Models and Deployment

Basketball Salaries Team

Load Primary Dataset

```r
df_p.all <- read.csv('data/pooled/primary.csv')
df_p.all$year <- as.factor(df_p.all$year)
str(df_p.all)
```

```
## 'data.frame':    734 obs. of  51 variables:
##  $ name  : Factor w/ 439 levels "aaron brooks",..: 1 1 2 2 3 4 5 5 6 6 ...
##  $ year  : Factor w/ 2 levels "2016","2017": 1 2 1 2 1 2 1 2 1 2 ...
##  $ salary: num  2700000 2116955 4351320 5504420 2022240 ...
##  $ Pos   : Factor w/ 5 levels "C","PF","PG",..: 3 3 2 4 2 1 4 4 1 1 ...
##  $ Age   : int  31 32 20 21 24 24 25 26 29 30 ...
##  $ Tm    : Factor w/ 31 levels "ATL","BOS","BRK",..: 4 12 22 22 18 7 25 25 1 2 ...
##  $ G     : int  69 65 78 80 52 22 82 61 82 68 ...
##  $ GS    : int  0 0 37 72 2 0 82 25 82 68 ...
##  $ MP    : int  1108 894 1863 2298 486 163 2341 1773 2631 2193 ...
##  $ PER   : num  11.8 9.5 17 14.4 5.6 8.4 12.7 11.3 19.4 17.7 ...
##  $ TS.   : num  0.494 0.507 0.541 0.53 0.422 0.472 0.533 0.506 0.565 0.553 ...
##  $ X3PAr : num  0.394 0.427 0.245 0.309 0.221 0.238 0.485 0.455 0.244 0.302 ...
##  $ FTr   : num  0.136 0.133 0.333 0.251 0.179 0.476 0.217 0.292 0.123 0.169 ...
##  $ ORB.  : num  2 2.3 9 5.3 4.8 5.4 4.5 4.8 6.3 4.9 ...
##  $ DRB.  : num  7.5 6.3 21.3 14.1 21.5 20.9 18.6 23.5 18.2 18.6 ...
##  $ TRB.  : num  4.8 4.3 15.1 9.6 13.3 12.8 11.5 14.1 12.4 11.8 ...
##  $ AST.  : num  26 20.7 10.3 10.5 8.9 3.8 8.8 7.9 16.7 24.4 ...
##  $ STL.  : num  1.4 1.4 1.6 1.4 1.7 0.3 1.5 1.7 1.3 1.2 ...
##  $ BLK.  : num  0.7 0.9 2.4 1.4 1.8 7.2 1.8 2 3.6 3.3 ...
##  $ TOV.  : num  14.2 17.2 9 8.5 18.7 16.4 13.2 15.2 8.8 11.9 ...
##  $ USG.  : num  22.9 19.2 17.3 20.1 17.7 17.6 16.9 15.4 20.6 19.8 ...
##  $ OWS   : num  0.2 -0.2 3.2 2 -0.9 -0.2 1.7 -0.1 4.9 3.6 ...
##  $ DWS   : num  0.7 0.5 2.2 1.7 0.4 0.2 2.3 2 4.5 2.7 ...
##  $ WS    : num  0.9 0.3 5.4 3.7 -0.5 0 4 1.9 9.4 6.3 ...
##  $ WS.48 : num  0.04 0.016 0.139 0.076 -0.047 -0.001 0.082 0.051 0.172 0.137 ...
##  $ OBPM  : num  -0.5 -2.1 0.6 -0.2 -5.9 -7.5 -0.4 -2.3 1.5 1 ...
##  $ DBPM  : num  -2.8 -2.6 1.2 -0.4 -0.2 1.9 0.7 1.2 2.6 2.1 ...
##  $ BPM   : num  -3.3 -4.6 1.8 -0.7 -6.1 -5.6 0.2 -1.1 4.1 3.1 ...
##  $ VORP  : num  -0.4 -0.6 1.8 0.8 -0.5 -0.1 1.3 0.4 4.1 2.8 ...
##  $ FG    : int  188 121 274 393 53 17 299 183 529 379 ...
##  $ FGA   : int  469 300 579 865 145 42 719 466 1048 801 ...
##  $ FG.   : num  0.401 0.403 0.473 0.454 0.366 0.405 0.416 0.393 0.505 0.473 ...
##  $ X3P   : int  66 48 42 77 9 5 126 70 88 86 ...
##  $ X3PA  : int  185 128 142 267 32 10 349 212 256 242 ...
##  $ X3P.  : num  0.357 0.375 0.296 0.288 0.281 0.5 0.361 0.33 0.344 0.355 ...
##  $ X2P   : int  122 73 232 316 44 12 173 113 441 293 ...
##  $ X2PA  : int  284 172 437 598 113 32 370 254 792 559 ...
##  $ X2P.  : num  0.43 0.424 0.531 0.528 0.389 0.375 0.468 0.445 0.557 0.524 ...
##  $ eFG.  : num  0.471 0.483 0.509 0.499 0.397 0.464 0.503 0.468 0.547 0.527 ...
##  $ FT    : int  49 32 129 156 17 9 115 96 103 108 ...
##  $ FTA   : int  64 40 193 217 26 20 156 136 129 135 ...
##  $ FT.   : num  0.766 0.8 0.668 0.719 0.654 0.45 0.737 0.706 0.798 0.8 ...
##  $ ORB   : int  21 18 154 116 20 8 98 77 148 95 ...
##  $ DRB   : int  80 51 353 289 91 28 401 374 448 369 ...
##  $ TRB   : int  101 69 507 405 111 36 499 451 596 464 ...
##  $ AST   : int  180 125 128 150 29 4 138 99 263 337 ...
```

```
##  $ STL   : int  30 25 59 64 16 1 72 60 68 52 ...
##  $ BLK   : int  10 9 55 40 11 13 53 44 121 87 ...
##  $ TOV   : int  82 66 66 89 36 10 120 94 107 116 ...
##  $ PF    : int  132 93 153 172 77 21 171 102 163 138 ...
##  $ PTS   : int  491 322 719 1019 132 48 839 532 1249 952 ...
```

```r
head(df_p.all)
```

```
##             name year  salary Pos Age  Tm  G GS   MP  PER   TS. X3PAr   FTr ORB.
## 1  aaron brooks 2016 2700000  PG  31 CHI 69  0 1108 11.8 0.494 0.394 0.136  2.0
## 2  aaron brooks 2017 2116955  PG  32 IND 65  0  894  9.5 0.507 0.427 0.133  2.3
## 3  aaron gordon 2016 4351320  PF  20 ORL 78 37 1863 17.0 0.541 0.245 0.333  9.0
## 4  aaron gordon 2017 5504420  SF  21 ORL 80 72 2298 14.4 0.530 0.309 0.251  5.3
## 5 adreian payne 2016 2022240  PF  24 MIN 52  2  486  5.6 0.422 0.221 0.179  4.8
## 6    aj hammons 2017 1312611   C  24 DAL 22  0  163  8.4 0.472 0.238 0.476  5.4
##   DRB. TRB. AST. STL. BLK. TOV. USG.  OWS DWS   WS  WS.48 OBPM DBPM  BPM VORP
## 1  7.5  4.8 26.0  1.4  0.7 14.2 22.9  0.2 0.7  0.9  0.040 -0.5 -2.8 -3.3 -0.4
## 2  6.3  4.3 20.7  1.4  0.9 17.2 19.2 -0.2 0.5  0.3  0.016 -2.1 -2.6 -4.6 -0.6
## 3 21.3 15.1 10.3  1.6  2.4  9.0 17.3  3.2 2.2  5.4  0.139  0.6  1.2  1.8  1.8
## 4 14.1  9.6 10.5  1.4  1.4  8.5 20.1  2.0 1.7  3.7  0.076 -0.2 -0.4 -0.7  0.8
## 5 21.5 13.3  8.9  1.7  1.8 18.7 17.7 -0.9 0.4 -0.5 -0.047 -5.9 -0.2 -6.1 -0.5
## 6 20.9 12.8  3.8  0.3  7.2 16.4 17.6 -0.2 0.2  0.0 -0.001 -7.5  1.9 -5.6 -0.1
##    FG FGA   FG. X3P X3PA  X3P. X2P X2PA  X2P.  eFG.  FT FTA   FT. ORB DRB TRB
## 1 188 469 0.401  66  185 0.357 122  284 0.430 0.471  49  64 0.766  21  80 101
## 2 121 300 0.403  48  128 0.375  73  172 0.424 0.483  32  40 0.800  18  51  69
## 3 274 579 0.473  42  142 0.296 232  437 0.531 0.509 129 193 0.668 154 353 507
## 4 393 865 0.454  77  267 0.288 316  598 0.528 0.499 156 217 0.719 116 289 405
## 5  53 145 0.366   9   32 0.281  44  113 0.389 0.397  17  26 0.654  20  91 111
## 6  17  42 0.405   5   10 0.500  12   32 0.375 0.464   9  20 0.450   8  28  36
##   AST STL BLK TOV  PF  PTS
## 1 180  30  10  82 132  491
## 2 125  25   9  66  93  322
## 3 128  59  55  66 153  719
## 4 150  64  40  89 172 1019
## 5  29  16  11  36  77  132
## 6   4   1  13  10  21   48
```

Load Complete (Primary + Secondary) Dataset

```r
df_c.all <- read.csv('data/pooled/complete.csv')
df_c.all$year <- as.factor(df_c.all$year)
str(df_c.all)
```

```
## 'data.frame':    734 obs. of  58 variables:
##  $ name  : Factor w/ 439 levels "aaron brooks",..: 1 1 2 2 3 4 5 5 6 6 ...
##  $ year  : Factor w/ 2 levels "2016","2017": 1 2 1 2 1 2 1 2 1 2 ...
##  $ salary: num  2700000 2116955 4351320 5504420 2022240 ...
##  $ Pos   : Factor w/ 5 levels "C","PF","PG",..: 3 3 2 4 2 1 4 4 1 1 ...
##  $ Age   : int  31 32 20 21 24 24 25 26 29 30 ...
##  $ Tm    : Factor w/ 31 levels "ATL","BOS","BRK",..: 4 12 22 22 18 7 25 25 1 2 ...
##  $ G     : int  69 65 78 80 52 22 82 61 82 68 ...
##  $ GS    : int  0 0 37 72 2 0 82 25 82 68 ...
##  $ MP    : int  1108 894 1863 2298 486 163 2341 1773 2631 2193 ...
##  $ PER   : num  11.8 9.5 17 14.4 5.6 8.4 12.7 11.3 19.4 17.7 ...
##  $ TS.   : num  0.494 0.507 0.541 0.53 0.422 0.472 0.533 0.506 0.565 0.553 ...
##  $ X3PAr : num  0.394 0.427 0.245 0.309 0.221 0.238 0.485 0.455 0.244 0.302 ...
##  $ FTr   : num  0.136 0.133 0.333 0.251 0.179 0.476 0.217 0.292 0.123 0.169 ...
##  $ ORB.  : num  2 2.3 9 5.3 4.8 5.4 4.5 4.8 6.3 4.9 ...
##  $ DRB.  : num  7.5 6.3 21.3 14.1 21.5 20.9 18.6 23.5 18.2 18.6 ...
##  $ TRB.  : num  4.8 4.3 15.1 9.6 13.3 12.8 11.5 14.1 12.4 11.8 ...
##  $ AST.  : num  26 20.7 10.3 10.5 8.9 3.8 8.8 7.9 16.7 24.4 ...
##  $ STL.  : num  1.4 1.4 1.6 1.4 1.7 0.3 1.5 1.7 1.3 1.2 ...
##  $ BLK.  : num  0.7 0.9 2.4 1.4 1.8 7.2 1.8 2 3.6 3.3 ...
```

```
## $ TOV.  : num  14.2 17.2 9 8.5 18.7 16.4 13.2 15.2 8.8 11.9 ...
## $ USG.  : num  22.9 19.2 17.3 20.1 17.7 17.6 16.9 15.4 20.6 19.8 ...
## $ OWS   : num  0.2 -0.2 3.2 2 -0.9 -0.2 1.7 -0.1 4.9 3.6 ...
## $ DWS   : num  0.7 0.5 2.2 1.7 0.4 0.2 2.3 2 4.5 2.7 ...
## $ WS    : num  0.9 0.3 5.4 3.7 -0.5 0 4 1.9 9.4 6.3 ...
## $ WS.48 : num  0.04 0.016 0.139 0.076 -0.047 -0.001 0.082 0.051 0.172 0.137 ...
## $ OBPM  : num  -0.5 -2.1 0.6 -0.2 -5.9 -7.5 -0.4 -2.3 1.5 1 ...
## $ DBPM  : num  -2.8 -2.6 1.2 -0.4 -0.2 1.9 0.7 1.2 2.6 2.1 ...
## $ BPM   : num  -3.3 -4.6 1.8 -0.7 -6.1 -5.6 0.2 -1.1 4.1 3.1 ...
## $ VORP  : num  -0.4 -0.6 1.8 0.8 -0.5 -0.1 1.3 0.4 4.1 2.8 ...
## $ FG    : int  188 121 274 393 53 17 299 183 529 379 ...
## $ FGA   : int  469 300 579 865 145 42 719 466 1048 801 ...
## $ FG.   : num  0.401 0.403 0.473 0.454 0.366 0.405 0.416 0.393 0.505 0.473 ...
## $ X3P   : int  66 48 42 77 9 5 126 70 88 86 ...
## $ X3PA  : int  185 128 142 267 32 10 349 212 256 242 ...
## $ X3P.  : num  0.357 0.375 0.296 0.288 0.281 0.5 0.361 0.33 0.344 0.355 ...
## $ X2P   : int  122 73 232 316 44 12 173 113 441 293 ...
## $ X2PA  : int  284 172 437 598 113 32 370 254 792 559 ...
## $ X2P.  : num  0.43 0.424 0.531 0.528 0.389 0.375 0.468 0.445 0.557 0.524 ...
## $ eFG.  : num  0.471 0.483 0.509 0.499 0.397 0.464 0.503 0.468 0.547 0.527 ...
## $ FT    : int  49 32 129 156 17 9 115 96 103 108 ...
## $ FTA   : int  64 40 193 217 26 20 156 136 129 135 ...
## $ FT.   : num  0.766 0.8 0.668 0.719 0.654 0.45 0.737 0.706 0.798 0.8 ...
## $ ORB   : int  21 18 154 116 20 8 98 77 148 95 ...
## $ DRB   : int  80 51 353 289 91 28 401 374 448 369 ...
## $ TRB   : int  101 69 507 405 111 36 499 451 596 464 ...
## $ AST   : int  180 125 128 150 29 4 138 99 263 337 ...
## $ STL   : int  30 25 59 64 16 1 72 60 68 52 ...
## $ BLK   : int  10 9 55 40 11 13 53 44 121 87 ...
## $ TOV   : int  82 66 66 89 36 10 120 94 107 116 ...
## $ PF    : int  132 93 153 172 77 21 171 102 163 138 ...
## $ PTS   : int  491 322 719 1019 132 48 839 532 1249 952 ...
## $ out   : int  79 87 87 86 56 47 90 75 81 80 ...
## $ ovr   : int  75 85 90 92 69 66 91 83 83 91 ...
## $ ins   : int  52 51 91 91 65 64 77 72 76 82 ...
## $ pla   : int  74 81 69 49 43 40 60 59 58 82 ...
## $ ath   : int  77 82 86 86 66 58 81 75 75 77 ...
## $ def   : int  52 57 69 75 64 57 76 66 70 80 ...
## $ reb   : int  36 37 87 94 68 71 94 65 73 87 ...
```

```r
head(df_c.all)
```

```
##             name year  salary Pos Age  Tm  G GS   MP  PER   TS. X3PAr   FTr ORB.
## 1  aaron brooks 2016 2700000  PG  31 CHI 69  0 1108 11.8 0.494 0.394 0.136  2.0
## 2  aaron brooks 2017 2116955  PG  32 IND 65  0  894  9.5 0.507 0.427 0.133  2.3
## 3  aaron gordon 2016 4351320  PF  20 ORL 78 37 1863 17.0 0.541 0.245 0.333  9.0
## 4  aaron gordon 2017 5504420  SF  21 ORL 80 72 2298 14.4 0.530 0.309 0.251  5.3
## 5 adreian payne 2016 2022240  PF  24 MIN 52  2  486  5.6 0.422 0.221 0.179  4.8
## 6    aj hammons 2017 1312611   C  24 DAL 22  0  163  8.4 0.472 0.238 0.476  5.4
##   DRB. TRB. AST. STL. BLK. TOV. USG.  OWS DWS   WS  WS.48 OBPM DBPM  BPM VORP
## 1  7.5  4.8 26.0  1.4  0.7 14.2 22.9  0.2 0.7  0.9  0.040 -0.5 -2.8 -3.3 -0.4
## 2  6.3  4.3 20.7  1.4  0.9 17.2 19.2 -0.2 0.5  0.3  0.016 -2.1 -2.6 -4.6 -0.6
## 3 21.3 15.1 10.3  1.6  2.4  9.0 17.3  3.2 2.2  5.4  0.139  0.6  1.2  1.8  1.8
## 4 14.1  9.6 10.5  1.4  1.4  8.5 20.1  2.0 1.7  3.7  0.076 -0.2 -0.4 -0.7  0.8
## 5 21.5 13.3  8.9  1.7  1.8 18.7 17.7 -0.9 0.4 -0.5 -0.047 -5.9 -0.2 -6.1 -0.5
## 6 20.9 12.8  3.8  0.3  7.2 16.4 17.6 -0.2 0.2  0.0 -0.001 -7.5  1.9 -5.6 -0.1
##    FG FGA   FG. X3P X3PA  X3P. X2P X2PA  X2P.  eFG.  FT FTA   FT. ORB DRB TRB
## 1 188 469 0.401  66  185 0.357 122  284 0.430 0.471  49  64 0.766  21  80 101
## 2 121 300 0.403  48  128 0.375  73  172 0.424 0.483  32  40 0.800  18  51  69
## 3 274 579 0.473  42  142 0.296 232  437 0.531 0.509 129 193 0.668 154 353 507
## 4 393 865 0.454  77  267 0.288 316  598 0.528 0.499 156 217 0.719 116 289 405
## 5  53 145 0.366   9   32 0.281  44  113 0.389 0.397  17  26 0.654  20  91 111
## 6  17  42 0.405   5   10 0.500  12   32 0.375 0.464   9  20 0.450   8  28  36
##   AST STL BLK TOV  PF  PTS out ovr ins pla ath def reb
```

```
## 1 180  30  10  82 132  491  79  75  52  74  77  52  36
## 2 125  25   9  66  93  322  87  85  51  81  82  57  37
## 3 128  59  55  66 153  719  87  90  91  69  86  69  87
## 4 150  64  40  89 172 1019  86  92  91  49  86  75  94
## 5  29  16  11  36  77  132  56  69  65  43  66  64  68
## 6   4   1  13  10  21   48  47  66  64  40  58  57  71
```

Split Primary & Complete Datasets into Train Test

```r
library(caret)
set.seed(7)
# primary dataset
train_rows.p <- createDataPartition(y=df_p.all[,'salary'], list=FALSE, p=.8)
df_p.train <- df_p.all[train_rows.p,]
df_p.test <- df_p.all[-train_rows.p,]
nrow(df_p.all)
```

```
## [1] 734
```

```r
nrow(df_p.train)
```

```
## [1] 590
```

```r
nrow(df_p.test)
```

```
## [1] 144
```

```r
# complete dataset
train_rows.c <- createDataPartition(y=df_c.all[,'salary'], list=FALSE, p=.8)
df_c.train <- df_c.all[train_rows.c,]
df_c.test <- df_c.all[-train_rows.c,]
nrow(df_c.all)
```

```
## [1] 734
```

```r
nrow(df_c.train)
```

```
## [1] 590
```

```r
nrow(df_c.test)
```

```
## [1] 144
```

Modeling Helper Functions

```r
r_squared <- function(y,yHat){1-sum((y-yHat)^2)/sum((y-mean(y))^2)}
mse <- function(y,yHat){mean((y-yHat)^2)}
model_results <- function(model,dataset,y,yHat){
  r2_test <- r_squared(y,yHat)
  mse_test <- mse(y,yHat)
  cat(sprintf('Model: %-25s Dataset: %-10s R^2 Test: %-10.3f MSE: %-10.3e\n',model,dataset,r2_test,mse_test))}
```

Simple Linear Regression Models

```r
# modeling function
slr_modeling <- function(dataset,df_train,df_test){
  model <- 'simple linear regression'
  x_vars <- names(df_train)[!(names(df_train)%in%c('name','salary','X2P','X2PA','TRB','PTS'))]
  f <- as.formula(sprintf('salary ~ `%s`',paste(x_vars,collapse='` + `')))
  slr_model <- lm(f,data=df_train)
  yhat <- predict(slr_model,df_test)
  model_results(model,dataset,df_test[['salary']],yhat)
  return(slr_model)}
```

```r
# train/test Simple Linear Regression models
ignore <- slr_modeling('primary',df_p.train,df_p.test)
```

```
## Model: simple linear regression  Dataset: primary    R^2 Test: 0.464      MSE: 2.668e+13
```

```r
ignore <- slr_modeling('complete',df_c.train,df_c.test)
```

```
## Model: simple linear regression  Dataset: complete   R^2 Test: 0.469      MSE: 2.575e+13
```

Lasso, Ridge, and Elastic Net Models with 10-fold Cross validation for alpha = seq(0,1,by=.1)

```r
library(glmnet)
# modeling function
lre_modeling <- function(dataset,x_train,y_train,x_test,y_test,alphas,mkplot){
  set.seed(7) # seed for reproducibility
  # fit models
  for (i in alphas){
    model_name <- paste('fit_alpha_',i,sep='')
    assign(model_name, cv.glmnet(x_train, y_train, type.measure="mse",alpha=i,family="gaussian"))
    model <- get(model_name)
    yhat <- predict(model,s=model$lambda.min,newx=x_test)
    model_results(model_name,dataset,y_test,yhat)}
  # plot
  if(mkplot){
    path = sprintf("figures/lasso_ridge_elasticnet_models_%s.png",dataset)
    png(file=path,width=5,height=15,units='in',res=1200)
    par(mfrow=c(6,1))
    #    lasso
    lasso_model <- glmnet(x_train, y_train, family="gaussian", alpha=1)
    lasso_model_cv <- get('fit_alpha_1')
    plot(lasso_model,main="LASSO")
    plot(lasso_model_cv,xvar="lambda")
    #    ridge
    ridge_model <- glmnet(x_train, y_train, family="gaussian", alpha=0)
    ridge_model_cv <- get('fit_alpha_0')
    plot(ridge_model,main="Ridge")
    plot(ridge_model_cv,xvar="lambda")
    #    elastic net
    enet_model <- glmnet(x_train, y_train, family="gaussian", alpha=.5)
    enet_model_cv <- get('fit_alpha_0.5')
    plot(enet_model,main="Elastic Net")
    plot(enet_model_cv,xvar="lambda")
    dev.off()}
  return(model)} # return final model created
# extract train/test datasets of only numeric variables as required by glmnet models
#    primary dataset
numeric_vars.p <- names(Filter(is.numeric,df_p.train))
numeric_x_vars.p <- numeric_vars.p[!(numeric_vars.p%in%c('salary'))]
x_train.p <- data.matrix(df_p.train[,numeric_x_vars.p])
y_train.p <- df_p.train[['salary']]
x_test.p <- data.matrix(df_p.test[,numeric_x_vars.p])
y_test.p <- df_p.test[['salary']]
#    complete dataset
numeric_vars.c <- names(Filter(is.numeric,df_c.train))
numeric_x_vars.c <- numeric_vars.c[!(numeric_vars.c%in%c('salary'))]
x_train.c <- data.matrix(df_c.train[,numeric_x_vars.c])
y_train.c <- df_c.train[['salary']]
x_test.c <- data.matrix(df_c.test[,numeric_x_vars.c])
y_test.c <- df_c.test[['salary']]
# train/test Simple Linear Regression models
mkplots <- FALSE # change to TRUE if you want to generate plots
ignore <- lre_modeling('primary',x_train.p,y_train.p,x_test.p,y_test.p,seq(0,1,by=.1),mkplots)
```

```
## Model: fit_alpha_0               Dataset: primary    R^2 Test: 0.520      MSE: 2.390e+13
## Model: fit_alpha_0.1             Dataset: primary    R^2 Test: 0.522      MSE: 2.378e+13
## Model: fit_alpha_0.2             Dataset: primary    R^2 Test: 0.517      MSE: 2.403e+13
## Model: fit_alpha_0.3             Dataset: primary    R^2 Test: 0.509      MSE: 2.442e+13
## Model: fit_alpha_0.4             Dataset: primary    R^2 Test: 0.519      MSE: 2.393e+13
## Model: fit_alpha_0.5             Dataset: primary    R^2 Test: 0.514      MSE: 2.416e+13
## Model: fit_alpha_0.6             Dataset: primary    R^2 Test: 0.517      MSE: 2.401e+13
## Model: fit_alpha_0.7             Dataset: primary    R^2 Test: 0.508      MSE: 2.449e+13
## Model: fit_alpha_0.8             Dataset: primary    R^2 Test: 0.513      MSE: 2.421e+13
## Model: fit_alpha_0.9             Dataset: primary    R^2 Test: 0.511      MSE: 2.432e+13
## Model: fit_alpha_1               Dataset: primary    R^2 Test: 0.517      MSE: 2.403e+13
```

```
ignore <- lre_modeling('complete',x_train.c,y_train.c,x_test.c,y_test.c,seq(0,1,by=.1),mkplots)
```

```
## Model: fit_alpha_0               Dataset: complete   R^2 Test: 0.490      MSE: 2.477e+13
## Model: fit_alpha_0.1             Dataset: complete   R^2 Test: 0.481      MSE: 2.517e+13
## Model: fit_alpha_0.2             Dataset: complete   R^2 Test: 0.486      MSE: 2.494e+13
## Model: fit_alpha_0.3             Dataset: complete   R^2 Test: 0.484      MSE: 2.501e+13
## Model: fit_alpha_0.4             Dataset: complete   R^2 Test: 0.482      MSE: 2.512e+13
## Model: fit_alpha_0.5             Dataset: complete   R^2 Test: 0.487      MSE: 2.488e+13
## Model: fit_alpha_0.6             Dataset: complete   R^2 Test: 0.481      MSE: 2.517e+13
## Model: fit_alpha_0.7             Dataset: complete   R^2 Test: 0.482      MSE: 2.512e+13
## Model: fit_alpha_0.8             Dataset: complete   R^2 Test: 0.482      MSE: 2.513e+13
## Model: fit_alpha_0.9             Dataset: complete   R^2 Test: 0.480      MSE: 2.523e+13
## Model: fit_alpha_1               Dataset: complete   R^2 Test: 0.480      MSE: 2.524e+13
```

Save Optimal Model to File

Optimal model with largest R^2 Test and lowest MSE is elasticnet model with alpha = .1

```
optimal_model <- lre_modeling('primary',x_train.p,y_train.p,x_test.p,y_test.p,c(0.1),FALSE)
```

```
## Model: fit_alpha_0.1             Dataset: primary    R^2 Test: 0.522      MSE: 2.378e+13
```

```
saveRDS(optimal_model,file='data/optimal_model.rds')
```

Example Deployment of Optimal Model

```
library(plumber)
r <- plumb("deploy_optimal_model.R")
r$run(port=8000)
# Single Player Prediction Command
```

```
# 2016 LeBron James Example
cmd <- paste(
          "curl -X POST 'http://localhost:8000/predict_salary?",
         "mat=0&Age=31&G=76&GS=76&MP=2709&PER=27.5&TS.=.588&X3PAr=0.199&FTr=0.347&ORB.=4.7&DRB.=18.8&",
         "TRB.=11.8&AST.=36.0&STL.=2.0&BLK.=1.5&TOV.=13.2&USG.=31.4&OWS=9.6&DWS=4&WS=13.6&WS.48=0.242&",
         "OBPM=6.9&DBPM=2.3&BPM=9.1&VORP=7.6&FG=737&FGA=1416&FG.=0.520&X3P=87&X3PA=282&X3P.=0.309&X2P=650&",
         "X2PA=1134&X2P.=0.573&eFG.=0.551&FT=359&FTA=491&FT.=0.731&ORB=111&DRB=454&TRB=565&AST=514&STL=104&",
         "BLK=49&TOV=249&PF=143&PTS=1920",
         sep='')
```