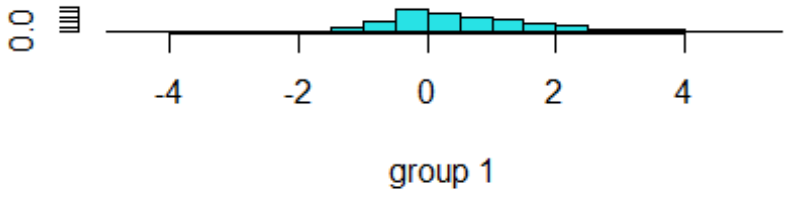
#This is showing that the logistic regression correctly predicted the Above average or below aveage rate 68.91% of the time.

**## Now using LDA.**

#Now moving on to Linear Discriminant Analysis:

```
Lda.fit <- lda(Corsi.BIN ~G + A1 + A2 + iBLK + GIVE + TAKE + iPENT2 + iPEND2 +
EVO_GAR + EVD_GAR, data = Final_NHL.train)
Lda.fit

#The output indicates that 50.1% of the training observations fall above the median
crime rate.


plot(Lda.fit)
```




```
Lda.pred <- predict(Lda.fit, Final_NHL.test)
#The predict function returns a list with 3 elements. Class : contains LDA predictions
about the movement of the market. Posterior : contains the probability that the
corresponding observation belongs to the kth class. X : contains the linear discriminant.
```

```
lda.class <- Lda.pred$class
table(lda.class, Corsi.test)

##          Corsi.test
## lda.class    0    1
##         0 2613 1232
##         1  748 2130

mean(lda.class == Corsi.test)

## [1] 0.7054886

#This gives us a test error rate of 70.54%%

##(f) Repeat (d) using QDA.
#Quadratic Discriminant Analysis:

#Now we want to fit a QDA model to the SMarket Data
qda.fit <- qda(Corsi.BIN ~G + A1 + A2 + iBLK + GIVE + TAKE + iPENT2 + iPEND2 +
EVO_GAR + EVD_GAR, data = Final_NHL.train)
qda.fit

#Output shows the group means but does not contain the coefficients of the linear
discriminant.

qda.class <- predict(qda.fit, Final_NHL.test)$class
table(qda.class, Corsi.test)

##          Corsi.test
## qda.class    0    1
##         0 2665 1630
##         1  696 1732

mean(qda.class == Corsi.test)
```

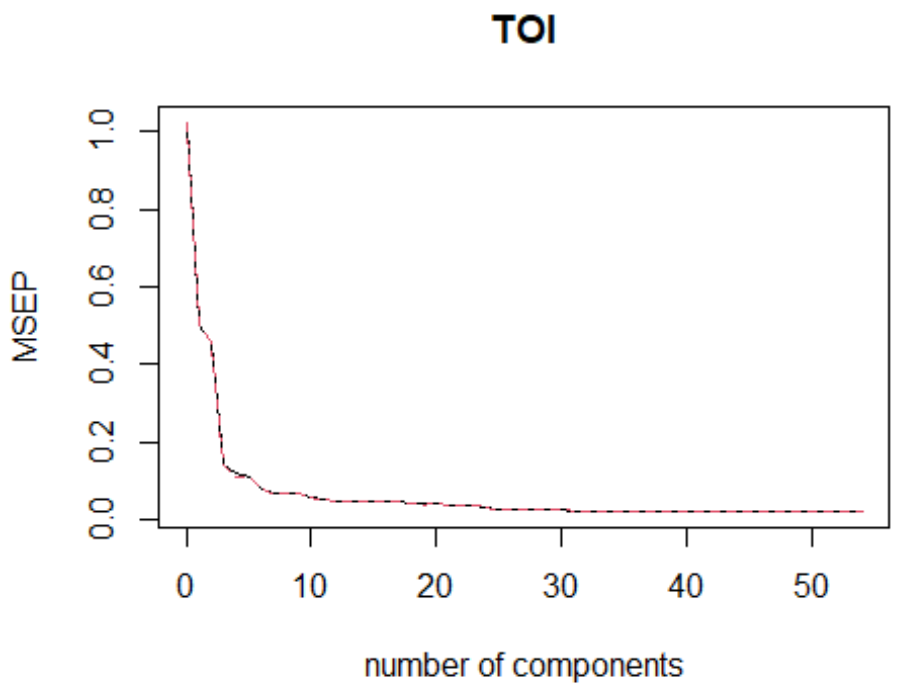
```
## [1] 0.6540235

#The QDA predictions are accurate almost 65.40% of the time.


library(class) # The class library is used for KNN

## Warning: package 'class' was built under R version 4.2.2

# Before performing KNN, separate dataframes are made for the predictors and
responses
Smarket_train_predictors = data.frame(Final_NHL.train$G, Final_NHL.train$A1,
Final_NHL.train$A2, Final_NHL.train$iBLK , Final_NHL.train$GIVE ,
Final_NHL.train$TAKE , Final_NHL.train$iPENT2 , Final_NHL.train$iPEND2 ,
Final_NHL.train$EVO_GAR , Final_NHL.train$EVD_GAR)
Smarket_test_predictors = data.frame(Final_NHL.test$G, Final_NHL.test$A1,
Final_NHL.test$A2,  Final_NHL.test$iBLK ,  Final_NHL.test$GIVE ,
Final_NHL.test$TAKE ,  Final_NHL.test$iPENT2 ,  Final_NHL.test$iPEND2 ,
Final_NHL.test$EVO_GAR ,  Final_NHL.test$EVD_GAR)
Smarket_train_response = Final_NHL.train$Corsi.BIN
Smarket_test_response = Final_NHL.test$Corsi.BIN

# Perform KNN with K=3
set.seed(1)

Smarket_predictions_knn = knn(Smarket_train_predictors,
                      Smarket_test_predictors,
                      Smarket_train_response,
                      k=3)

# See a confusion matrix to assess the accuracy of KNN
table(Smarket_predictions_knn, Smarket_test_response)
```

```
##                     Smarket_test_response
## Smarket_predictions_knn    0    1
##                     0 2209 1144
##                     1 1152 2218

(2209 + 2218) /6723

## [1] 0.6584858

# With KNN = 3  we can accurately predict 65.85% of the time.

# Perform KNN with K=15
set.seed(1)

Smarket_predictions_knn = knn(Smarket_train_predictors,
                     Smarket_test_predictors,
                     Smarket_train_response,
                     k=15)

# See a confusion matrix to assess the accuracy of KNN
table(Smarket_predictions_knn, Smarket_test_response)

##                     Smarket_test_response
## Smarket_predictions_knn    0    1
##                     0 2350 1038
##                     1 1011 2324

(2305 + 2328) /6723

## [1] 0.6891269

# With KNN = 15  we can accurately predict 68.91% of the time.
```

```
# Perform KNN with K=35
set.seed(1)
```

Objective 3 Technical Analysis:

```
#### Fit a PCR model on the training set, with M chosen by cross validation. Report the test error obtained, along with the value of M selected by cross-validation.


Final_NHL <- read.csv("C:||Users||jm351||OneDrive||Documents||School||Predictive Modeling||Final_NHL.csv")

Final_NHL <- na.omit(Final_NHL)
attach(Final_NHL)

set.seed(5)
train <- sample(c(TRUE, FALSE), nrow(Final_NHL), rep = TRUE)
test <- (!train)
Final.train <- Final_NHL[train,]
Final.test <- Final_NHL[test, ]

library(pls)

set.seed(1)
pcr.fit=pcr(TOI ~.-Player - Season - Team - Position-G-A1-A2, data=Final.train, scale=TRUE,validation="CV")

summary(pcr.fit)
validationplot(pcr.fit,val.type="MSEP")
```

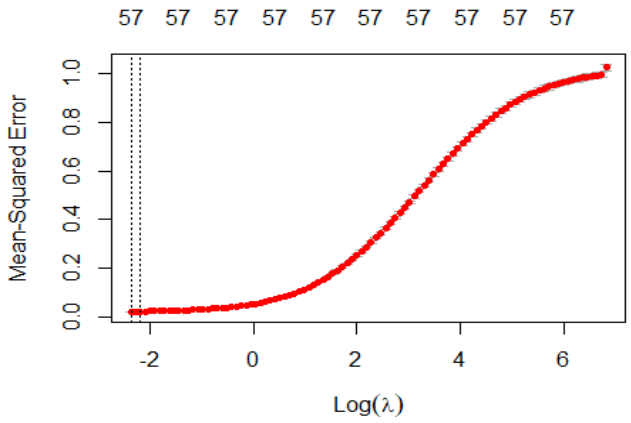



```
# We see that the smallest cross-validation error occurs when M = 14 components are used.

pcr.pred=predict(pcr.fit,Final.test,ncomp=14)

mean((pcr.pred-Final.test$Points)^2)

## [1] 0.4318436

# This test set MSE 0.4318436 competitive with the results obtained below.

# We run some tests against TOI to see how well that MSE holds up

# We can also use the regsubsets function to preform forward or backward step wise section

regfit.FW <- regsubsets(TOI ~.  -Player - Season - Team - Position-TOI, data = Final_NHL, method = "forward")
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax, force.in =
## force.in, : 1 linear dependencies found

## Reordering variables and trying again:

summary(regfit.FW)



# Testing model with forward stepwise 9 predictors
set.seed(1)
glm.fit <- lm(TOI~ GP + A2 + Points + iCF + iBLK + GIVE + iPENT2 + iPEND5+
Take_GAR.x, data = Final_NHL)

lm.pred <- predict(glm.fit, Final.test, type = "response")

mean((lm.pred - Final.test$Points)^2)

## [1] 0.4315324

# MSE for forward selection with 9 predictors .4315
ols_vif_tol(glm.fit)

##    Variables  Tolerance       VIF
## 1         GP 0.18225899  5.486698
## 2         A2 0.17930968  5.576944
## 3     Points 0.09098482 10.990845
## 4        iCF 0.13093294  7.637497
## 5       iBLK 0.33656334  2.971209
## 6       GIVE 0.28061955  3.563544
## 7     iPENT2 0.12905599  7.748575
## 8     iPEND5 0.77644196  1.287926
## 9 Take_GAR.x 0.23757378  4.209219
```

```
regfit.BW <- regsubsets(TOI ~.  -Player - Season - Team - Position-TOI, data =
Final_NHL, method = "backward")

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax, force.in =
## force.in, : 1 linear dependencies found

# Testing model with Backward stepwise 7 predictors
set.seed(1)
glm.fit <- lm(TOI~ GP + Points + iCF + iBLK + GIVE + iPENT2 + iPEND5 + iPEN. +
Take_GAR.x, data = Final_NHL)


lm.pred <- predict(glm.fit, Final.test, type = "response")


mean((lm.pred - Final.test$Points)^2)

## [1] 0.4321217

# MSE for forward selection wiht 9 predictors is  .4321
ols_vif_tol(glm.fit)

##    Variables Tolerance      VIF
## 1         GP 0.1840564 5.433116
## 2     Points 0.1534109 6.518441
## 3        iCF 0.1335648 7.487000
## 4       iBLK 0.3412662 2.930264
## 5       GIVE 0.2843261 3.517089
## 6     iPENT2 0.1273793 7.850566
## 7     iPEND5 0.7729089 1.293814
## 8      iPEN. 0.4731077 2.113684
## 9 Take_GAR.x 0.2166368 4.616022

#
# So far the PCR method with has the best MSE.
```

```
# we use glmnet package to preform ridge regression and the lasso, with λ chosen by cross validation.

# Starting with Ridge

x=model.matrix(TOI ~.  -Player - Season - Team - Position-TOI, data = Final_NHL)
y=Final_NHL$TOI



set.seed(1)
y.test<-y[test]

set.seed(1)
cv.out <- cv.glmnet(x[train,], y[train], alpha = 0plot(cv.out)
```




```
bestlam = cv.out$lambda.min
bestlam

## [1] 0.09277917

# Therefore, we see that the value of λ that results in the smallest cross validation error is .0922
ridge.mod=glmnet(x[train,],y[train],alpha=0)
```

```
ridge.pred <- predict(ridge.mod, s = bestlam, newx = x[test,])
mean((ridge.pred - y.test)^2)

## [1] 0.02239384

# Using the cross-validation we get a MSE of .022
out <- glmnet(x,y,alpha=0)


predict(out,type="coefficients",s=bestlam)

## 59 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept) -2.155182e-03
## (Intercept)  .
## X            1.595180e-07
## ...1         1.636699e-07
## GP           2.180408e-01
## G           -1.344235e-02
## A1           4.876998e-02
## A2           8.557069e-02
## Points       3.796361e-02
## iSF          5.251121e-02
## iFF          5.819923e-02
## iCF          8.059317e-02
## ixG          1.907275e-03
## Sh.          1.700587e-03
## FSh.         2.495149e-03
## xFSh.       -1.431054e-02
## iBLK         1.909053e-01
## GIVE         7.536780e-02
## TAKE         3.927684e-02
## iHF          1.642352e-02
```

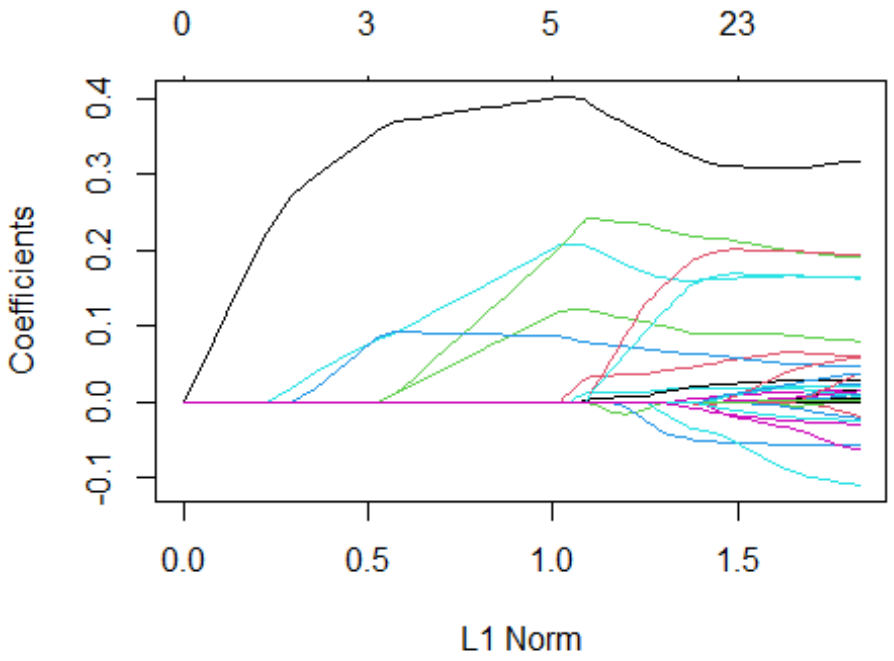
```
## iHA          6.229419e-02
## iPENT2       1.351285e-01
## iPEND2       4.695023e-02
## iPENT5      -2.219059e-02
## iPEND5      -2.442175e-02
## iPEN.       -1.023052e-01
## FOW          8.002695e-03
## FOL          7.842374e-03
## FO.          3.198881e-03
## EVO_GAR     -1.301517e-02
## EVD_GAR      5.883566e-03
## PPO_GAR     -1.472730e-02
## SHD_GAR      7.918568e-03
## Take_GAR.x   1.158415e-01
## Draw_GAR.x  -3.931772e-02
## Off_GAR     -1.565417e-02
## Def_GAR      7.670473e-03
## Pens_GAR.x   6.217591e-02
## GAR          2.877064e-03
## WAR          2.709965e-03
## SPAR         2.075913e-03
## xEVO_GAR    -1.459037e-03
## xEVD_GAR     9.297070e-03
## xPPO_GAR    -1.320205e-03
## xSHD_GAR    -1.924204e-02
## xOff_GAR    -1.405216e-03
## xDef_GAR     1.955033e-03
## xGAR         1.213684e-02
## xWAR         1.218632e-02
## xSPAR        1.106131e-02
## G..60       -3.738621e-03
```

```
## xG..60      -1.673578e-02
## C..60       -3.046515e-03
## GF.60       -5.828922e-03
## GA.60       -2.643844e-04
## xGF.60       3.944337e-03
## xGA.60       3.003434e-02
## CF.60       -2.194439e-02
## CA.60       -1.853862e-02
```

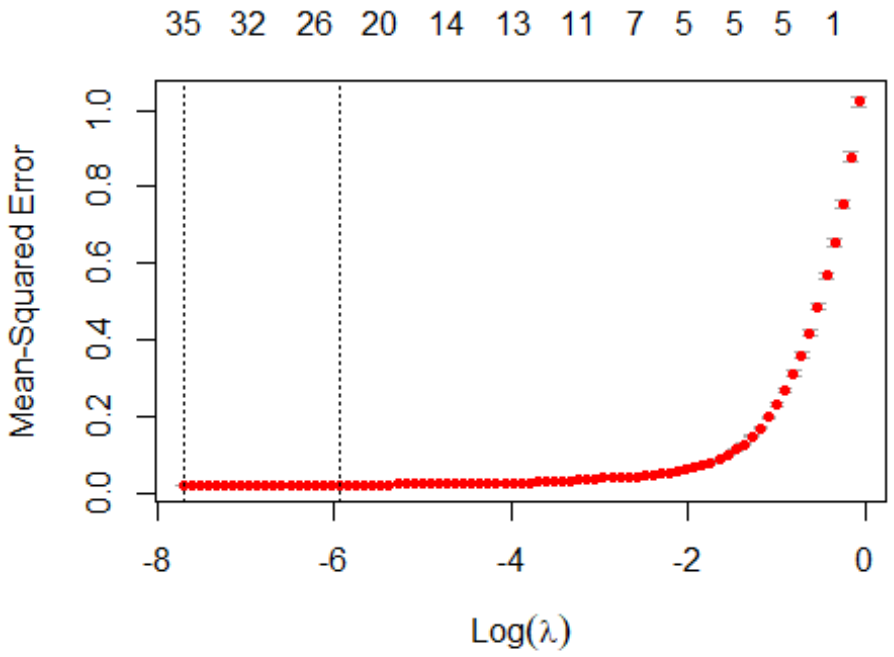
# none of the coefficients are exactly zero, but they are shrunken towards 0. This is because ridge regression does not preform variable selection.

# Next we Fit a lasso model on the training set, with λ chosen by cross validation. Report the test error obtained, along with the number of non-zero coefficient estimates.

```
lasso.mod=glmnet(x[train,],y[train],alpha=1)
plot(lasso.mod)
```




```
# We can see from the coefficient plot that depending on the choice of tuning parameter, some of the coefficients will be exactly equal to zero. We now perform cross-validation and compute the associated test error

set.seed(1)
cv.out=cv.glmnet(x[train,],y[train],alpha=1)
plot(cv.out)
```




```
bestlam=cv.out$lambda.min
lasso.pred=predict(lasso.mod,s=bestlam,newx=x[test,])
mean((lasso.pred-y.test)^2)

## [1] 0.01981515
```

# The MSE is .0198, this is  lower than the test set MSE of the  of least squares and ridge regression with λ chosen by cross-validation. However, the lasso has a substantial advantage over ridge regression in that the resulting coefficient estimates are sparse. Here we see that 9 of the 16 coefficient estimates are exactly zero. So the lasso model with λ chosen by cross-validation contains only 9 variables.

```
out=glmnet(x,y,alpha=1)
lasso.coef=predict(out,type="coefficients",s=bestlam)
lasso.coef

## 59 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
```

```
## (Intercept) -1.281394e-03
## (Intercept)  .
## X            1.916548e-07
## ...1         .
## GP           3.134973e-01
## G            .
## A1           5.360610e-02
## A2           7.681090e-02
## Points       2.835549e-02
## iSF          4.823945e-03
## iFF          .
## iCF          1.617985e-01
## ixG          3.689175e-03
## Sh.          .
## FSh.         3.081901e-03
## xFSh.        .
## iBLK         1.900683e-01
## GIVE         4.874396e-02
## TAKE         2.148957e-02
## iHF          .
## iHA          2.605080e-02
## iPENT2       1.986538e-01
## iPEND2       .
## iPENT5       .
## iPEND5      -5.557904e-02
## iPEN.       -1.151125e-01
## FOW          1.701110e-02
## FOL          .
## FO.          2.034236e-03
## EVO_GAR     -1.565881e-02
## EVD_GAR      .
```

```
## PPO_GAR     -1.367332e-03
## SHD_GAR      8.852440e-03
## Take_GAR.x   1.652513e-01
## Draw_GAR.x   .
## Off_GAR     -5.323708e-02
## Def_GAR      2.506834e-02
## Pens_GAR.x   6.625737e-02
## GAR          .
## WAR          .
## SPAR         .
## xEVO_GAR     .
## xEVD_GAR     .
## xPPO_GAR     1.570033e-04
## xSHD_GAR    -2.190758e-02
## xOff_GAR     .
## xDef_GAR     .
## xGAR         8.085269e-03
## xWAR         1.026210e-02
## xSPAR        .
## G..60        .
## xG..60      -4.492590e-06
## C..60        .
## GF.60        2.993436e-02
## GA.60        2.528397e-03
## xGF.60       .
## xGA.60       4.498705e-02
## CF.60       -2.412381e-02
## CA.60       -3.127672e-02
```