Luke Williamson

"Project: SGEMM GPU kernel performance Data Set Regression"

This is an R Markdown Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Ctrl+Shift+Enter*.

Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Ctrl+Alt+I*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Ctrl+Shift+K* to preview the HTML file).

The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike *Knit*, *Preview* does not run any R code chunks. Instead, the output of the chunk when it was last run in the editor is displayed. ##

Raw Data Exploration

-Link for the data: https://archive.ics.uci.edu/ml/datasets/SGEMM+GPU+kernel+performance

Citations for the data: - Rafael Ballester-Ripoll, Enrique G. Paredes, Renato Pajarola. Sobol Tensor Trains for Global Sensitivity Analysis. In arXiv Computer Science / Numerical Analysis e-prints, 2017 https://arxiv.org/abs/1712.00233 - Cedric Nugteren and Valeriu Codreanu. CLTune: A Generic Auto-Tuner for OpenCL Kernels. In: MCSoC: 9th International Symposium on Embedded Multicore/Many-core Systems-on-Chip. IEEE, 2015 https://ieeexplore.ieee.org/document/7328205

-***After downloading the data, be sure to change the full file path name accordingly in the chunk below.*** -Read in file to dataframe and explore raw data:

```
df <- read.csv("C:/Users/law/Documents/sgemm_product.csv", header=TRUE)
dim(df)
```

```
## [1] 241600     18
```

```
names(df)
```

```
##  [1] "MWG"       "NWG"       "KWG"       "MDIMC"     "NDIMC"
##  [6] "MDIMA"     "NDIMB"     "KWI"       "VWM"       "VWN"
## [11] "STRM"      "STRN"      "SA"        "SB"        "Run1..ms."
## [16] "Run2..ms." "Run3..ms." "Run4..ms."
```

```
str(df)
```

```
## 'data.frame':    241600 obs. of  18 variables:
##  $ MWG      : int  16 16 16 16 16 16 16 16 16 16 ...
##  $ NWG      : int  16 16 16 16 16 16 16 16 16 16 ...
##  $ KWG      : int  16 16 16 16 16 16 16 16 16 16 ...
##  $ MDIMC    : int  8 8 8 8 8 8 8 8 8 8 ...
##  $ NDIMC    : int  8 8 8 8 8 8 8 8 8 8 ...
##  $ MDIMA    : int  8 8 8 8 8 8 8 8 8 8 ...
##  $ NDIMB    : int  8 8 8 8 8 8 8 8 8 8 ...
```

```
##  $ KWI      : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ VWM      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ VWN      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ STRM     : int  0 0 0 0 0 0 0 0 1 1 ...
##  $ STRN     : int  0 0 0 0 1 1 1 1 0 0 ...
##  $ SA       : int  0 0 1 1 0 0 1 1 0 0 ...
##  $ SB       : int  0 1 0 1 0 1 0 1 0 1 ...
##  $ Run1..ms.: num  115.3 78.1 79.8 84.3 115.1 ...
##  $ Run2..ms.: num  115.9 78.2 80.7 89.9 122 ...
##  $ Run3..ms.: num  118.5 79.2 80.8 86.8 122.7 ...
##  $ Run4..ms.: num  115.8 79.2 81 85.6 114.8 ...
```

```
head(df)
```

```
##   MWG NWG KWG MDIMC NDIMC MDIMA NDIMB KWI VWM VWN STRM STRN SA SB
## 1  16  16  16     8     8     8     8   2   1   1    0    0  0  0
## 2  16  16  16     8     8     8     8   2   1   1    0    0  0  1
## 3  16  16  16     8     8     8     8   2   1   1    0    0  1  0
## 4  16  16  16     8     8     8     8   2   1   1    0    0  1  1
## 5  16  16  16     8     8     8     8   2   1   1    0    1  0  0
## 6  16  16  16     8     8     8     8   2   1   1    0    1  0  1
##   Run1..ms. Run2..ms. Run3..ms. Run4..ms.
## 1    115.26    115.87    118.55    115.80
## 2     78.13     78.25     79.25     79.19
## 3     79.84     80.69     80.76     80.97
## 4     84.32     89.90     86.75     85.58
## 5    115.13    121.98    122.73    114.81
## 6     81.10     82.41     87.01     82.14
```

```
tail(df)
```

```
##        MWG NWG KWG MDIMC NDIMC MDIMA NDIMB KWI VWM VWN STRM STRN SA SB
## 241595 128 128  32    32    32    32    32   8   4   4    1    0  1  0
## 241596 128 128  32    32    32    32    32   8   4   4    1    0  1  1
## 241597 128 128  32    32    32    32    32   8   4   4    1    1  0  0
## 241598 128 128  32    32    32    32    32   8   4   4    1    1  0  1
## 241599 128 128  32    32    32    32    32   8   4   4    1    1  1  0
## 241600 128 128  32    32    32    32    32   8   4   4    1    1  1  1
##        Run1..ms. Run2..ms. Run3..ms. Run4..ms.
## 241595     28.84     28.87     28.83     28.83
## 241596     17.96     17.77     17.77     17.77
## 241597     36.04     36.03     36.04     36.03
## 241598     35.28     34.82     35.27     35.27
## 241599     28.43     28.49     28.44     28.45
## 241600     17.94     17.79     17.77     17.77
```

```
summary(df)
```

```
##       MWG             NWG             KWG            MDIMC
##  Min.   : 16.00   Min.   : 16.00   Min.   :16.00   Min.   : 8.00
##  1st Qu.: 32.00   1st Qu.: 32.00   1st Qu.:16.00   1st Qu.: 8.00
##  Median : 64.00   Median : 64.00   Median :32.00   Median : 8.00
```

```
##  Mean   : 80.42   Mean   : 80.42   Mean   :25.51   Mean   :13.94
##  3rd Qu.:128.00   3rd Qu.:128.00   3rd Qu.:32.00   3rd Qu.:16.00
##  Max.   :128.00   Max.   :128.00   Max.   :32.00   Max.   :32.00
##      NDIMC           MDIMA           NDIMB            KWI
##  Min.   : 8.00   Min.   : 8.00   Min.   : 8.00   Min.   :2
##  1st Qu.: 8.00   1st Qu.: 8.00   1st Qu.: 8.00   1st Qu.:2
##  Median : 8.00   Median :16.00   Median :16.00   Median :5
##  Mean   :13.94   Mean   :17.37   Mean   :17.37   Mean   :5
##  3rd Qu.:16.00   3rd Qu.:32.00   3rd Qu.:32.00   3rd Qu.:8
##  Max.   :32.00   Max.   :32.00   Max.   :32.00   Max.   :8
##       VWM             VWN             STRM          STRN           SA
##  Min.   :1.000   Min.   :1.000   Min.   :0.0   Min.   :0.0   Min.   :0.0
##  1st Qu.:1.000   1st Qu.:1.000   1st Qu.:0.0   1st Qu.:0.0   1st Qu.:0.0
##  Median :2.000   Median :2.000   Median :0.5   Median :0.5   Median :0.5
##  Mean   :2.449   Mean   :2.449   Mean   :0.5   Mean   :0.5   Mean   :0.5
##  3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:1.0   3rd Qu.:1.0   3rd Qu.:1.0
##  Max.   :8.000   Max.   :8.000   Max.   :1.0   Max.   :1.0   Max.   :1.0
##        SB        Run1..ms.         Run2..ms.         Run3..ms.
##  Min.   :0.0   Min.   :  13.29   Min.   :  13.25   Min.   :  13.36
##  1st Qu.:0.0   1st Qu.:  40.66   1st Qu.:  40.71   1st Qu.:  40.66
##  Median :0.5   Median :  69.83   Median :  69.93   Median :  69.79
##  Mean   :0.5   Mean   : 217.65   Mean   : 217.58   Mean   : 217.53
##  3rd Qu.:1.0   3rd Qu.: 228.53   3rd Qu.: 228.31   3rd Qu.: 228.32
##  Max.   :1.0   Max.   :3339.63   Max.   :3375.42   Max.   :3397.08
##    Run4..ms.
##  Min.   :  13.37
##  1st Qu.:  40.64
##  Median :  69.82
##  Mean   : 217.53
##  3rd Qu.: 228.32
##  Max.   :3361.71
```

```
  sapply(df, class)
```

```
##       MWG       NWG       KWG     MDIMC     NDIMC     MDIMA     NDIMB
## "integer" "integer" "integer" "integer" "integer" "integer" "integer"
##       KWI       VWM       VWN      STRM      STRN        SA        SB
## "integer" "integer" "integer" "integer" "integer" "integer" "integer"
## Run1..ms. Run2..ms. Run3..ms. Run4..ms.
## "numeric" "numeric" "numeric" "numeric"
```

```
  sapply(df, function(x) sum(is.na(x)==TRUE))
```

```
##       MWG       NWG       KWG     MDIMC     NDIMC     MDIMA     NDIMB
##         0         0         0         0         0         0         0
##       KWI       VWM       VWN      STRM      STRN        SA        SB
##         0         0         0         0         0         0         0
## Run1..ms. Run2..ms. Run3..ms. Run4..ms.
##         0         0         0         0
```

Cleaned Data Exploration

-To clean the data, I averaged the last 4 columns (which were 4 seperate trials (not attributes)) over all instances and added the averages to a new column in the dataframe called run_avg. -I then removed columns 15:18 which contained the 4 seperate runtime trials. -I then convert columns 11:14 to factors, because we can see that in the above chunk, R interpreted them as integers. -Due to processing constraints, I took a random subsetted sample of size 10000 the dataset. Cleaned dataframe and explore new data:

```
  df$run_avg <- ((df$Run1..ms. + df$Run2..ms. + df$Run3..ms. + df$Run4..ms.) / 4)
  df[15:18] <- list(NULL)
  df$STRM <- as.factor(df$STRM)
  df$STRN <- as.factor(df$STRN)
  df$SA <- as.factor(df$SA)
  df$SB <- as.factor(df$SB)
  df <- df[sample(1:nrow(df), 10000, replace=FALSE),]
  dim(df)
```

```
## [1] 10000    15
```

```
  names(df)
```

```
##  [1] "MWG"     "NWG"     "KWG"     "MDIMC"   "NDIMC"   "MDIMA"   "NDIMB"
##  [8] "KWI"     "VWM"     "VWN"     "STRM"    "STRN"    "SA"      "SB"
## [15] "run_avg"
```

```
  str(df)
```

```
## 'data.frame':    10000 obs. of  15 variables:
##  $ MWG    : int  64 32 128 64 128 64 64 64 128 32 ...
##  $ NWG    : int  64 64 32 128 128 32 128 128 128 128 ...
##  $ KWG    : int  32 32 32 16 32 32 32 16 32 16 ...
##  $ MDIMC  : int  16 16 16 8 32 32 8 16 16 16 ...
##  $ NDIMC  : int  8 16 16 8 8 8 32 8 8 16 ...
##  $ MDIMA  : int  16 32 32 8 32 16 8 8 16 32 ...
##  $ NDIMB  : int  8 32 8 16 8 8 8 8 16 16 ...
##  $ KWI    : int  8 2 2 8 8 2 2 8 2 2 ...
##  $ VWM    : int  1 1 4 1 1 1 1 1 2 1 ...
##  $ VWN    : int  1 2 1 1 8 2 1 2 4 1 ...
##  $ STRM   : Factor w/ 2 levels "0","1": 2 1 2 2 2 2 1 1 1 1 ...
##  $ STRN   : Factor w/ 2 levels "0","1": 1 1 2 2 1 1 1 2 1 2 ...
##  $ SA     : Factor w/ 2 levels "0","1": 1 1 1 1 2 2 1 1 1 1 ...
##  $ SB     : Factor w/ 2 levels "0","1": 2 2 2 2 1 2 1 1 2 1 ...
##  $ run_avg: num  46.8 40.7 65.9 793.9 145.2 ...
```

```
  head(df)
```

```
##        MWG NWG KWG MDIMC NDIMC MDIMA NDIMB KWI VWM VWN STRM STRN SA SB
## 104138  64  64  32    16     8    16     8   8   1   1    1    0  0  1
## 45010   32  64  32    16    16    32    32   2   1   2    0    0  0  1
## 165870 128  32  32    16    16    32     8   2   4   1    1    1  0  1
## 111342  64 128  16     8     8     8    16   8   1   1    1    1  0  1
```

```
## 239163 128 128  32    32     8    32     8   8   1   8    1    0  1  0
## 85724   64  32  32    32     8    16     8   2   1   2    1    0  1  1
##         run_avg
## 104138  46.7900
## 45010   40.6975
## 165870  65.9225
## 111342 793.9475
## 239163 145.2400
## 85724   35.5275
```

```
tail(df)
```

```
##        MWG NWG KWG MDIMC NDIMC MDIMA NDIMB KWI VWM VWN STRM STRN SA SB
## 106222  64  64  32    16    16    16     8   2   2   2    1    1  0  1
## 111185  64 128  16     8     8     8    16   2   2   8    0    0  0  0
## 11210   16 128  16     8     8     8     8   2   2   1    1    0  0  1
## 176912 128  64  16    16     8     8    32   2   1   1    1    1  1  1
## 165081 128  32  32    16    16     8    32   2   8   1    1    0  0  0
## 25242   32  32  16    16     8     8    16   8   1   2    1    0  0  1
##         run_avg
## 106222  33.4475
## 111185 506.9075
## 11210  126.5575
## 176912 138.3600
## 165081 128.8425
## 25242   34.9075
```

```
summary(df)
```

```
##       MWG           NWG             KWG            MDIMC
##  Min.   : 16   Min.   : 16.00   Min.   :16.00   Min.   : 8
##  1st Qu.: 32   1st Qu.: 32.00   1st Qu.:16.00   1st Qu.: 8
##  Median : 64   Median : 64.00   Median :32.00   Median : 8
##  Mean   : 81   Mean   : 80.53   Mean   :25.63   Mean   :14
##  3rd Qu.:128   3rd Qu.:128.00   3rd Qu.:32.00   3rd Qu.:16
##  Max.   :128   Max.   :128.00   Max.   :32.00   Max.   :32
##      NDIMC           MDIMA          NDIMB           KWI
##  Min.   : 8.00   Min.   : 8.0   Min.   : 8.0   Min.   :2.000
##  1st Qu.: 8.00   1st Qu.: 8.0   1st Qu.: 8.0   1st Qu.:2.000
##  Median : 8.00   Median :16.0   Median :16.0   Median :8.000
##  Mean   :13.87   Mean   :17.4   Mean   :17.3   Mean   :5.022
##  3rd Qu.:16.00   3rd Qu.:32.0   3rd Qu.:32.0   3rd Qu.:8.000
##  Max.   :32.00   Max.   :32.0   Max.   :32.0   Max.   :8.000
##       VWM             VWN         STRM     STRN     SA       SB
##  Min.   :1.000   Min.   :1.000   0:4967   0:4987   0:5035   0:4987
##  1st Qu.:1.000   1st Qu.:1.000   1:5033   1:5013   1:4965   1:5013
##  Median :2.000   Median :2.000
##  Mean   :2.428   Mean   :2.459
##  3rd Qu.:4.000   3rd Qu.:4.000
##  Max.   :8.000   Max.   :8.000
##     run_avg
##  Min.   :  15.57
##  1st Qu.:  41.14
```

```
##  Median :  71.36
##  Mean   : 222.49
##  3rd Qu.: 227.67
##  Max.   :3259.89
```

```
  sapply(df, class)
```

```
##       MWG       NWG       KWG     MDIMC     NDIMC     MDIMA     NDIMB
## "integer" "integer" "integer" "integer" "integer" "integer" "integer"
##       KWI       VWM       VWN      STRM      STRN        SA        SB
## "integer" "integer" "integer"  "factor"  "factor"  "factor"  "factor"
##   run_avg
## "numeric"
```

Establish train/test Split

```
  set.seed(1234)
  i <- sample(1:nrow(df), 0.75*nrow(df), replace=FALSE)
  train <- df[i,]
  test <- df[-i,]
```

Linear Regression 1

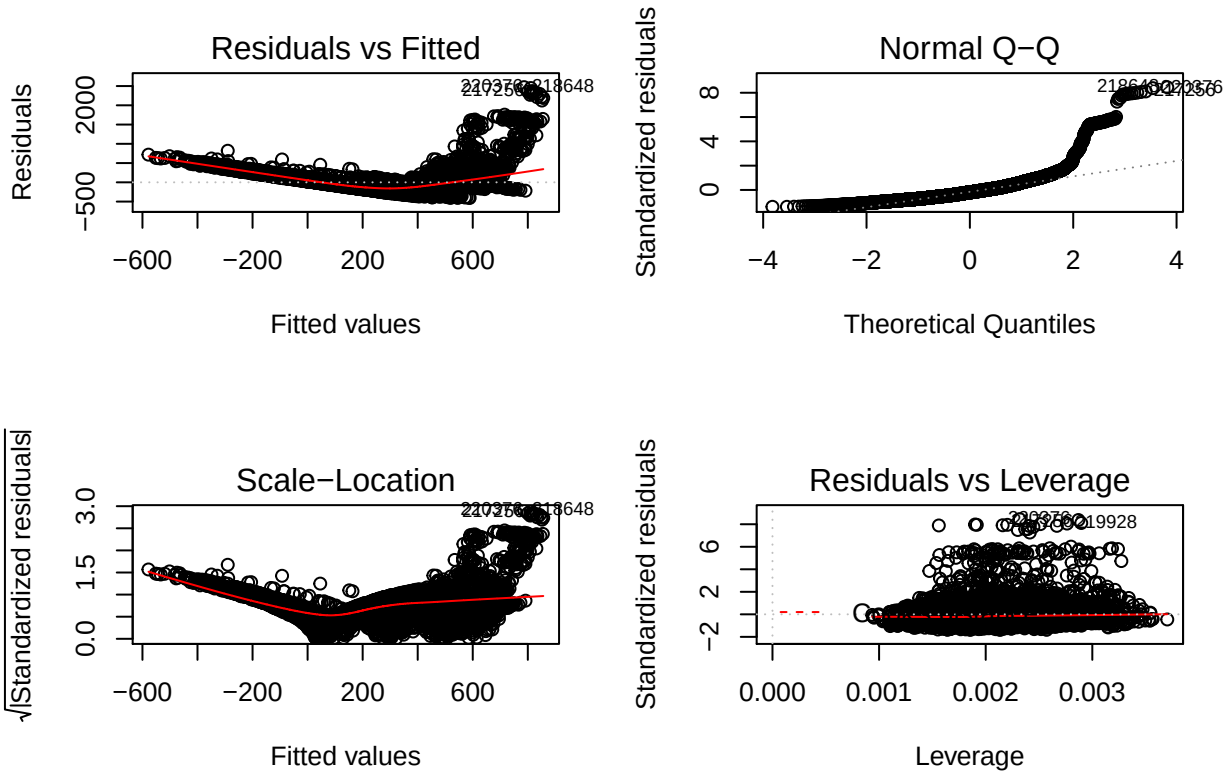
This Multiple Linear Regression model has a poor R^2, correlation, and MSE/RMSE. R^2 = 0.42, cor = 0.630, MSE = 86062, RMSE = 293

```
  lm1 <- lm(run_avg~., data=train)
  summary(lm1)
```

```
##
## Call:
## lm(formula = run_avg ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -405.37 -169.36  -57.38   81.02 2471.44
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -61.31439   20.17834  -3.039 0.002385 **
## MWG           3.42145    0.08789  38.929  < 2e-16 ***
## NWG           3.17863    0.08780  36.202  < 2e-16 ***
## KWG           5.96771    0.44798  13.321  < 2e-16 ***
## MDIMC       -17.09032    0.46814 -36.507  < 2e-16 ***
## NDIMC       -17.01837    0.46294 -36.761  < 2e-16 ***
## MDIMA         0.79650    0.38858   2.050 0.040423 *
## NDIMB         1.45905    0.39391   3.704 0.000214 ***
## KWI           3.05293    1.12744   2.708 0.006788 **
```

```
## VWM          -0.51102    1.94783  -0.262 0.793057
## VWN          -5.52163    1.92105  -2.874 0.004061 **
## STRM1       -10.20621    6.76725  -1.508 0.131551
## STRN1         1.45405    6.76569   0.215 0.829839
## SA1          39.33952    6.76576   5.815 6.33e-09 ***
## SB1          41.50731    6.76748   6.133 9.04e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 292.7 on 7485 degrees of freedom
## Multiple R-squared:  0.4124, Adjusted R-squared:  0.4113
## F-statistic: 375.3 on 14 and 7485 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,2))
plot(lm1)
```




```
pred1 <- predict(lm1, newdata=test)
cor(pred1, test$run_avg)
```

```
## [1] 0.6492963
```

```
mse1 <- mean((pred1-test$run_avg)^2)
mse1
```

```
## [1] 75806.34
```

```
  rmse1 <- sqrt(mse1)
  rmse1
```

```
## [1] 275.3295
```

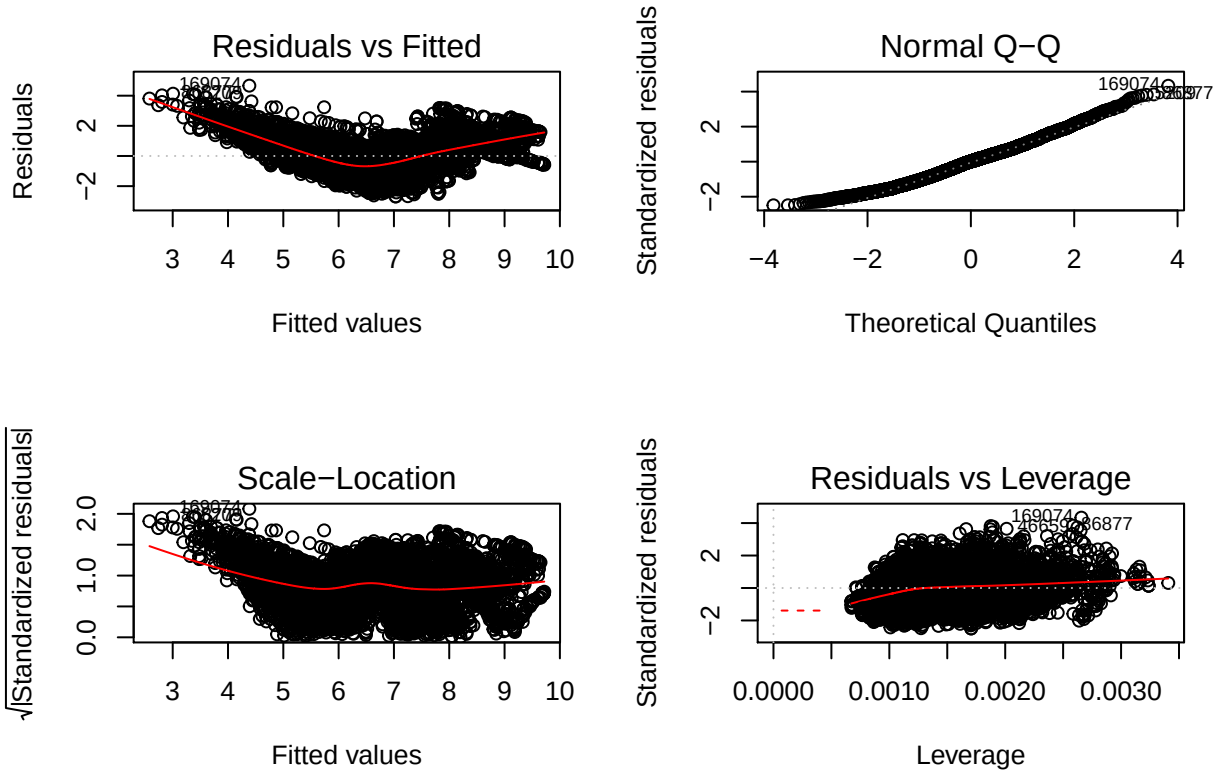
Linear Regression 2

Next, I predicted the log of run_avg. I took away the 4 worst predictors (MDIMA, NDIMB, KWI, STRN) to get the best possible metrics using log. Although R^2 is better than the previous model, the correlation and MSE/RMSE are worse. R^2 = 0.57, cor = 0.606, MSE = 186561, RMSE = 432

```
  lm2 <- lm(log2(run_avg)~. -MDIMA-NDIMB-KWI-STRN, data=train)
  summary(lm2)
```

```
##
## Call:
## lm(formula = log2(run_avg) ~ . - MDIMA - NDIMB - KWI - STRN,
##     data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6836 -0.7912  0.0140  0.6749  4.6637
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.0523207  0.0646403  93.631  < 2e-16 ***
## MWG          0.0194703  0.0003152  61.771  < 2e-16 ***
## NWG          0.0151142  0.0003149  47.994  < 2e-16 ***
## KWG          0.0173248  0.0016344  10.600  < 2e-16 ***
## MDIMC       -0.0802644  0.0016797 -47.784  < 2e-16 ***
## NDIMC       -0.0786219  0.0016551 -47.503  < 2e-16 ***
## VWM         -0.0096596  0.0069394  -1.392   0.1640
## VWN         -0.0347765  0.0068377  -5.086 3.75e-07 ***
## STRM1       -0.1920017  0.0249128  -7.707 1.45e-14 ***
## SA1         -0.2614275  0.0249090 -10.495  < 2e-16 ***
## SB1         -0.0680524  0.0249069  -2.732   0.0063 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.078 on 7489 degrees of freedom
## Multiple R-squared:  0.5662, Adjusted R-squared:  0.5656
## F-statistic: 977.4 on 10 and 7489 DF,  p-value: < 2.2e-16
```

```
  par(mfrow=c(2,2))
  plot(lm2)
```




```
  pred2 <- predict(lm2, newdata=test)
  cor(pred2, test$run_avg)
```

```
## [1] 0.6285575
```

```
  mse2 <- mean((pred2-test$run_avg)^2)
  mse2
```

```
## [1] 174751.5
```

```
  rmse2 <- sqrt(mse2)
  rmse2
```

```
## [1] 418.0329
```

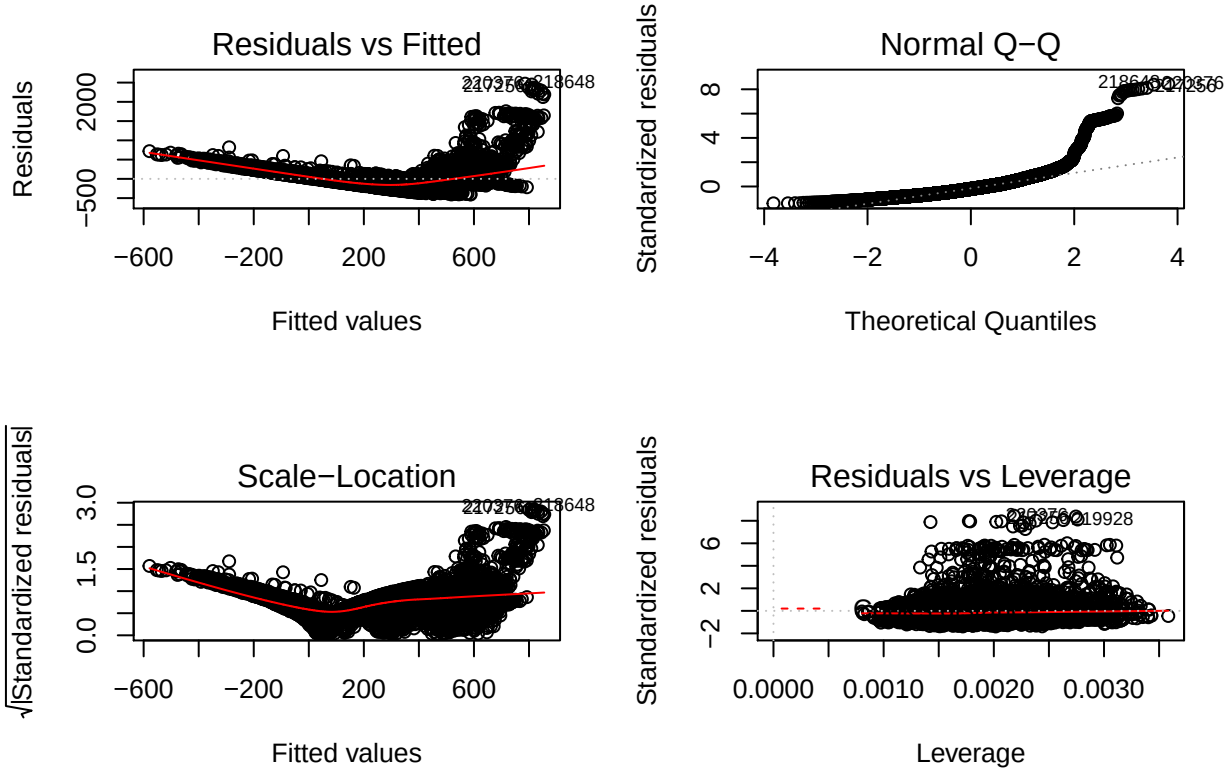
Linear Regression 3

Next, I jumped back to non-log linear regression and began varying my feature selections. This is the best result I got (predicting based on all attributes except STRN). lm1 still performs slightly better than lm3. R^2 = 0.42, cor = 0.630, MSE = 86070, RMSE = 293 Still, not much is learned from the data as it is a poor model.

```r
lm3 <- lm(run_avg~. -STRN, data=train)
summary(lm3)
```

```
##
## Call:
## lm(formula = run_avg ~ . - STRN, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -404.69 -169.62  -57.77   80.85 2472.17
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -60.54649   19.85821  -3.049  0.00230 **
## MWG           3.42113    0.08787  38.933  < 2e-16 ***
## NWG           3.17838    0.08779  36.204  < 2e-16 ***
## KWG           5.96828    0.44795  13.324  < 2e-16 ***
## MDIMC       -17.09153    0.46808 -36.514  < 2e-16 ***
## NDIMC       -17.02073    0.46278 -36.779  < 2e-16 ***
## MDIMA         0.79754    0.38852   2.053  0.04013 *
## NDIMB         1.46062    0.39382   3.709  0.00021 ***
## KWI           3.05068    1.12732   2.706  0.00682 **
## VWM          -0.50781    1.94765  -0.261  0.79431
## VWN          -5.52100    1.92093  -2.874  0.00406 **
## STRM1       -10.20847    6.76681  -1.509  0.13144
## SA1          39.34834    6.76520   5.816 6.26e-09 ***
## SB1          41.50666    6.76705   6.134 9.03e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 292.7 on 7486 degrees of freedom
## Multiple R-squared:  0.4124, Adjusted R-squared:  0.4114
## F-statistic: 404.2 on 13 and 7486 DF,  p-value: < 2.2e-16
```

```r
par(mfrow=c(2,2))
plot(lm3)
```




```
  pred3 <- predict(lm3, newdata=test)
  cor(pred3, test$run_avg)
```

```
## [1] 0.6492775
```

```
  mse3 <- mean((pred3-test$run_avg)^2)
  mse3
```

```
## [1] 75809.82
```

```
  rmse3 <- sqrt(mse3)
  rmse3
```

```
## [1] 275.3358
```

kNN Regression 1

This first chunk for kNN Regression is to establish a baseline for kNN Reg using k=3. Initially, we can see that kNN is far superior to Linear Regression. cor = 0.851, MSE = 40700, RMSE = 202

```
  library(caret)
```

```
## Warning: package 'caret' was built under R version 3.5.3
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
  library(DMwR)
```

```
## Loading required package: grid
```

```
  knn4 <- knnreg(run_avg~., data=train, k=3)
  summary(knn4)
```

```
##         Length Class  Mode
## learn   2      -none- list
## k       1      -none- numeric
## terms   3      terms  call
## xlevels 4      -none- list
## theDots 0      -none- list
```

```
  pred4 <- predict(knn4, newdata=test)
  cor(pred4, test$run_avg)
```

```
## [1] 0.8688814
```

```
  mse4 <- mean((pred4-test$run_avg)^2)
  mse4
```

```
## [1] 34434.5
```

```
  rmse4 <- sqrt(mse4)
  rmse4
```

```
## [1] 185.5654
```

kNN Regression 2 (feature selection)

In this chunk, the optimal k to use for kNN Regression is determined. Due to processing constraints, I only tested for k=1,3,5,7,9. k = 7 gave the highest cor = 0.864 and lowest mse = 36545, so k = 7 is out optimal feature.

```
  k_cor <- rep(0,10)
  k_mse <- rep(0,10)
  i <- 1
  for(k in seq(1,19,2)) {
    knnk <- knnreg(run_avg~., data=train, k=k)
    predk <- predict(knnk, newdata=test)
    k_cor[i] <- cor(predk, test$run_avg)
    k_mse[i] <- mean((predk-test$run_avg)^2)
    print(paste("k =", k, k_cor[i], k_mse[i]))
    i <- i+1
  }
```

```
## [1] "k = 1 0.830387035566725 46790.1540222021"
## [1] "k = 3 0.86888140223242 34434.5010987845"
## [1] "k = 5 0.874295133926274 32027.7215467625"
## [1] "k = 7 0.876543574542385 30965.5774648212"
## [1] "k = 9 0.87223858957019 31829.3335487566"
## [1] "k = 11 0.872542814614998 31556.4010995271"
## [1] "k = 13 0.869136734771664 32185.3009535108"
## [1] "k = 15 0.865540109415789 32925.9909352236"
## [1] "k = 17 0.859932594143607 34218.975112585"
## [1] "k = 19 0.855791030584439 35145.6532993789"
```

kNN Regression 3

Now, we plot the k values to better visualize how our feature selection affects kNN.

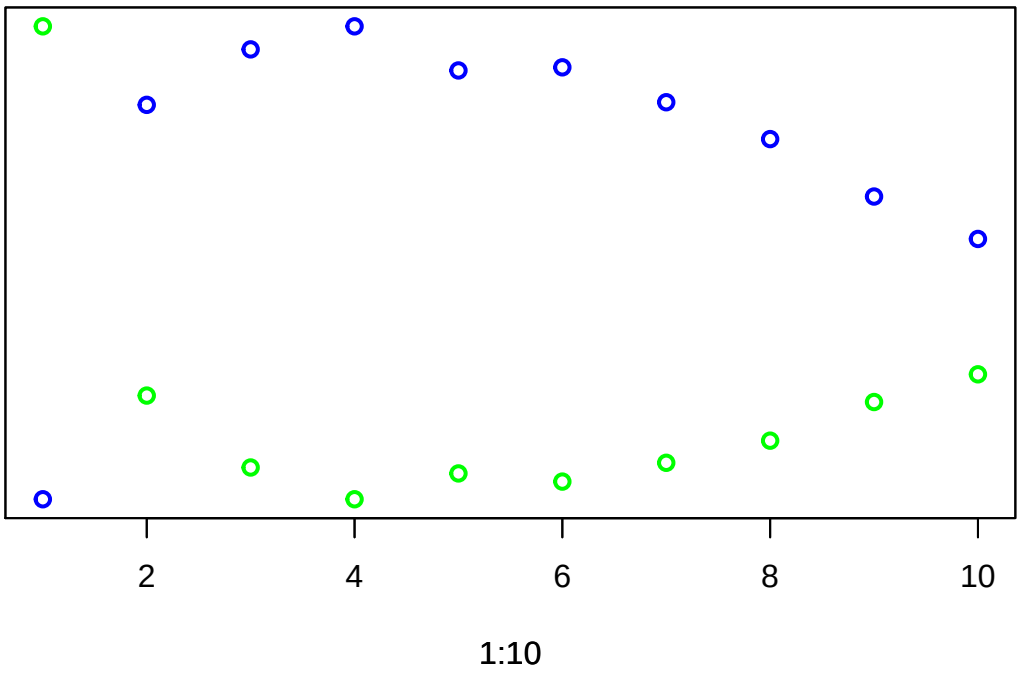
```
plot(1:10, k_cor, lwd=2, col='blue', ylab="", yaxt='n')
par(new=TRUE)
plot(1:10, k_mse, lwd=2, col='green', labels=FALSE, ylab="", yaxt='n')
```

```
## Warning in plot.window(...): "labels" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "labels" is not a graphical parameter

## Warning in box(...): "labels" is not a graphical parameter

## Warning in title(...): "labels" is not a graphical parameter
```



SVM Regression 1 (linear kernel)

kNN with k=7 is still the best model, so polynomial and radial kernels will not make a significant difference. cor = 0.588, MSE = 109672, RMSE = 331

```
  library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.5.3
```

```
  set.seed(1234)
  svm5 <- svm(run_avg~., data=train, kernel="linear", cost=1, scale=FALSE)
  summary(svm5)
```

```
##
## Call:
## svm(formula = run_avg ~ ., data = train, kernel = "linear", cost = 1,
##     scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  1
```

```
##       gamma:  0.06666667
##     epsilon:  0.1
##
##
## Number of Support Vectors:  7494
```

```
  pred5 <- predict(svm5, newdata=test)
  cor(pred5, test$run_avg)
```

```
## [1] 0.6117476
```

```
  mse5 <- mean((pred5-test$run_avg)^2)
  mse5
```

```
## [1] 98629.13
```

```
  rmse5 <- sqrt(mse5)
  rmse5
```

```
## [1] 314.0528
```

Decision Tree Regression

While correlation is lower than kNN's, MSE is much lower which is great. This is overall the best model because it has a high correlation and a relatively low MSE/RMSE. cor = 0.930, MSE = 19588, RMSE = 140

```
  library(tree)
```

```
## Warning: package 'tree' was built under R version 3.5.3
```

```
  tree6 <- tree(run_avg~., data=train)
  summary(tree6)
```

```
##
## Regression tree:
## tree(formula = run_avg ~ ., data = train)
## Variables actually used in tree construction:
## [1] "MWG"   "NWG"   "NDIMC" "MDIMC" "SB"    "SA"    "KWG"   "KWI"
## Number of terminal nodes:  23
## Residual mean deviance:  15430 = 115400000 / 7477
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -740.10  -48.25  -22.13    0.00   28.74 1193.00
```

```
  pred6 <- predict(tree6, newdata=test)
  cor(pred6, test$run_avg)
```

```
## [1] 0.9366255
```

```r
  mse6 <- mean((pred6-test$run_avg)^2)
  mse6
```

```
## [1] 16069.89
```

```r
  rmse6 <- sqrt(mse6)
  rmse6
```

```
## [1] 126.7671
```

Analysis

-Decision Tree Regression has the highest correlation and lowet MSE/RMSE, thus it is the best performing algorithm out of linear regression, kNN regression, SVM, and decision tree regression. -If the data is simple, then decision tree regression will usually perform worse than other regression algorithms. However, since our dataset is complex and decision tree regression is high variance/low bias, the algorithm performs better than the others.