P8106 - Midterm Project

Mingkuan Xu (mx2262)

3/27/2022

```
library(tidyverse)
library(corrplot)
library(caret)
library(mgcv)
```

Part 0 - Introduction

In this project, we aimed to predict the salary of NBA players in the 2021-2022 season based on their game statistics.

Part 1 - Data Preprocessig

```
df_salary = read_csv("NBA_season2122_player_salary.csv") %>%
  janitor::clean_names() %>%
  select(Player=x2,Team=x3,Salary=salary_4) %>%
  na.omit()
```

```
## New names:
## * `` -> ...1
## * `` -> ...2
## * `` -> ...3
## * Salary -> Salary...4
## * Salary -> Salary...5
## * ...

## Rows: 578 Columns: 11

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (11): ...1, ...2, ...3, Salary...4, Salary...5, Salary...6, Salary...7, ...

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
df_salary = df_salary[-1,]

df_stats = read_csv("NBA_season2122_player_stats.csv") %>%
  rename(Team=Tm) %>%
  select(-Rk)
```

```
## Rows: 784 Columns: 30

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (3): Player, Pos, Tm
## dbl (27): Rk, Age, G, GS, MP, FG, FGA, FG%, 3P, 3PA, 3P%, 2P, 2PA, 2P%, eFG%...

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

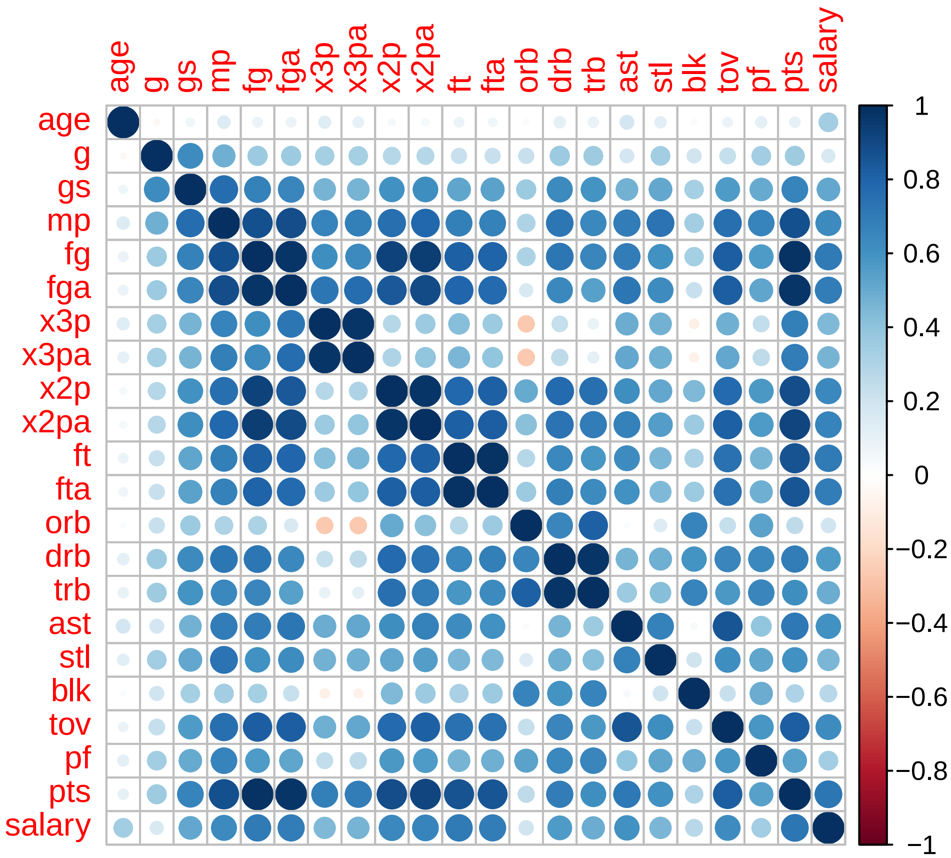
```r
df_players = inner_join(x=df_salary,y=df_stats,by=c("Player","Team")) %>%
  separate(col = Salary,sep=1,into=c("Dollar","Salary")) %>% select(-Dollar) %>%
  mutate(Salary=as.numeric(Salary)/1000000) %>%
  distinct() %>%
  relocate(Salary,.after = last_col())
  # Remove dollar sign

df_players = df_players %>%
  select(-"FG%",-"3P%",-"eFG%",-"FT%",-"2P%")

## Keep largest number of games for the same player
df_players = df_players %>%
  arrange(Player,desc(G)) %>%
  distinct(Player,.keep_all = TRUE)

df_players = df_players %>%
  janitor::clean_names() %>%
  mutate(team=factor(team),
         pos=factor(pos)) %>%
  select(-player)
```

Part 2 - Training/Test Set Splitting

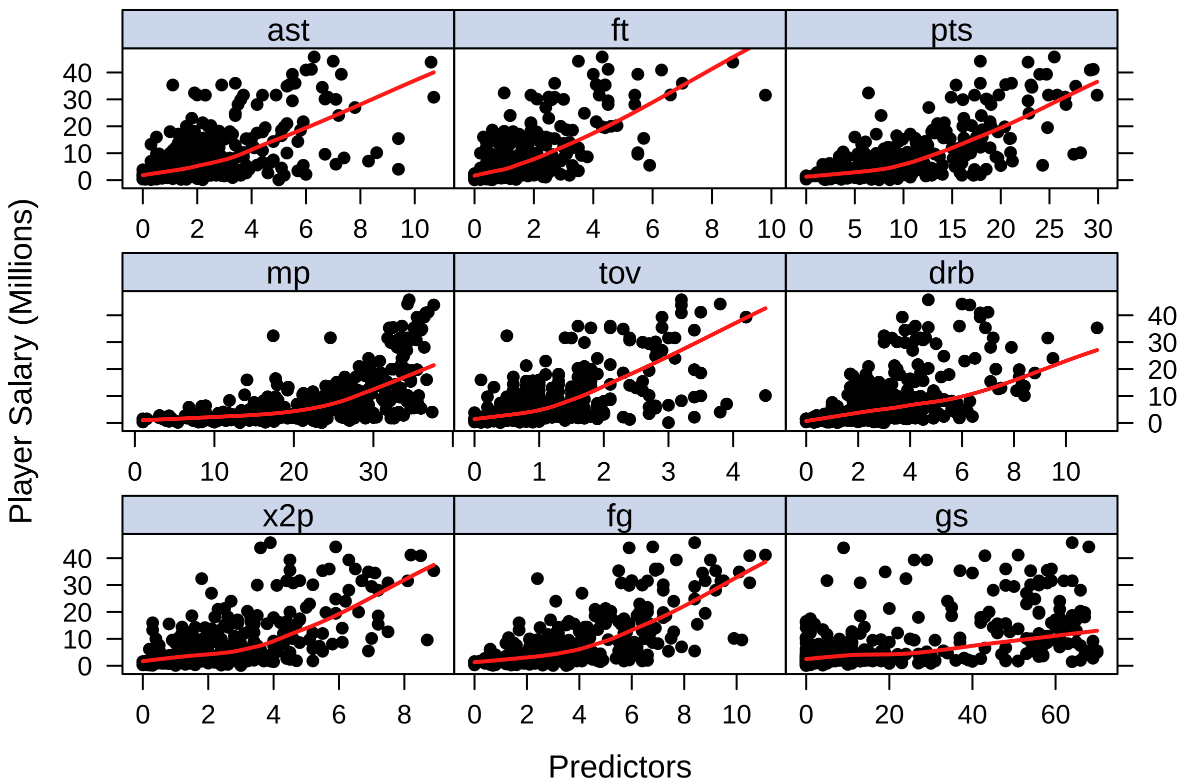
```r
index_train <- createDataPartition(y = df_players$salary, p = 0.8, list = FALSE)
df_train <- df_players[index_train, ]
df_test <- df_players[-index_train, ]
```

Part 3 - Exploratory Analysis

```r
corrplot(cor(df_train %>% select(-team,-pos)),
         method = "circle",
         type = "full")
```




```
#df_train = df_train %>%
#  mutate(gs_rate = as.numeric(gs)/as.numeric(g)) %>%
#  relocate()
theme1 <- trellis.par.get()
theme1$plot.symbol$col <- rgb(0, 0, 0, 1)
theme1$plot.symbol$pch <- 16
theme1$plot.line$col <- rgb(1, .1, .1, 1)
theme1$plot.line$lwd <- 2
theme1$strip.background$col <- rgb(.0, .2, .6, .2)
trellis.par.set(theme1)

df_features = df_train %>%
  select(x2p,fg,gs,mp,tov,drb,ast,ft,pts)
featurePlot(x = df_features,
            y = df_train$salary,
            plot = "scatter",
            # span = .5,
            labels = c("Predictors","Player Salary (Millions)"),
            type = c("p", "smooth"),
            layout = c(3, 3))
```



Part 4 - Linear/Lasso/Ridge Regression

```
set.seed(8106)

df_train_2 = model.matrix(salary ~ ., df_train)[ ,-1]
df_test_2 = model.matrix(salary ~ .,df_test)[ ,-1]
x = df_train_2
y = df_train %>% pull(salary)

ctrl1 <- trainControl(method = "repeatedcv", number = 10, repeats = 5)

# Least Square
lm.fit <- train(x, y, method = "lm", trControl = ctrl1)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.681   -3.026    0.182    2.699   20.822
##
```

```
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8.763582   3.174274  -2.761  0.00612 **
## teamBOS     -1.859639   2.515989  -0.739  0.46041
## teamBRK     -0.888042   2.542817  -0.349  0.72716
## teamCHI     -1.500734   2.359524  -0.636  0.52524
## teamCHO     -4.308349   2.388247  -1.804  0.07224 .
## teamCLE     -3.428315   2.347178  -1.461  0.14517
## teamDAL     -2.728291   2.365861  -1.153  0.24975
## teamDEN     -4.337680   2.477102  -1.751  0.08095 .
## teamDET     -3.228586   2.433057  -1.327  0.18553
## teamGSW     -1.612977   2.384695  -0.676  0.49932
## teamHOU     -4.542837   2.383238  -1.906  0.05759 .
## teamIND     -5.652772   2.763139  -2.046  0.04165 *
## teamLAC     -2.868240   2.466758  -1.163  0.24586
## teamLAL     -1.829367   2.492712  -0.734  0.46359
## teamMEM     -3.773123   2.302286  -1.639  0.10229
## teamMIA     -2.630567   2.468036  -1.066  0.28735
## teamMIL      0.085825   2.834221   0.030  0.97586
## teamMIN     -0.396356   2.370710  -0.167  0.86733
## teamNOP     -1.761291   2.514076  -0.701  0.48412
## teamNYK     -2.735163   2.469688  -1.107  0.26897
## teamOKC     -5.115386   2.415535  -2.118  0.03503 *
## teamORL     -5.108309   2.722790  -1.876  0.06161 .
## teamPHI     -1.723433   2.422420  -0.711  0.47736
## teamPHO     -4.468335   2.278007  -1.962  0.05075 .
## teamPOR     -4.680084   2.410638  -1.941  0.05315 .
## teamSAC     -3.532823   2.356814  -1.499  0.13493
## teamSAS     -3.695818   2.358748  -1.567  0.11821
## teamTOR     -2.009240   2.505635  -0.802  0.42325
## teamUTA     -1.594715   2.409268  -0.662  0.50854
## teamWAS     -2.357003   2.308977  -1.021  0.30817
## posPF        0.417614   1.184850   0.352  0.72474
## posPG       -2.538762   1.698466  -1.495  0.13604
## posSF        0.207331   1.351181   0.153  0.87815
## posSG        0.004325   1.498536   0.003  0.99770
## age          0.526646   0.083050   6.341 8.39e-10 ***
## g           -0.059799   0.021256  -2.813  0.00523 **
## gs           0.063506   0.023967   2.650  0.00848 **
## mp          -0.073266   0.131576  -0.557  0.57806
## fg          14.259029  10.320758   1.382  0.16813
## fga          6.455177   6.623106   0.975  0.33052
## x3p          1.588656   9.043065   0.176  0.86067
## x3pa        -5.250468   6.695088  -0.784  0.43353
## x2p         -2.362143   7.225997  -0.327  0.74398
## x2pa        -6.008975   6.660559  -0.902  0.36769
## ft           8.192751   4.538826   1.805  0.07207 .
## fta         -0.335058   1.216639  -0.275  0.78320
## orb          5.691813   6.350036   0.896  0.37079
## drb          6.481094   6.390224   1.014  0.31130
## trb         -5.969031   6.352467  -0.940  0.34816
## ast          1.248042   0.449512   2.776  0.00584 **
## stl          0.443869   1.291519   0.344  0.73133
## blk          2.427148   1.181150   2.055  0.04076 *
```

```
## tov         -0.666176   1.055240  -0.631  0.52832
## pf          -1.872416   0.685312  -2.732  0.00667 **
## pts         -5.792300   4.462626  -1.298  0.19530
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.639 on 299 degrees of freedom
## Multiple R-squared:  0.707,  Adjusted R-squared:  0.654
## F-statistic: 13.36 on 54 and 299 DF,  p-value: < 2.2e-16
```

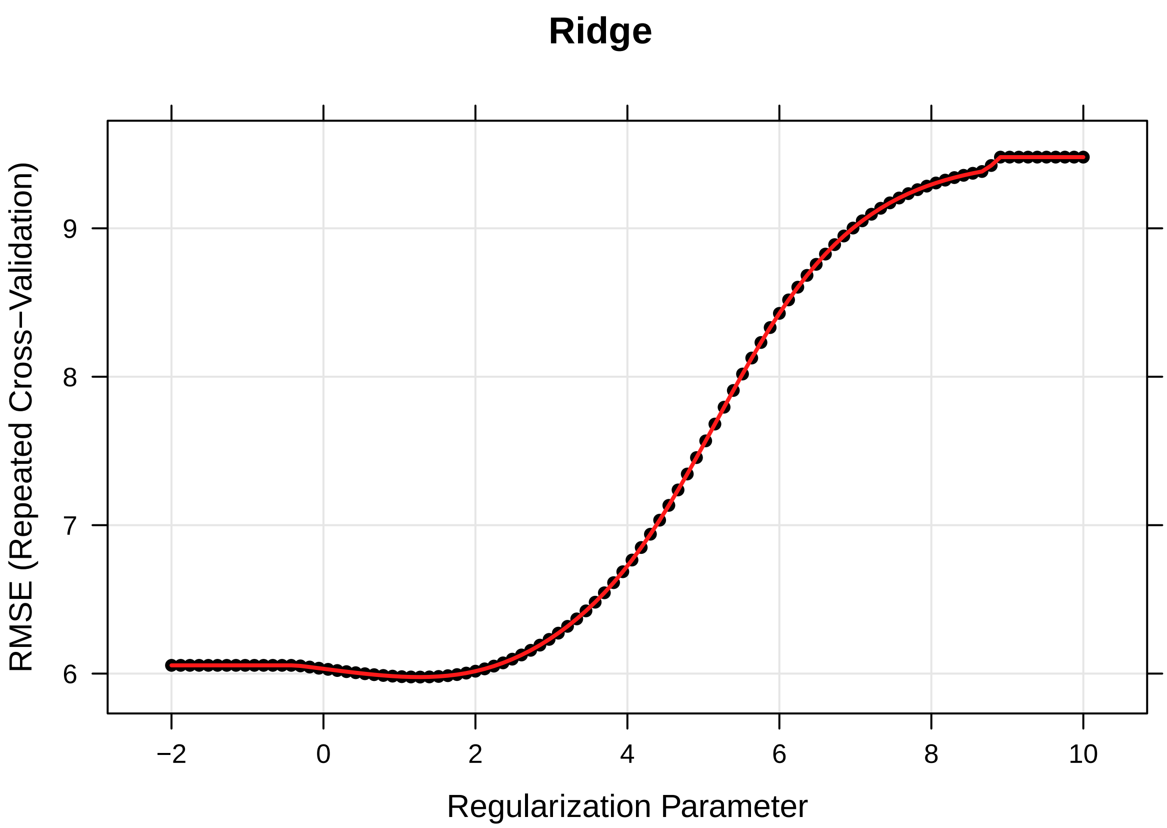
```r
lm.pred <- predict(lm.fit, newdata = df_test_2)
lm.mse = mean((lm.pred - df_test$salary)^2)
lm.mse
```

```
## [1] 40.2934
```

```r
# Ridge
ridge.fit <- train(x, y,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 0,
                                          lambda = exp(seq(10, -2, length=100))),
                   # preProc =c("center", "scale"),
                   trControl = ctrl1)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```r
plot(ridge.fit, xTrans = log, main="Ridge")
```




```
coef(ridge.fit$finalModel, ridge.fit$bestTune$lambda)
```

```
## 55 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept) -9.997068590
## teamBOS      1.006538342
## teamBRK      1.243611767
## teamCHI      0.567317422
## teamCHO     -1.208691078
## teamCLE     -0.894174625
## teamDAL     -0.100188661
## teamDEN     -1.431264170
## teamDET     -1.011538868
## teamGSW      0.938601990
## teamHOU     -2.178478960
## teamIND     -1.911889053
## teamLAC     -0.008729488
## teamLAL      0.757654460
## teamMEM     -0.870957003
## teamMIA      0.645557618
## teamMIL      2.324776714
## teamMIN      1.485379847
## teamNOP      0.456477471
## teamNYK     -0.159403348
## teamOKC     -2.409333050
```

```
## teamORL     -2.099122157
## teamPHI      1.330039033
## teamPHO     -1.010049064
## teamPOR     -2.065357186
## teamSAC     -0.586281955
## teamSAS     -0.694311651
## teamTOR      0.423821840
## teamUTA      0.850946675
## teamWAS      0.729172711
## posPF        0.520154447
## posPG       -0.889521485
## posSF        0.110079918
## posSG        0.163368929
## age          0.383461728
## g           -0.035158354
## gs           0.027942427
## mp           0.025179399
## fg           0.212764006
## fga          0.118134159
## x3p          0.328900226
## x3pa         0.173892625
## x2p          0.236104052
## x2pa         0.152987932
## ft           0.863525984
## fta          0.595451229
## orb         -0.289556788
## drb          0.273581236
## trb          0.122985817
## ast          0.514170784
## stl          0.371987688
## blk          1.125604699
## tov          0.350791829
## pf          -1.044962241
## pts          0.100757517
```

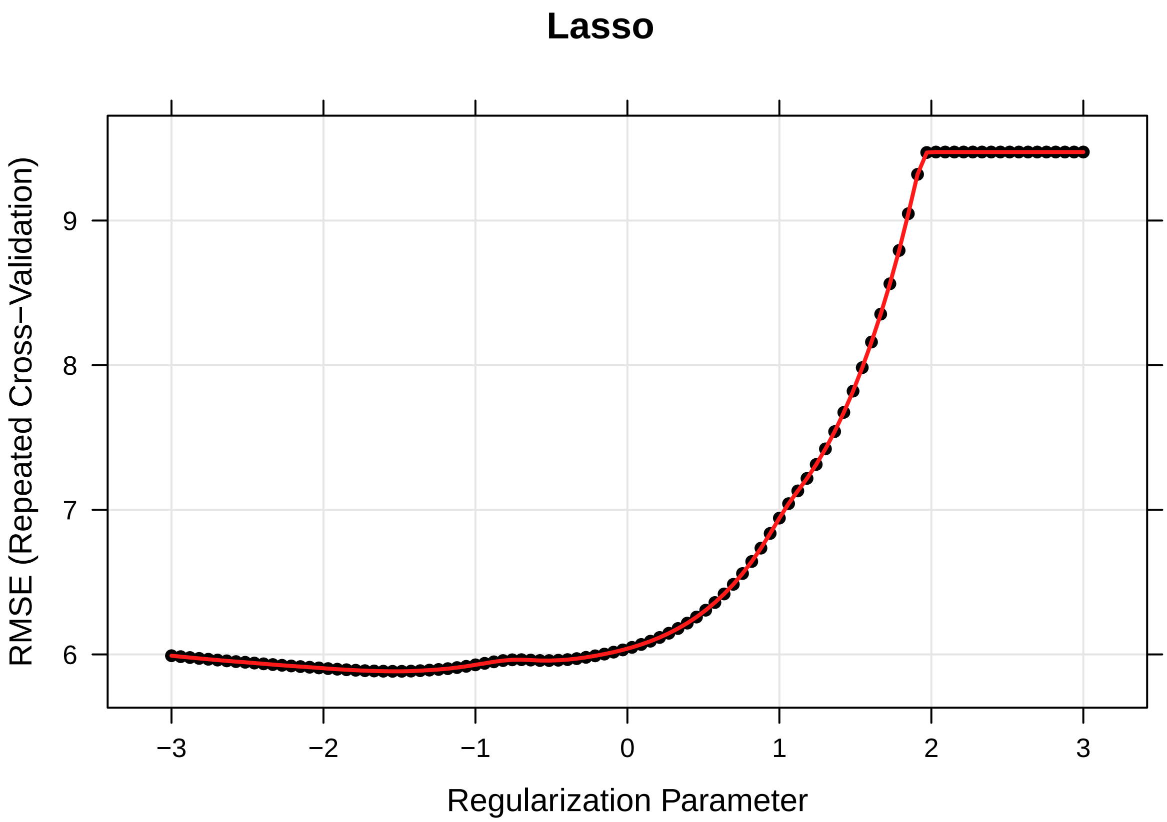
```r
ridge.pred <- predict(ridge.fit, newdata = df_test_2)
ridge.mse = mean((ridge.pred - df_test$salary)^2)
ridge.mse
```

```
## [1] 38.06952
```

```r
# Lasso
lasso.fit <- train(x, y,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 1,
                                          lambda = exp(seq(3, -3, length=100))),
                   trControl = ctrl1)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```
plot(lasso.fit, xTrans = log, main="Lasso")
```




```
coef(lasso.fit$finalModel, lasso.fit$bestTune$lambda)
```

```
## 55 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept) -12.31348666
## teamBOS       0.04331880
## teamBRK       0.66167933
## teamCHI       .
## teamCHO      -0.34183879
## teamCLE       .
## teamDAL       .
## teamDEN      -0.66081435
## teamDET       .
## teamGSW       0.02135790
## teamHOU      -1.63908563
## teamIND      -1.00356418
## teamLAC       .
## teamLAL       .
## teamMEM       .
## teamMIA       .
## teamMIL       1.30638360
## teamMIN       1.06564456
## teamNOP       .
```

```
## teamNYK        .
## teamOKC      -1.82660482
## teamORL      -1.05331268
## teamPHI       0.09615487
## teamPHO      -0.49436720
## teamPOR      -1.17092425
## teamSAC        .
## teamSAS        .
## teamTOR        .
## teamUTA        .
## teamWAS        .
## posPF         0.01416372
## posPG        -1.58079697
## posSF          .
## posSG          .
## age           0.50973505
## g            -0.02955847
## gs            0.03842201
## mp             .
## fg             .
## fga           0.32578646
## x3p            .
## x3pa           .
## x2p            .
## x2pa           .
## ft            2.05101652
## fta            .
## orb            .
## drb           0.31029187
## trb            .
## ast           0.80370304
## stl            .
## blk           1.32699216
## tov            .
## pf           -1.26388573
## pts           0.15520798
```

```
lasso.pred <- predict(lasso.fit, newdata = df_test_2)
lasso.mse = mean((lasso.pred - df_test$salary)^2)
lasso.mse
```

```
## [1] 38.1572
```

Part 5 - Generalized Addictive Model

```
gam.fit <- gam(salary~
                 s(age)+s(g)+s(gs)+s(ft)+s(fta)+s(drb)+s(ast)+s(blk)+s(pf)+s(pts),
              data = df_train)
gam.pred = predict(gam.fit, newdata = df_test)
gam.mse = mean((gam.pred - df_test$salary)^2)
gam.mse
```

```
## [1] 50.66677
```

Part 6 - Multivariate Adaptive Regression Spline Model

```r
mars_grid <- expand.grid(degree = 1:3,
                         nprune = 2:15)

set.seed(2)
mars.fit <- train(x, y,
                  method = "earth",
                  tuneGrid = mars_grid,
                  trControl = ctrl1)
```

```
## Loading required package: earth

## Loading required package: Formula

## Loading required package: plotmo

## Loading required package: plotrix

## Loading required package: TeachingDemos
```

```r
mars.fit$bestTune
```

```
##    nprune degree
## 18      5      2
```

```r
coef(mars.fit$finalModel)
```

```
##             (Intercept)             h(pts-10.3) h(28-age) * h(pts-10.3)
##               7.0285646               1.6928466              -0.2172044
##              h(26.9-mp)  h(age-27) * h(mp-26.9)
##              -0.2750163               0.1933464
```

```r
mars.pred = predict(mars.fit, newdata = df_test_2)
mars.mse = mean((df_test$salary - mars.pred)^2)
mars.mse
```

```
## [1] 39.59242
```

```r
resamp <- resamples(list(lasso = lasso.fit, ridge = ridge.fit, lm = lm.fit))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
```

```
## Models: lasso, ridge, lm
## Number of resamples: 50
##
## MAE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lasso 2.761564 3.958169 4.296022 4.345734 4.821419 5.868841    0
## ridge 2.965247 4.088718 4.374228 4.381683 4.601151 6.003518    0
## lm    3.382046 4.150212 4.626312 4.602174 5.081193 5.687795    0
##
## RMSE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lasso 3.599492 5.259617 5.856918 5.883114 6.517518 8.167985    0
## ridge 4.053355 5.409692 5.894620 5.976771 6.424806 8.287976    0
## lm    4.343169 5.521760 6.146287 6.224871 7.117622 7.745402    0
##
## Rsquared
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lasso 0.3679520 0.5508223 0.6274519 0.6270410 0.7255002 0.8682571    0
## ridge 0.3398791 0.5477482 0.6154440 0.6101063 0.6784948 0.8262507    0
## lm    0.2930701 0.5203138 0.6025807 0.5936558 0.6828017 0.8138288    0
```

```r
test_RMSE <- data.frame (
  Methods = c("Lease-Squared", "Lasso", "Rigde","GAM","MARS"),
  Test_MSE = c(lm.mse,lasso.mse,ridge.mse,gam.mse,mars.mse)
) %>%
  mutate(RMSE=round(sqrt(Test_MSE),digit=2)) %>%
  select(-Test_MSE)

test_RMSE %>% knitr::kable()
```

Methods	RMSE
Lease-Squared	6.35
Lasso	6.18
Rigde	6.17
GAM	7.12
MARS	6.29