APAN5420 — HW 8, Stocks

Megan Wilder

7/21/18

Contents

---

1 Load Stock Data

```
#load packages
library(readr)
library(dplyr)
library(tidyr)
library(purrr)
library(stringr)
library(ggplot2)
library(tidyquant)
library(RcppRoll)
library(TTR)
library(RCurl)
library(kableExtra)
library(Metrics)
library(caret)
library(ggsn)
library(h2o)

#read in the one file
stockDF <-
read_csv("stocks.csv", col_types = c("dcDddddnnddddd")) %>%
arrange(Symbol, Date)

#remove X1 column and created feature columns
stockDF$X1 <- NULL
stockDF$Log_Open <- NULL
stockDF$Log_High <- NULL
stockDF$Log_Low <- NULL
stockDF$Log_Close <- NULL
```

```r
stockDF$Log_Volume <- NULL

#remove OpenInt as it is all 0's
stockDF$OpenInt <- NULL

#view stocks with volume of NA
stockDF %>% filter(is.na(Volume)) #none
```

```
## # A tibble: 0 x 7
## # ... with 7 variables: Symbol <chr>, Date <date>, Open <dbl>, High <dbl>,
## #   Low <dbl>, Close <dbl>, Volume <dbl>
```

```r
#view stocks with volume of 0
stockDF %>% filter(Volume == 0)
```

```
## # A tibble: 24,625 x 7
##    Symbol       Date    Open    High     Low   Close Volume
##     <chr>     <date>   <dbl>   <dbl>   <dbl>   <dbl>  <dbl>
##  1      a 2010-04-26 24.6390 24.9680 24.6190 24.8560      0
##  2     aa 2010-04-26 29.5390 29.8370 29.2790 29.3400      0
##  3   aaba 2010-04-26 17.7100 17.7100 17.3400 17.3900      0
##  4   aame 2009-07-27  0.6930  0.6930  0.6930  0.6930      0
##  5   aame 2009-07-31  0.8181  0.8181  0.8181  0.8181      0
##  6   aame 2009-08-24  0.7701  0.7701  0.7701  0.7701      0
##  7   aapl 2010-04-26 34.8290 34.8920 34.3450 34.5140      0
##  8   abac 2010-07-19 24.0000 24.0000 24.0000 24.0000      0
##  9    abc 2010-04-26 28.1930 28.5830 28.1010 28.4000      0
## 10  abr_a 2016-01-19 23.7440 23.7440 23.7440 23.7440      0
## # ... with 24,615 more rows
```

```r
#remove rows with volume of 0
stockDF <- stockDF %>% filter(Volume != 0)

#explore DF
summary(stockDF)
```

```
##     Symbol               Date                 Open
##  Length:14863031    Min.   :1962-01-02   Min.   :0.000e+00
##  Class :character   1st Qu.:2007-11-28   1st Qu.:8.000e+00
##  Mode  :character   Median :2012-02-16   Median :1.600e+01
##                     Mean   :2010-06-18   Mean   :7.323e+03
##                     3rd Qu.:2015-05-19   3rd Qu.:2.900e+01
##                     Max.   :2017-11-10   Max.   :1.424e+09
##       High                Low                Close
##  Min.   :0.000e+00   Min.   :0.000e+00   Min.   :0.000e+00
##  1st Qu.:8.000e+00   1st Qu.:8.000e+00   1st Qu.:8.000e+00
##  Median :1.600e+01   Median :1.500e+01   Median :1.600e+01
##  Mean   :7.574e+03   Mean   :6.971e+03   Mean   :7.245e+03
##  3rd Qu.:2.900e+01   3rd Qu.:2.800e+01   3rd Qu.:2.900e+01
##  Max.   :1.433e+09   Max.   :1.331e+09   Max.   :1.392e+09
##      Volume
##  Min.   :1.000e+00
##  1st Qu.:3.316e+04
##  Median :1.932e+05
##  Mean   :1.588e+06
```

```
##   3rd Qu.:8.943e+05
##   Max.   :2.304e+09
```

```r
kable(head(stockDF)) %>% kable_styling(latex_options = "scale_down")
```

Symbol	Date	Open	High	Low	Close	Volume
a	1999-11-18	30.713	33.754	27.002	29.702	66277506
a	1999-11-19	28.986	29.027	26.872	27.257	16142920
a	1999-11-22	27.886	29.702	27.044	29.702	6970266
a	1999-11-23	28.688	29.446	27.002	27.002	6332082
a	1999-11-24	27.083	28.309	27.002	27.717	5132147
a	1999-11-26	27.594	28.012	27.509	27.807	1832635

```r
#change stock symbols to all caps to match SP500
stockDF$Symbol <- sapply(stockDF$Symbol, toupper)

#SP500 information
sp500URL <-
getURL(
"https://raw.githubusercontent.com/datasets/s-and-p-500-companies/master/data/constituents.csv"
)
sp500File <- file.path('data', 'constituents.csv')

if (file.exists(sp500File)) {
sp500URL <- sp500File
}

# read in the SP500 information
sp500Members <- read_csv(sp500URL)

sp500DF <- stockDF %>% filter (Symbol %in% sp500Members$Symbol)

summary(sp500DF)
```

```
##     Symbol               Date                 Open         
##  Length:2774811     Min.   :1962-01-02   Min.   :    0.008  
##  Class :character   1st Qu.:1995-12-08   1st Qu.:    8.784  
##  Mode  :character   Median :2006-01-23   Median :   23.075  
##                     Mean   :2003-05-26   Mean   :   45.974  
##                     3rd Qu.:2012-03-08   3rd Qu.:   43.889  
##                     Max.   :2017-11-10   Max.   :11720.240  
##       High               Low               Close          
##  Min.   :    0.008   Min.   :    0.008   Min.   :    0.008  
##  1st Qu.:    8.912   1st Qu.:    8.659   1st Qu.:    8.789  
##  Median :   23.385   Median :   22.752   Median :   23.081  
##  Mean   :   46.512   Mean   :   45.422   Mean   :   45.975  
##  3rd Qu.:   44.388   3rd Qu.:   43.380   3rd Qu.:   43.898  
##  Max.   :11768.390   Max.   :11400.200   Max.   :11697.130  
##      Volume         
##  Min.   :9.000e+01  
##  1st Qu.:8.143e+05  
```

```
##   Median :2.011e+06
##   Mean   :5.629e+06
##   3rd Qu.:4.884e+06
##   Max.   :2.070e+09
```

```
gStockDF <- sp500DF %>%
group_by(Symbol) %>%
arrange(Date)

nrow(gStockDF)
```

```
## [1] 2774811
```

```
# Check number of rows per symbol, need to keep EMA window under the minimum
gStockDF %>%
group_by(Symbol) %>%
summarise(Num = n()) %>%
arrange(Num)
```

```
## # A tibble: 498 x 2
##    Symbol   Num
##     <chr> <int>
##  1    BHF    83
##  2   BHGE    91
##  3    DXC   155
##  4   ARNC   260
##  5    FTV   343
##  6     UA   413
##  7   CSRA   502
##  8    HPE   511
##  9    KHC   596
## 10   PYPL   596
## # ... with 488 more rows
```

2 Add Features

```
#In addition to features created in class,
#added EMA (exponential moving average) for 30 days and
#rolling 30 day average for open, high, low, close and volume

df2 <- gStockDF %>%
mutate(
Open_Close_PctChange = (Close - Open) / Open * 100,

Open_Close_Delt = Delt(Open, Close, k = 0) * 100,

Open_Change   = Open   - lag(Open),
High_Change   = High   - lag(High),
Low_Change    = Low    - lag(Low),
Close_Change  = Close  - lag(Close),
Volume_Change = Volume - lag(Volume),

Daily_Return  = Close / Open,
```

```r
Open_PctChange   = Open_Change   / lag(Open)   * 100,
High_PctChange   = High_Change   / lag(High)   * 100,
Low_PctChange    = Low_Change    / lag(Low)    * 100,
Close_PctChange  = Close_Change  / lag(Close)  * 100,
Volume_PctChange = Volume_Change / lag(Volume) * 100,

Open_Mean10   = roll_mean(Open,   10, fill = NA, na.rm = TRUE),
High_Mean10   = roll_mean(High,   10, fill = NA, na.rm = TRUE),
Low_Mean10    = roll_mean(Low,    10, fill = NA, na.rm = TRUE),
Close_Mean10  = roll_mean(Close,  10, fill = NA, na.rm = TRUE),
Volume_Mean10 = roll_mean(Volume, 10, fill = NA, na.rm = TRUE),

Open_Mean10_R   = roll_meanr(Open,   10, fill = NA, na.rm = TRUE),
High_Mean10_R   = roll_meanr(High,   10, fill = NA, na.rm = TRUE),
Low_Mean10_R    = roll_meanr(Low,    10, fill = NA, na.rm = TRUE),
Close_Mean10_R  = roll_meanr(Close,  10, fill = NA, na.rm = TRUE),
Volume_Mean10_R = roll_meanr(Volume, 10, fill = NA, na.rm = TRUE),

Open_Mean30   = roll_mean(Open,   30, fill = NA, na.rm = TRUE),
High_Mean30   = roll_mean(High,   30, fill = NA, na.rm = TRUE),
Low_Mean30    = roll_mean(Low,    30, fill = NA, na.rm = TRUE),
Close_Mean30  = roll_mean(Close,  30, fill = NA, na.rm = TRUE),
Volume_Mean30 = roll_mean(Volume, 30, fill = NA, na.rm = TRUE),

Open_Mean30_R   = roll_meanr(Open,   30, fill = NA, na.rm = TRUE),
High_Mean30_R   = roll_meanr(High,   30, fill = NA, na.rm = TRUE),
Low_Mean30_R    = roll_meanr(Low,    30, fill = NA, na.rm = TRUE),
Close_Mean30_R  = roll_meanr(Close,  30, fill = NA, na.rm = TRUE),
Volume_Mean30_R = roll_meanr(Volume, 30, fill = NA, na.rm = TRUE),

Open_SD30   = roll_sd(Open,   30, fill = NA, na.rm = TRUE),
High_SD30   = roll_sd(High,   30, fill = NA, na.rm = TRUE),
Low_SD30    = roll_sd(Low,    30, fill = NA, na.rm = TRUE),
Close_SD30  = roll_sd(Close,  30, fill = NA, na.rm = TRUE),
Volume_SD30 = roll_sd(Volume, 30, fill = NA, na.rm = TRUE),

Open_VAR30   = roll_var(Open,   30, fill = NA, na.rm = TRUE),
High_VAR30   = roll_var(High,   30, fill = NA, na.rm = TRUE),
Low_VAR30    = roll_var(Low,    30, fill = NA, na.rm = TRUE),
Close_VAR30  = roll_var(Close,  30, fill = NA, na.rm = TRUE),
Volume_VAR30 = roll_var(Volume, 30, fill = NA, na.rm = TRUE),

Open_EMA10   = EMA(Open,   n = 10),
High_EMA10   = EMA(High,   n = 10),
Low_EMA10    = EMA(Low,    n = 10),
Close_EMA10  = EMA(Close,  n = 10),
Volume_EMA10 = EMA(Volume, n = 10),

Open_EMA30   = EMA(Open,   n = 30),
High_EMA30   = EMA(High,   n = 30),
Low_EMA30    = EMA(Low,    n = 30),
Close_EMA30  = EMA(Close,  n = 30),
Volume_EMA30 = EMA(Volume, n = 30)
```

```r
) %>%
arrange(Symbol, Date)

# add SP500 characteristics (sector)
df3 <- df2 %>% left_join(sp500Members, by = "Symbol")

#View number of stocks by sector
df3 %>%
group_by(Sector) %>%
summarise(Num = length(unique(Symbol))) %>%
arrange(Num)
```

```
## # A tibble: 11 x 2
##                        Sector   Num
##                         <chr> <int>
##  1 Telecommunication Services     3
##  2                  Materials    25
##  3                  Utilities    28
##  4                     Energy    31
##  5                Real Estate    31
##  6           Consumer Staples    33
##  7                Health Care    60
##  8                Industrials    67
##  9                 Financials    68
## 10     Information Technology    72
## 11     Consumer Discretionary    80
```

```r
# Add day of week and quarter features
df3$DOW <- weekdays(df3$Date)
df3$Quarter <- quarters(df3$Date)

#view number of instances per quarter
df3 %>%
group_by(Quarter) %>%
summarise(Num = n()) %>%
arrange(Num)
```

```
## # A tibble: 4 x 2
##   Quarter    Num
##     <chr>  <int>
## 1      Q1 671162
## 2      Q4 695799
## 3      Q2 699343
## 4      Q3 708507
```

```r
#view number of instances per day
df3 %>%
group_by(DOW) %>%
summarise(Num = n()) %>%
arrange(Num)
```

```
## # A tibble: 5 x 2
##         DOW    Num
##       <chr>  <int>
## 1    Monday 523390
```

```
## 2    Friday 555149
## 3  Thursday 558488
## 4   Tuesday 568083
## 5 Wednesday 569701
```

3 Split data by rolling time

```r
# add a year feature
df3$Year <- as.numeric(format(df3$Date, "%Y"))
df3$QtrYear <- sprintf("%s-%s", df3$Year, df3$Quarter)

# grab the last 16 quarters
qtrs <- df3 %>%
ungroup %>%
select(Year, QtrYear) %>%
unique() %>%
filter(between(Year, 2010, 2012)) %>%
arrange(QtrYear)

qtrs
```

```
## # A tibble: 12 x 2
##     Year QtrYear
##    <dbl>   <chr>
##  1  2010 2010-Q1
##  2  2010 2010-Q2
##  3  2010 2010-Q3
##  4  2010 2010-Q4
##  5  2011 2011-Q1
##  6  2011 2011-Q2
##  7  2011 2011-Q3
##  8  2011 2011-Q4
##  9  2012 2012-Q1
## 10  2012 2012-Q2
## 11  2012 2012-Q3
## 12  2012 2012-Q4
```

```r
# train/test are 3 and 1 qtrs in duration
trainLen <- 3
testLen <- 1

# rows in the data.frame
trainStart <- 1
trainStop <- nrow(qtrs) - trainLen

# lists we are collecting
trainSet <- list()
testSet <- list()

# identify and print the sets
while (trainStart <= trainStop) {
trainQ <- qtrs[seq(trainStart, length.out = trainLen),]$QtrYear
testQ <-
```

```r
qtrs[seq(trainStart + trainLen, length.out = testLen),]$QtrYear

print("Set:")
print(trainQ)
print(testQ)
print("")

# extract the training and testing from df3
trainDF <- df3 %>% filter(QtrYear %in% trainQ)
testDF <- df3 %>% filter(QtrYear %in% testQ)

key <- sprintf("SET_%d", trainStart)

trainSet[[key]] <- trainDF
testSet[[key]] <- testDF

trainStart <- trainStart + 1
}
```

```
## [1] "Set:"
## [1] "2010-Q1" "2010-Q2" "2010-Q3"
## [1] "2010-Q4"
## [1] ""
## [1] "Set:"
## [1] "2010-Q2" "2010-Q3" "2010-Q4"
## [1] "2011-Q1"
## [1] ""
## [1] "Set:"
## [1] "2010-Q3" "2010-Q4" "2011-Q1"
## [1] "2011-Q2"
## [1] ""
## [1] "Set:"
## [1] "2010-Q4" "2011-Q1" "2011-Q2"
## [1] "2011-Q3"
## [1] ""
## [1] "Set:"
## [1] "2011-Q1" "2011-Q2" "2011-Q3"
## [1] "2011-Q4"
## [1] ""
## [1] "Set:"
## [1] "2011-Q2" "2011-Q3" "2011-Q4"
## [1] "2012-Q1"
## [1] ""
## [1] "Set:"
## [1] "2011-Q3" "2011-Q4" "2012-Q1"
## [1] "2012-Q2"
## [1] ""
## [1] "Set:"
## [1] "2011-Q4" "2012-Q1" "2012-Q2"
## [1] "2012-Q3"
## [1] ""
## [1] "Set:"
## [1] "2012-Q1" "2012-Q2" "2012-Q3"
## [1] "2012-Q4"
```

```
## [1] ""
```

```r
#view names
names(trainSet)
```

```
## [1] "SET_1" "SET_2" "SET_3" "SET_4" "SET_5" "SET_6" "SET_7" "SET_8" "SET_9"
```

```r
names(testSet)
```

```
## [1] "SET_1" "SET_2" "SET_3" "SET_4" "SET_5" "SET_6" "SET_7" "SET_8" "SET_9"
```

4 Create Functions

4.1 GLM: Plot Important Variables

```r
#create function to plot GLM variable importance
plotGLMVariableImportance <- function(glmModel) {
pData <- na.omit(glmModel$standardized_coefficient_magnitudes)

pData <- pData %>%
arrange(abs(pData$coefficients))

pData$names <- factor(pData$names, levels = pData$names)

ggplot(data = pData) +
aes(x = names, y = coefficients, fill = sign) +
geom_col(alpha = 0.8) +
guides(fill = FALSE) +
labs(x = "Coefficient", y = "Value", title = 'Model Coefficients') +
scale_fill_manual(breaks = c('NEG', 'POS'),
values = c('darkred', 'darkgreen')) +
coord_flip()
}
```

4.2 RF: Plot Important Variables

```r
#create function to plot RF variable importance
plotRFVariableImportance <- function(rfModel) {
rData <- rfModel$variable_importances

rData <- rData %>%
arrange(rData$scaled_importance)

rData$variable <- factor(rData$variable, levels = rData$variable)

ggplot(data = rData) +
aes(x = variable, y = scaled_importance) +
geom_col(fill = 'blue', alpha = 0.8) +
guides(fill = FALSE) +
labs(x = "Variable", y = "Scaled Importance", title = 'Variable Importance') +
coord_flip()
}
```

4.3 Find Outliers

```
# define a function to find outliers
FindOutliers <- function(data) {
lowerq = quantile(data)[2]
upperq = quantile(data)[4]
iqr = upperq - lowerq
extreme.threshold.upper = (iqr * 400) + upperq
extreme.threshold.lower = lowerq - (iqr * 400)
result <-
which(data > extreme.threshold.upper |
data < extreme.threshold.lower)
}
```

5 View Variable Correlations

```
#calculate correlation matrix
correlationMatrix <-
round(cor(df3[sapply(df3, is.numeric)], use = "complete.obs"), 2)

#find attributes that are highly corrected (>0.9)
highlyCorrelated <-
findCorrelation(
correlationMatrix,
cutoff = (0.9),
verbose = FALSE,
names = FALSE,
exact = FALSE
)

#print indexes of highly correlated attributes
print(highlyCorrelated)
```

```
##  [1]  2  4 20 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 42 45 47 49
## [24] 50 51 52 53 54 55 56 57 58  1  6  7  3 19 39 40 44
```

```
#important variables
important_var = colnames(df3[, -highlyCorrelated])
```

6 Modeling Techniques

6.1 GLM

```
#Start H2O
h2o.init(nthreads = -1, max_mem_size = '8G')

# clean slate in case the cluster was already running
h2o.removeAll()
```

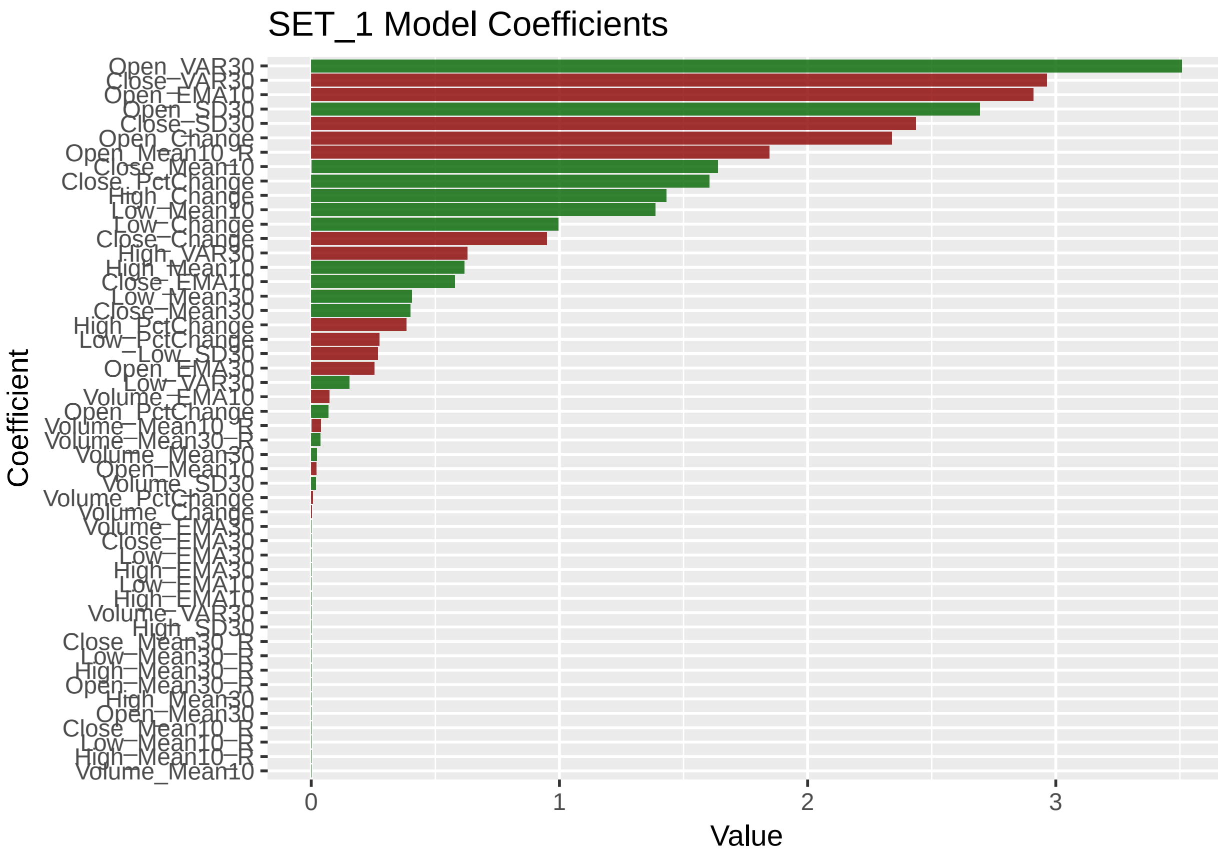
6.1.1 Response = Open Close % Change, Predictors = Most Variables

```
#Response = Open_Close_PctChange
#Predictors = All variables excluding Daily_Return,
#Open_Close_Delt ( due to high correlation).
#Also excludes Quarter, Symbol, Sector, Year, QtrYear,
#OpenInt, DOW, Name as these variables were dropped by the model.

# response and predictors to use
resp <- "Open_Close_PctChange"
pred <-
c(
"Open_Change",
"High_Change",
"Low_Change",
"Close_Change",
"Volume_Change",
"Open_PctChange",
"High_PctChange",
"Low_PctChange",
"Close_PctChange",
"Volume_PctChange",
"Open_Mean10",
"High_Mean10",
"Low_Mean10",
"Close_Mean10",
"Volume_Mean10",
"Open_Mean10_R",
"High_Mean10_R",
"Low_Mean10_R",
"Close_Mean10_R",
"Volume_Mean10_R",
"Open_Mean30",
"High_Mean30",
"Low_Mean30",
"Close_Mean30",
"Volume_Mean30",
"Open_Mean30_R",
"High_Mean30_R",
"Low_Mean30_R",
"Close_Mean30_R",
"Volume_Mean30_R",
"Open_SD30",
"High_SD30",
"Low_SD30",
"Close_SD30",
"Volume_SD30",
"Open_VAR30",
"High_VAR30",
"Low_VAR30",
"Close_VAR30",
"Volume_VAR30",
"Open_EMA10",
```

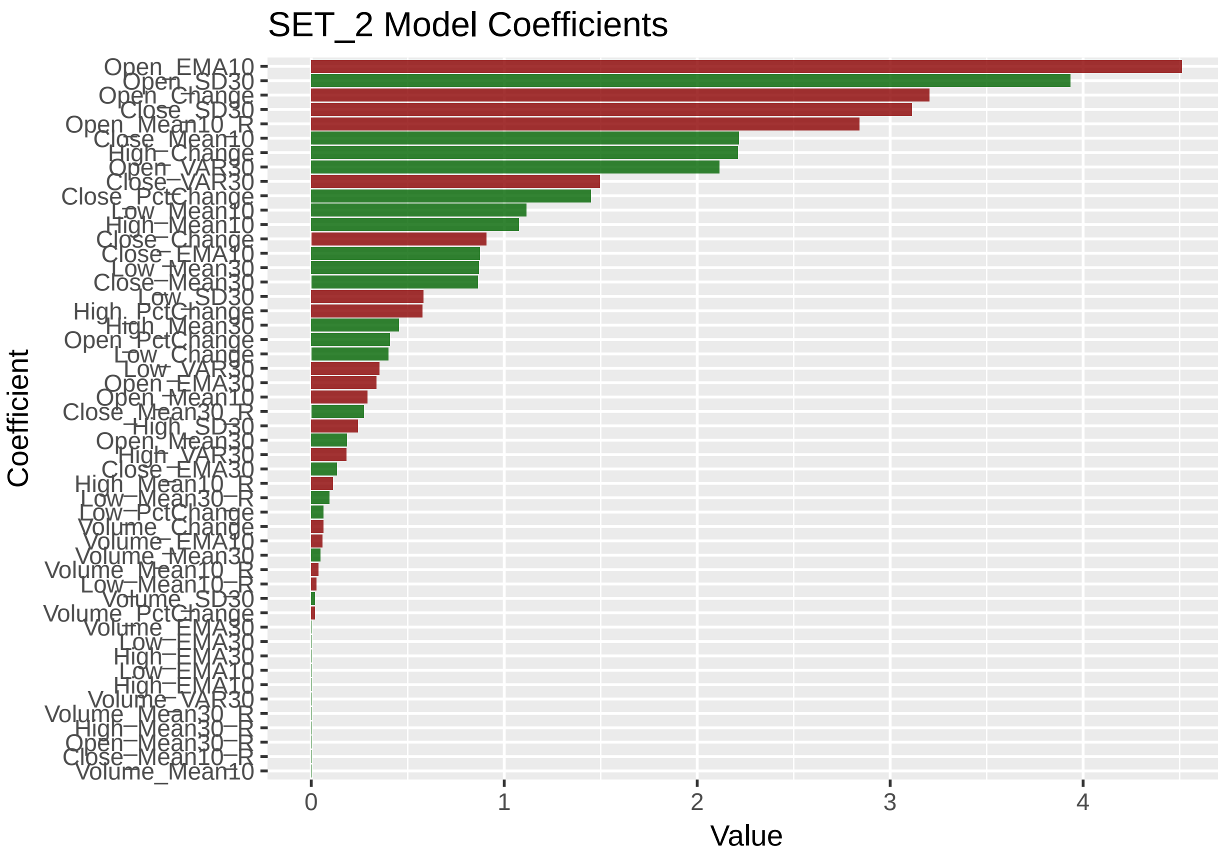
```r
"High_EMA10",
"Low_EMA10",
"Close_EMA10",
"Volume_EMA10",
"Open_EMA30",
"High_EMA30",
"Low_EMA30",
"Close_EMA30",
"Volume_EMA30"
)

# iterating over the keys
modelList <- list()
testList <- list()

for (aKey in names(trainSet)) {
aTrain <- trainSet[[aKey]]
aTest <- testSet[[aKey]]

# make h2o data.frame, loads into H2O service
train.hex <- as.h2o(aTrain)
test.hex <- as.h2o(aTest)

# Summary
summary(train.hex, exact_quantiles = TRUE)
summary(test.hex, exact_quantiles = TRUE)

# train a model with that data
loop.glm = h2o.glm(
x = pred,
y = resp,
training_frame = train.hex,
validation_frame = test.hex,
family = "gaussian",
alpha = 0.5
)

# save the model
modelList[[aKey]] <- loop.glm

# predict response variable
glm.pred = h2o.predict(object = loop.glm, newdata = test.hex)

# save the prediction
testList[[aKey]] <- glm.pred
}
```

```r
#plot important variables for training set, list test set model RMSE
for (aKey in names(modelList)) {
mm <- modelList[[aKey]]

pp <- plotGLMVariableImportance(mm@model) +
labs(title = sprintf('%s Model Coefficients', aKey))
print(pp)
```

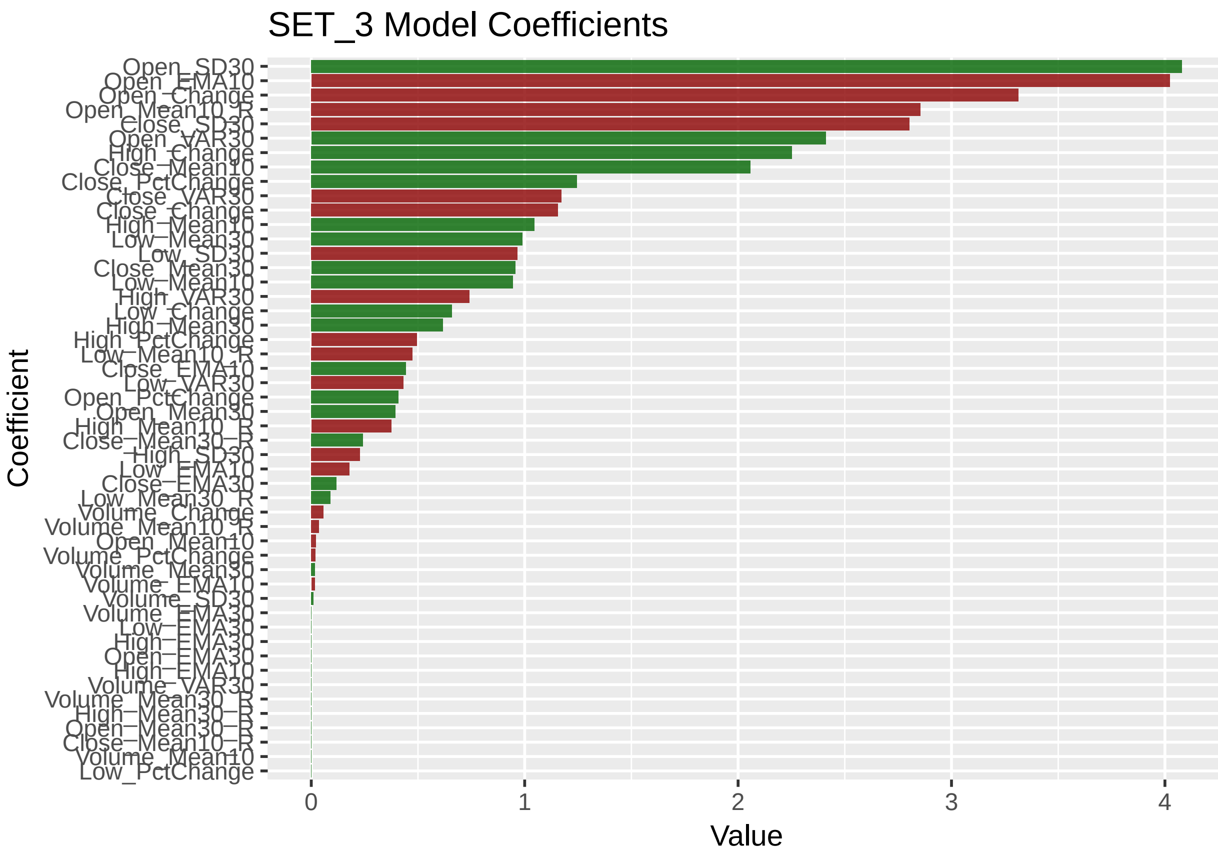
```r
print(sprintf("%s Model RMSE: %s", aKey, h2o.rmse(mm, valid = T)))

}
```



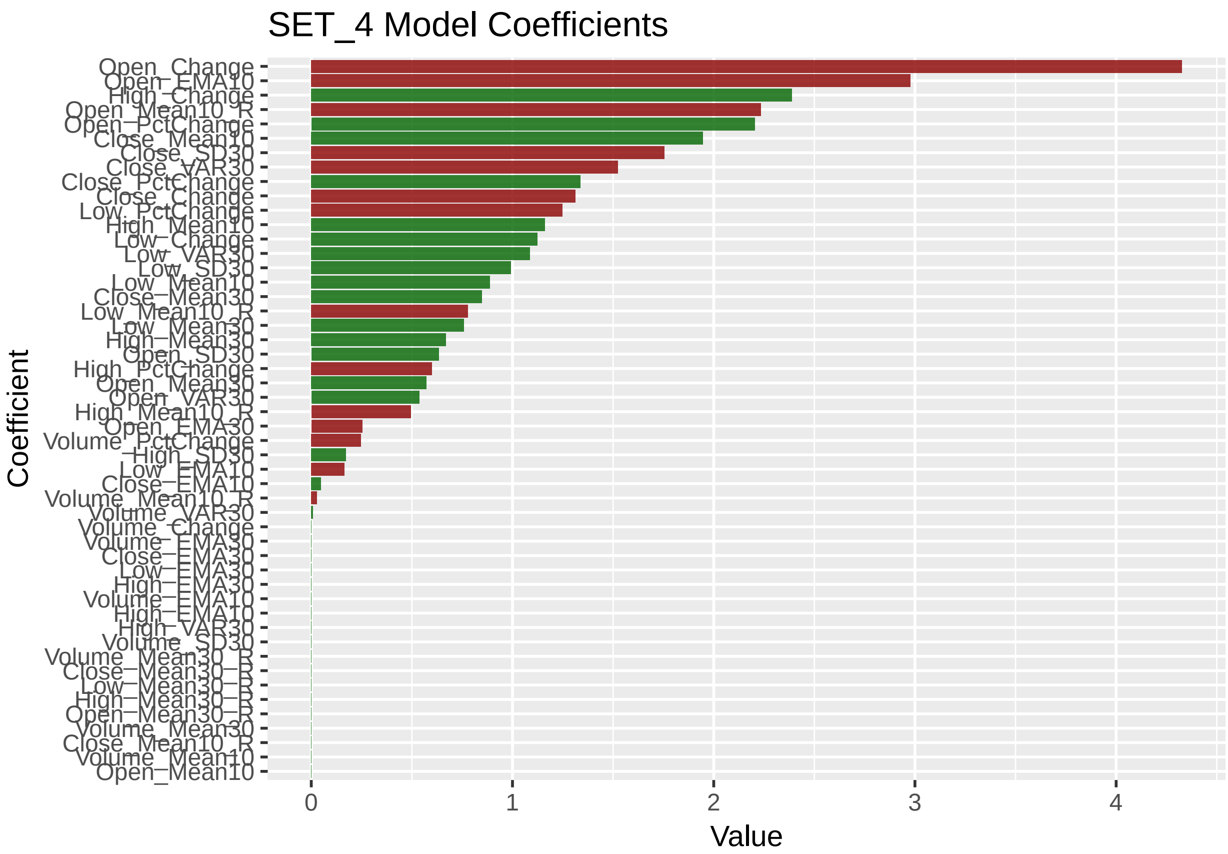

```
## [1] "SET_1 Model RMSE: 5.02204369296121"
```



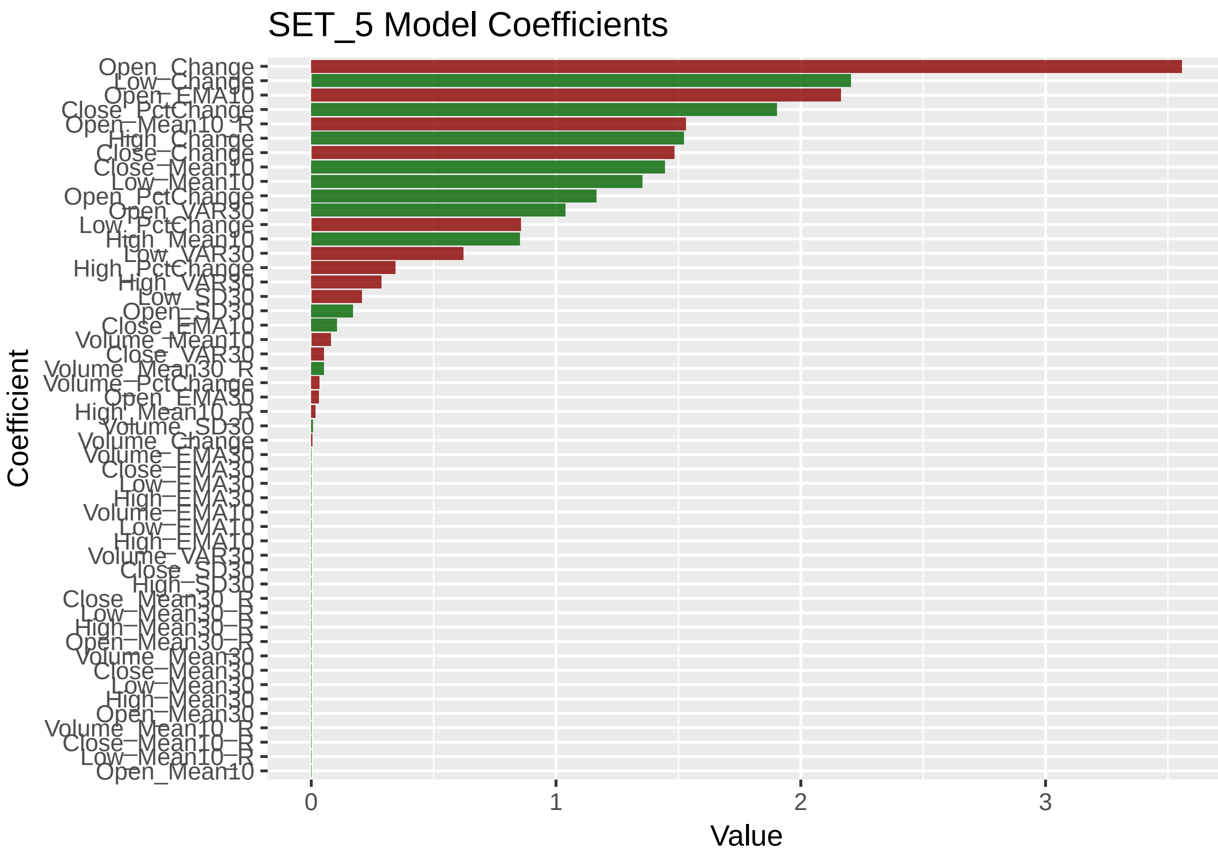

```
## [1] "SET_2 Model RMSE: 2.05590473315647"
```



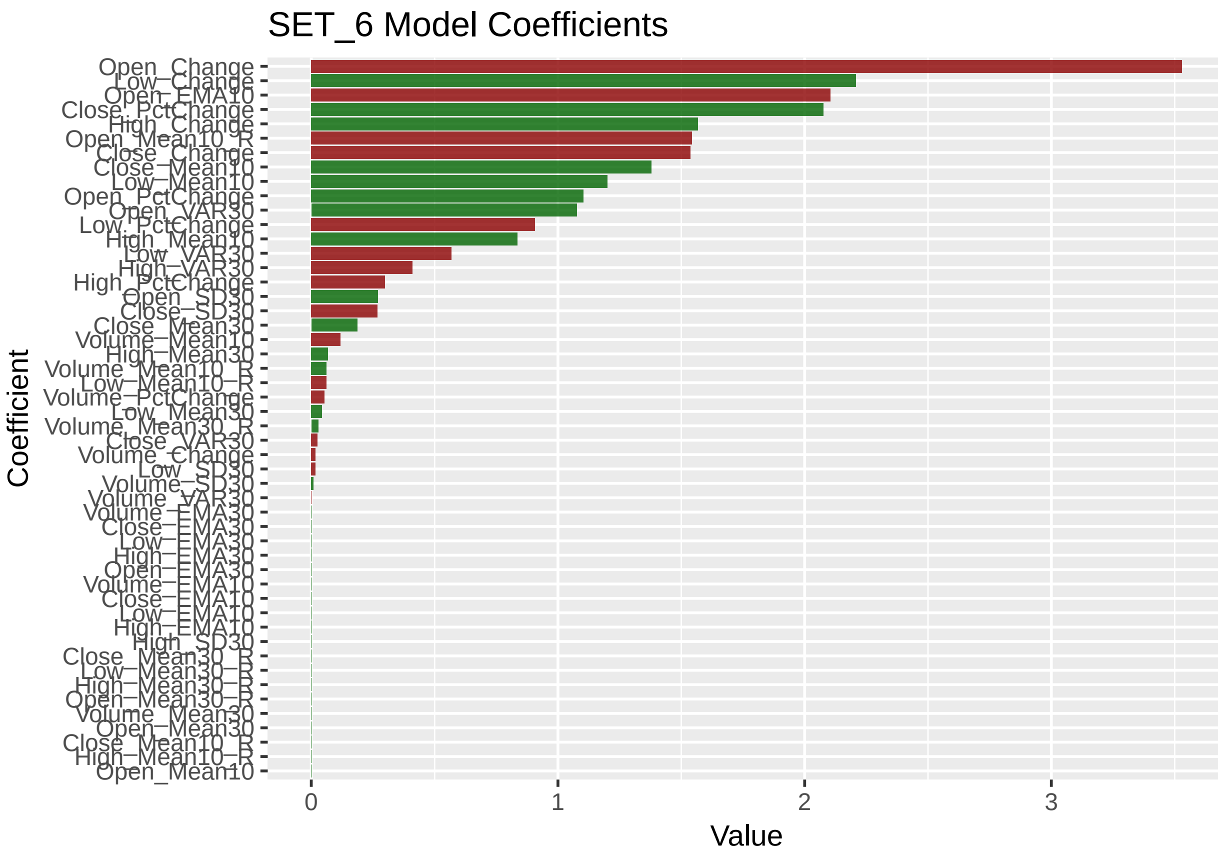

```
## [1] "SET_3 Model RMSE: 4.64972145326448"
```



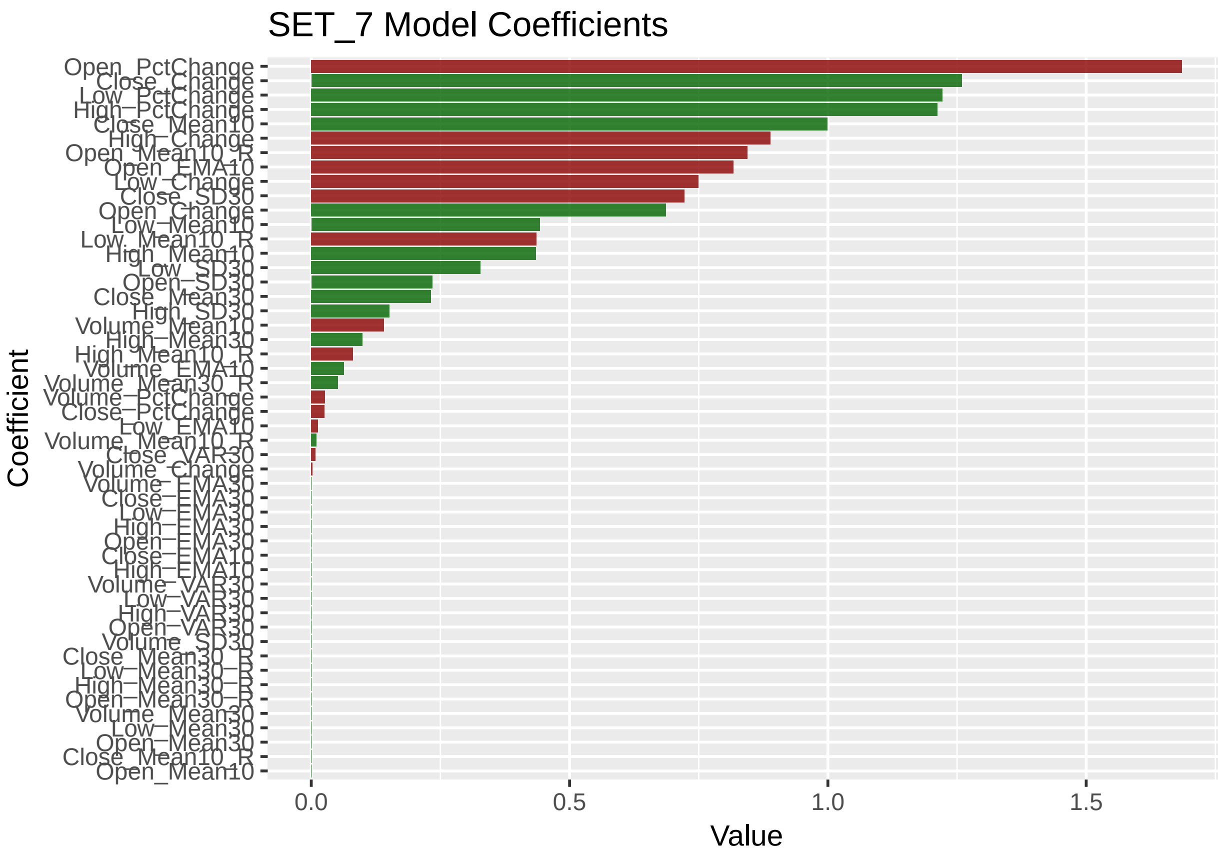

```
## [1] "SET_4 Model RMSE: 2.8907784953672"
```



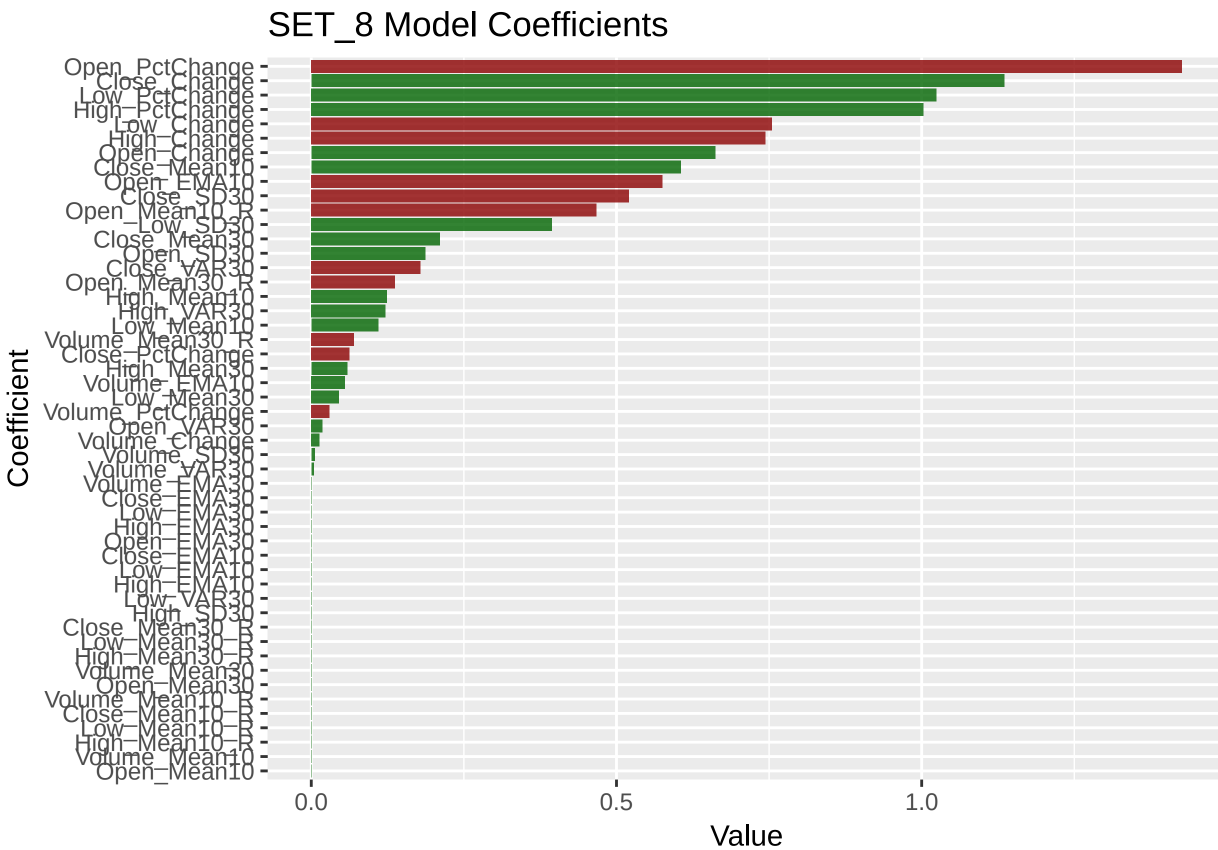

```
## [1] "SET_5 Model RMSE: 1.62181600910602"
```



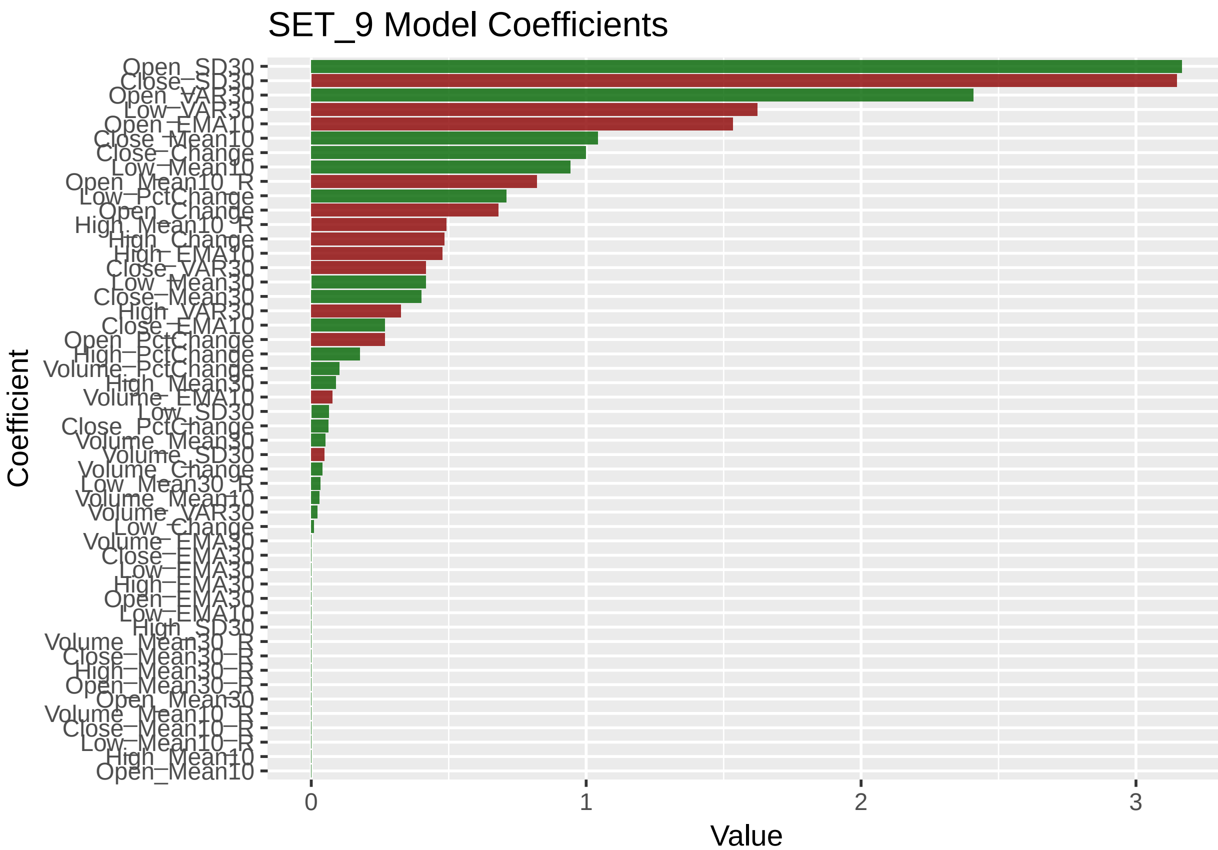

```
## [1] "SET_6 Model RMSE: 6.2200904839212"
```




```
## [1] "SET_7 Model RMSE: 0.905408476829758"
```




```
## [1] "SET_8 Model RMSE: 32.1572350288796"
```




```
## [1] "SET_9 Model RMSE: 7.6748903764146"
```

```r
#View RMSE for each test set
#[1] "SET_1 Model RMSE: 5.02204369296121"
#[1] "SET_2 Model RMSE: 2.05590473315647"
#[1] "SET_3 Model RMSE: 4.6497214532644"
#[1] "SET_4 Model RMSE: 2.89077849536722"
#[1] "SET_5 Model RMSE: 1.62181600910602"
#[1] "SET_6 Model RMSE: 6.2200904839212"
#[1] "SET_7 Model RMSE: 0.905408476829758"
#[1] "SET_8 Model RMSE: 32.1572350288796"
#[1] "SET_9 Model RMSE: 7.6748903764146"
#model appears unstable with Set_8 having a much larger RMSE

#Most important variables based on frequency across test sets
imp.var <- c(
"Open_EMA10",
"Open_Change",
"High_Change",
"Open_Mean10_R",
"Close_Mean10",
"Close_SD30",
"Open_SD30",
"Low_Change",
"Open_PctChange",
"Open_VAR30",
"Close_Change",
```

```r
"Close_PctChange",
"High_PctChange",
"Low_PctChange",
"Close_VAR30",
"Low_VAR30"
)

#change testSet sets to dataframes
test.set1 <- as.data.frame(testSet[["SET_1"]])
test.set2 <- as.data.frame(testSet[["SET_2"]])
test.set3 <- as.data.frame(testSet[["SET_3"]])
test.set4 <- as.data.frame(testSet[["SET_4"]])
test.set5 <- as.data.frame(testSet[["SET_5"]])
test.set6 <- as.data.frame(testSet[["SET_6"]])
test.set7 <- as.data.frame(testSet[["SET_7"]])
test.set8 <- as.data.frame(testSet[["SET_8"]])
test.set9 <- as.data.frame(testSet[["SET_9"]])

#change testList sets to dataframes
pred.set1 = as.data.frame(testList[["SET_1"]])
pred.set2 = as.data.frame(testList[["SET_2"]])
pred.set3 = as.data.frame(testList[["SET_3"]])
pred.set4 = as.data.frame(testList[["SET_4"]])
pred.set5 = as.data.frame(testList[["SET_5"]])
pred.set6 = as.data.frame(testList[["SET_6"]])
pred.set7 = as.data.frame(testList[["SET_7"]])
pred.set8 = as.data.frame(testList[["SET_8"]])
pred.set9 = as.data.frame(testList[["SET_9"]])

#attached predicted values to test dataframe
temp.1 <- cbind(test.set1, pred.set1)
temp.2 <- cbind(test.set2, pred.set2)
temp.3 <- cbind(test.set3, pred.set3)
temp.4 <- cbind(test.set4, pred.set4)
temp.5 <- cbind(test.set5, pred.set5)
temp.6 <- cbind(test.set6, pred.set6)
temp.7 <- cbind(test.set7, pred.set7)
temp.8 <- cbind(test.set8, pred.set8)
temp.9 <- cbind(test.set9, pred.set9)

pred.1 <-
rbind(temp.1,
temp.2,
temp.3,
temp.4,
temp.5,
temp.6,
temp.7,
temp.8,
temp.9)

#ape computes the elementwise absolute percent
#difference between two numeric vectors
```

```r
pred.1$APE <- ape(pred.1$Open_Close_PctChange, pred.1$predict)

# use the function to identify outliers
outliers <- FindOutliers(pred.1$APE)

# remove non outliers
GLM.Outliers.1 <- pred.1[outliers, ]

#remove rows with APE of Inf
GLM.Outliers.1 <- GLM.Outliers.1[is.finite(GLM.Outliers.1$APE),]
```

6.1.2 Response = Open Close % Change, Predictors = 16 Most Imp Variables

```r
#Response = Open_Close_PctChange
#Predictors = 16 most important variables found in the above model

# response and predictors to use
resp <- "Open_Close_PctChange"
pred <- imp.var

# iterating over the keys
modelList <- list()
testList <- list()

for (aKey in names(trainSet)) {
aTrain <- trainSet[[aKey]]
aTest <- testSet[[aKey]]

# make h2o data.frame, loads into H2O service
train.hex <- as.h2o(aTrain)
test.hex <- as.h2o(aTest)

# Summary
summary(train.hex, exact_quantiles = TRUE)
summary(test.hex, exact_quantiles = TRUE)

# train a model with that data
loop.glm = h2o.glm(
x = pred,
y = resp,
training_frame = train.hex,
validation_frame = test.hex,
family = "gaussian",
alpha = 0.5
)

# save the model
modelList[[aKey]] <- loop.glm

# predict response variable
glm.pred = h2o.predict(object = loop.glm, newdata = test.hex)
```

```r
# save the prediction
testList[[aKey]] <- glm.pred
}
```

```r
#plot important variables for training set, list test set model RMSE
for (aKey in names(modelList)) {
mm <- modelList[[aKey]]

pp <- plotGLMVariableImportance(mm@model) +
labs(title = sprintf('%s Model Coefficients', aKey))
print(pp)

print(sprintf("%s Model RMSE: %s", aKey, h2o.rmse(mm, valid = T)))
}
```

```r
#View RMSE for each test set
#[1] "SET_1 Model RMSE: 5.01826025688796"
#[1] "SET_2 Model RMSE: 2.05829593980223"
#[1] "SET_3 Model RMSE: 4.65575656155093"
#[1] "SET_4 Model RMSE: 2.8941446776997"
#[1] "SET_5 Model RMSE: 1.62553456396679"
#[1] "SET_6 Model RMSE: 6.18520264967136"
#[1] "SET_7 Model RMSE: 0.903506684521318"
#[1] "SET_8 Model RMSE: 31.9674181270305"
#[1] "SET_9 Model RMSE: 7.65861209427199"
#model appears unstable with Set_8 having a much larger RMSE

#change testSet sets to dataframes
test.set1 <- as.data.frame(testSet[["SET_1"]])
test.set2 <- as.data.frame(testSet[["SET_2"]])
test.set3 <- as.data.frame(testSet[["SET_3"]])
test.set4 <- as.data.frame(testSet[["SET_4"]])
test.set5 <- as.data.frame(testSet[["SET_5"]])
test.set6 <- as.data.frame(testSet[["SET_6"]])
test.set7 <- as.data.frame(testSet[["SET_7"]])
test.set8 <- as.data.frame(testSet[["SET_8"]])
test.set9 <- as.data.frame(testSet[["SET_9"]])

#change testList sets to dataframes
pred.set1 = as.data.frame(testList[["SET_1"]])
pred.set2 = as.data.frame(testList[["SET_2"]])
pred.set3 = as.data.frame(testList[["SET_3"]])
pred.set4 = as.data.frame(testList[["SET_4"]])
pred.set5 = as.data.frame(testList[["SET_5"]])
pred.set6 = as.data.frame(testList[["SET_6"]])
pred.set7 = as.data.frame(testList[["SET_7"]])
pred.set8 = as.data.frame(testList[["SET_8"]])
pred.set9 = as.data.frame(testList[["SET_9"]])

#attached predicted values to test dataframe
temp.1 <- cbind(test.set1, pred.set1)
temp.2 <- cbind(test.set2, pred.set2)
temp.3 <- cbind(test.set3, pred.set3)
temp.4 <- cbind(test.set4, pred.set4)
```

```
temp.5 <- cbind(test.set5, pred.set5)
temp.6 <- cbind(test.set6, pred.set6)
temp.7 <- cbind(test.set7, pred.set7)
temp.8 <- cbind(test.set8, pred.set8)
temp.9 <- cbind(test.set9, pred.set9)

pred.2 <-
rbind(temp.1,
temp.2,
temp.3,
temp.4,
temp.5,
temp.6,
temp.7,
temp.8,
temp.9)

#ape computes the elementwise absolute percent difference
#between two numeric vectors
pred.2$APE <- ape(pred.2$Open_Close_PctChange, pred.2$predict)

# use the function to identify outliers
outliers <- FindOutliers(pred.2$APE)

# remove non outliers
GLM.Outliers.2 <- pred.2[outliers, ]

#remove rows with APE of Inf
GLM.Outliers.2 <- GLM.Outliers.2[is.finite(GLM.Outliers.2$APE),]
```

6.1.3 Response = Open Close % Change, Predictors = 4 Most Imp Variables

```
#Response = Open_Close_PctChange
#Predictors = 4 most important variables found in first model

# response and predictors to use
resp <- "Open_Close_PctChange"
pred <- c("Open_EMA10",
"Open_Change",
"High_Change",
"Open_Mean10_R")

# iterating over the keys
modelList <- list()
testList <- list()

for (aKey in names(trainSet)) {
aTrain <- trainSet[[aKey]]
aTest <- testSet[[aKey]]

# make h2o data.frame, loads into H2O service
train.hex <- as.h2o(aTrain)
```

```r
test.hex <- as.h2o(aTest)

# Summary
summary(train.hex, exact_quantiles = TRUE)
summary(test.hex, exact_quantiles = TRUE)

# train a model with that data
loop.glm = h2o.glm(
x = pred,
y = resp,
training_frame = train.hex,
validation_frame = test.hex,
family = "gaussian",
alpha = 0.5
)

# save the model
modelList[[aKey]] <- loop.glm

# predict response variable
glm.pred = h2o.predict(object = loop.glm, newdata = test.hex)

# save the prediction
testList[[aKey]] <- glm.pred
}
```

```r
#plot important variables for training set, list test set model RMSE
for (aKey in names(modelList)) {
mm <- modelList[[aKey]]

pp <- plotGLMVariableImportance(mm@model) +
labs(title = sprintf('%s Model Coefficients', aKey))
print(pp)

print(sprintf("%s Model RMSE: %s", aKey, h2o.rmse(mm, valid = T)))
}
```

```r
#View RMSE for each test set
#[1] "SET_1 Model RMSE: 4.95343213843853"
#[1] "SET_2 Model RMSE: 2.10317604868142"
#[1] "SET_3 Model RMSE: 4.44100571544713"
#[1] "SET_4 Model RMSE: 2.71382600088799"
#[1] "SET_5 Model RMSE: 1.8225087431764"
#[1] "SET_6 Model RMSE: 1.54072411710327"
#[1] "SET_7 Model RMSE: 1.36217867236545"
#[1] "SET_8 Model RMSE: 5.49847658045872"
#[1] "SET_9 Model RMSE: 7.44228822179927"
#model appears stable with similar RMSE accross test sets

#change testSet sets to dataframes
test.set1 <- as.data.frame(testSet[["SET_1"]])
test.set2 <- as.data.frame(testSet[["SET_2"]])
test.set3 <- as.data.frame(testSet[["SET_3"]])
test.set4 <- as.data.frame(testSet[["SET_4"]])
```

```r
test.set5 <- as.data.frame(testSet[["SET_5"]])
test.set6 <- as.data.frame(testSet[["SET_6"]])
test.set7 <- as.data.frame(testSet[["SET_7"]])
test.set8 <- as.data.frame(testSet[["SET_8"]])
test.set9 <- as.data.frame(testSet[["SET_9"]])

#change testList sets to dataframes
pred.set1 = as.data.frame(testList[["SET_1"]])
pred.set2 = as.data.frame(testList[["SET_2"]])
pred.set3 = as.data.frame(testList[["SET_3"]])
pred.set4 = as.data.frame(testList[["SET_4"]])
pred.set5 = as.data.frame(testList[["SET_5"]])
pred.set6 = as.data.frame(testList[["SET_6"]])
pred.set7 = as.data.frame(testList[["SET_7"]])
pred.set8 = as.data.frame(testList[["SET_8"]])
pred.set9 = as.data.frame(testList[["SET_9"]])

#attached predicted values to test dataframe
temp.1 <- cbind(test.set1, pred.set1)
temp.2 <- cbind(test.set2, pred.set2)
temp.3 <- cbind(test.set3, pred.set3)
temp.4 <- cbind(test.set4, pred.set4)
temp.5 <- cbind(test.set5, pred.set5)
temp.6 <- cbind(test.set6, pred.set6)
temp.7 <- cbind(test.set7, pred.set7)
temp.8 <- cbind(test.set8, pred.set8)
temp.9 <- cbind(test.set9, pred.set9)

pred.3 <-
rbind(temp.1,
temp.2,
temp.3,
temp.4,
temp.5,
temp.6,
temp.7,
temp.8,
temp.9)

#ape computes the elementwise absolute percent difference
#between two numeric vectors
pred.3$APE <- ape(pred.3$Open_Close_PctChange, pred.3$predict)

# use the function to identify outliers
outliers <- FindOutliers(pred.3$APE)

# remove non outliers
GLM.Outliers.3 <- pred.3[outliers, ]

#remove rows with APE of Inf
GLM.Outliers.3 <- GLM.Outliers.3[is.finite(GLM.Outliers.3$APE),]
```

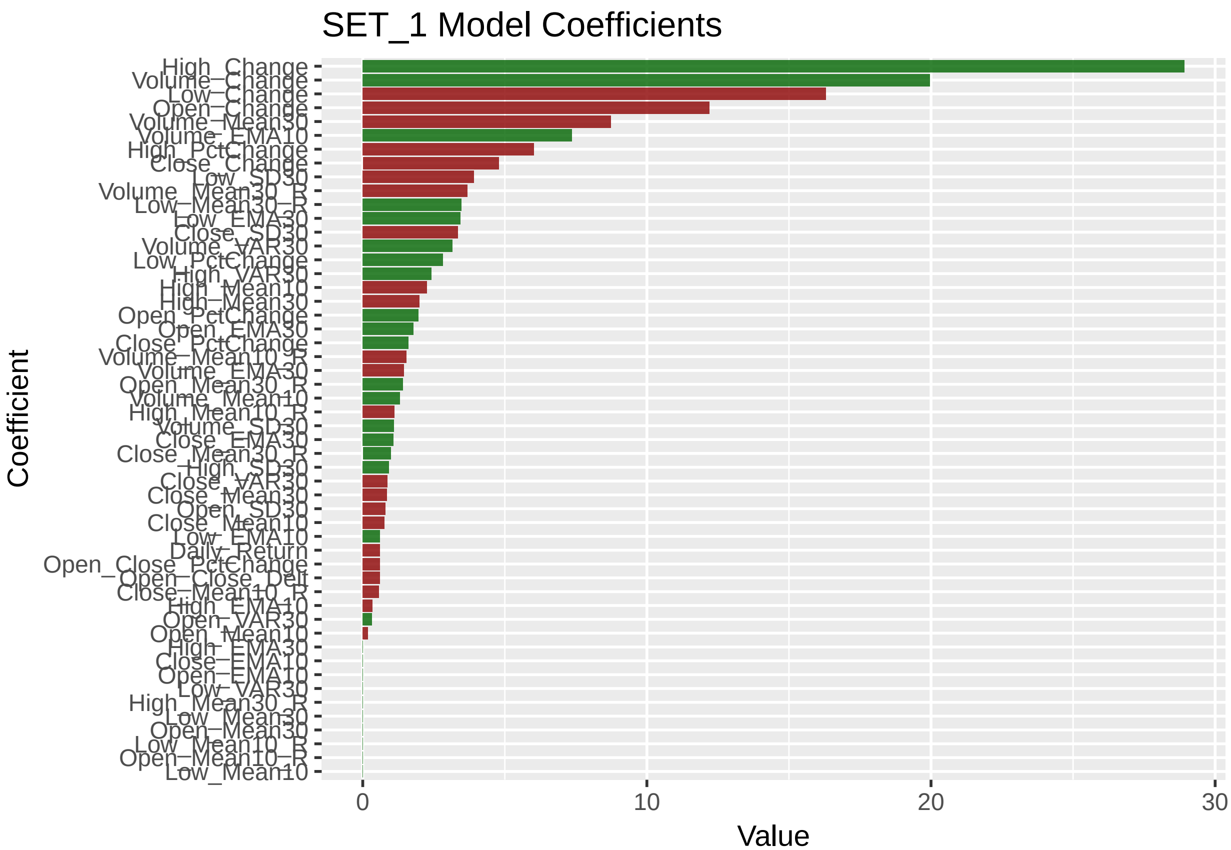
6.1.4 Response = Volume % Change, Predictors = Most Variables

```
#Response = Volume_PctChange
#Predictors = All variables excluding Quarter, Symbol,
#Sector, Year, QtrYear, OpenInt, DOW, Name as these
#variables were dropped by the model.

# response and predictors to use
resp <- "Volume_PctChange"
pred <-
c(
"Open_Close_PctChange",
"Daily_Return",
"Open_Close_Delt",
"Open_Change",
"High_Change",
"Low_Change",
"Close_Change",
"Volume_Change",
"Open_PctChange",
"High_PctChange",
"Low_PctChange",
"Close_PctChange",
"Volume_PctChange",
"Open_Mean10",
"High_Mean10",
"Low_Mean10",
"Close_Mean10",
"Volume_Mean10",
"Open_Mean10_R",
"High_Mean10_R",
"Low_Mean10_R",
"Close_Mean10_R",
"Volume_Mean10_R",
"Open_Mean30",
"High_Mean30",
"Low_Mean30",
"Close_Mean30",
"Volume_Mean30",
"Open_Mean30_R",
"High_Mean30_R",
"Low_Mean30_R",
"Close_Mean30_R",
"Volume_Mean30_R",
"Open_SD30",
"High_SD30",
"Low_SD30",
"Close_SD30",
"Volume_SD30",
"Open_VAR30",
"High_VAR30",
"Low_VAR30",
"Close_VAR30",
```

```
"Volume_VAR30",
"Open_EMA10",
"High_EMA10",
"Low_EMA10",
"Close_EMA10",
"Volume_EMA10",
"Open_EMA30",
"High_EMA30",
"Low_EMA30",
"Close_EMA30",
"Volume_EMA30"
)

# iterating over the keys
modelList <- list()
testList <- list()

for (aKey in names(trainSet)) {
aTrain <- trainSet[[aKey]]
aTest <- testSet[[aKey]]

# make h2o data.frame, loads into H2O service
train.hex <- as.h2o(aTrain)
test.hex <- as.h2o(aTest)

# Summary
summary(train.hex, exact_quantiles = TRUE)
summary(test.hex, exact_quantiles = TRUE)

# train a model with that data
loop.glm = h2o.glm(
x = pred,
y = resp,
training_frame = train.hex,
validation_frame = test.hex,
family = "gaussian",
alpha = 0.5
)

# save the model
modelList[[aKey]] <- loop.glm

# predict response variable
glm.pred = h2o.predict(object = loop.glm, newdata = test.hex)

# save the prediction
testList[[aKey]] <- glm.pred
}
```

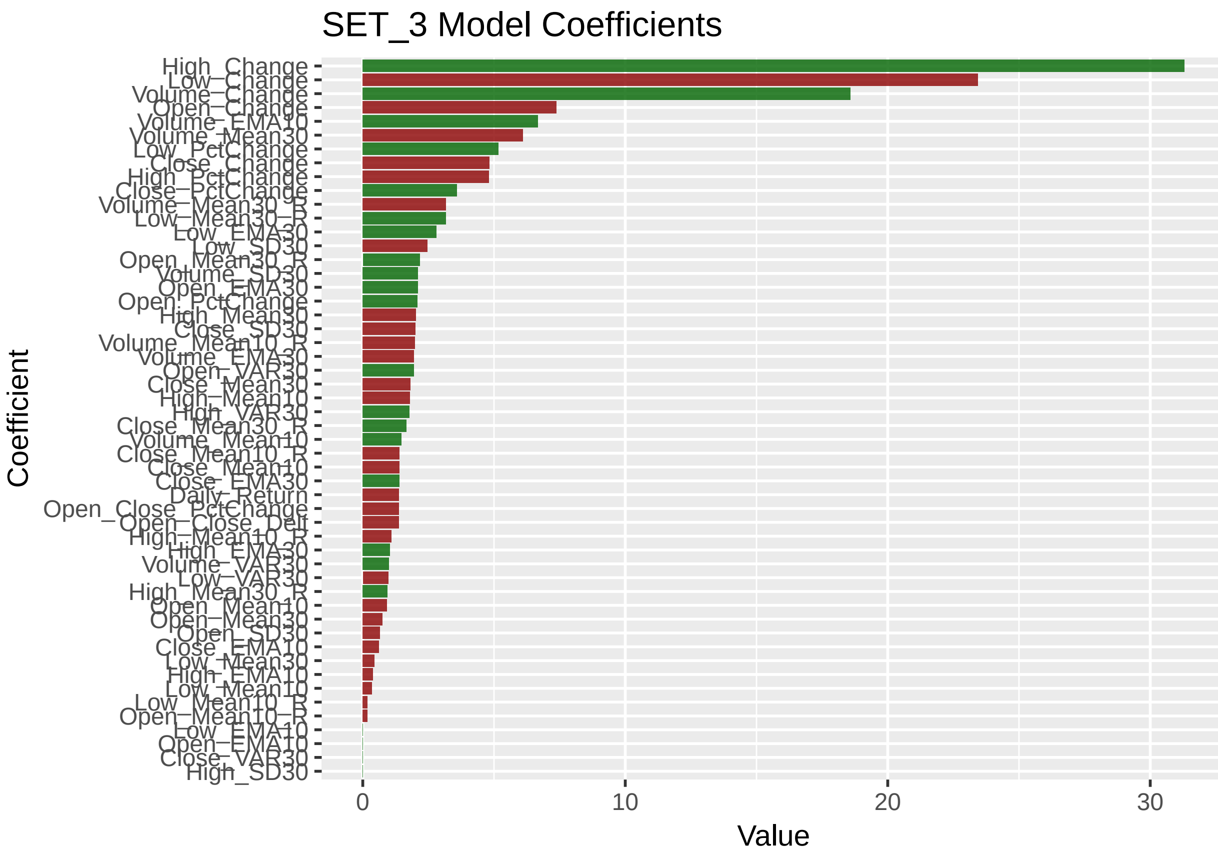
```
#plot important variables for training set, list test set model RMSE
for (aKey in names(modelList)) {
mm <- modelList[[aKey]]

pp <- plotGLMVariableImportance(mm@model) +
```

```
labs(title = sprintf('%s Model Coefficients', aKey))
print(pp)

print(sprintf("%s Model RMSE: %s", aKey, h2o.rmse(mm, valid = T)))

}
```

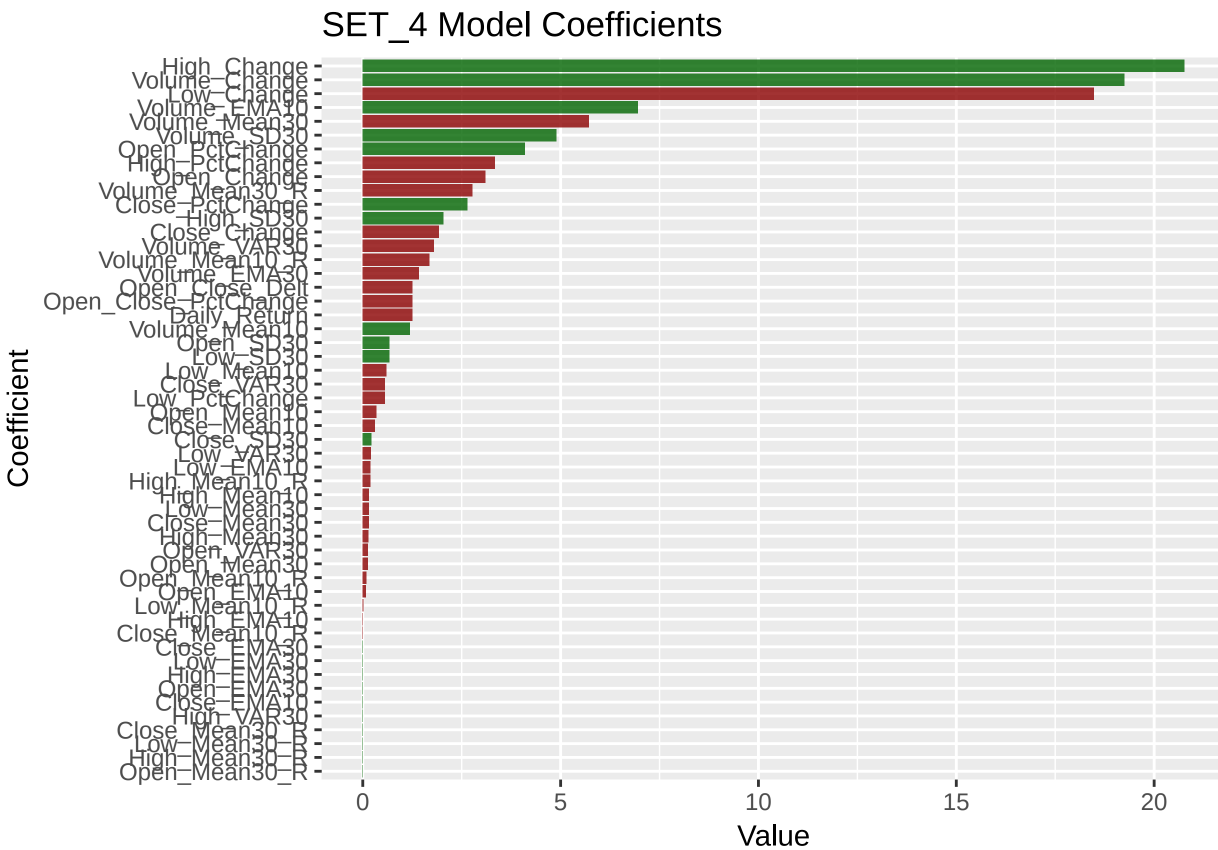



```
## [1] "SET_1 Model RMSE: 62.1731595679453"
```

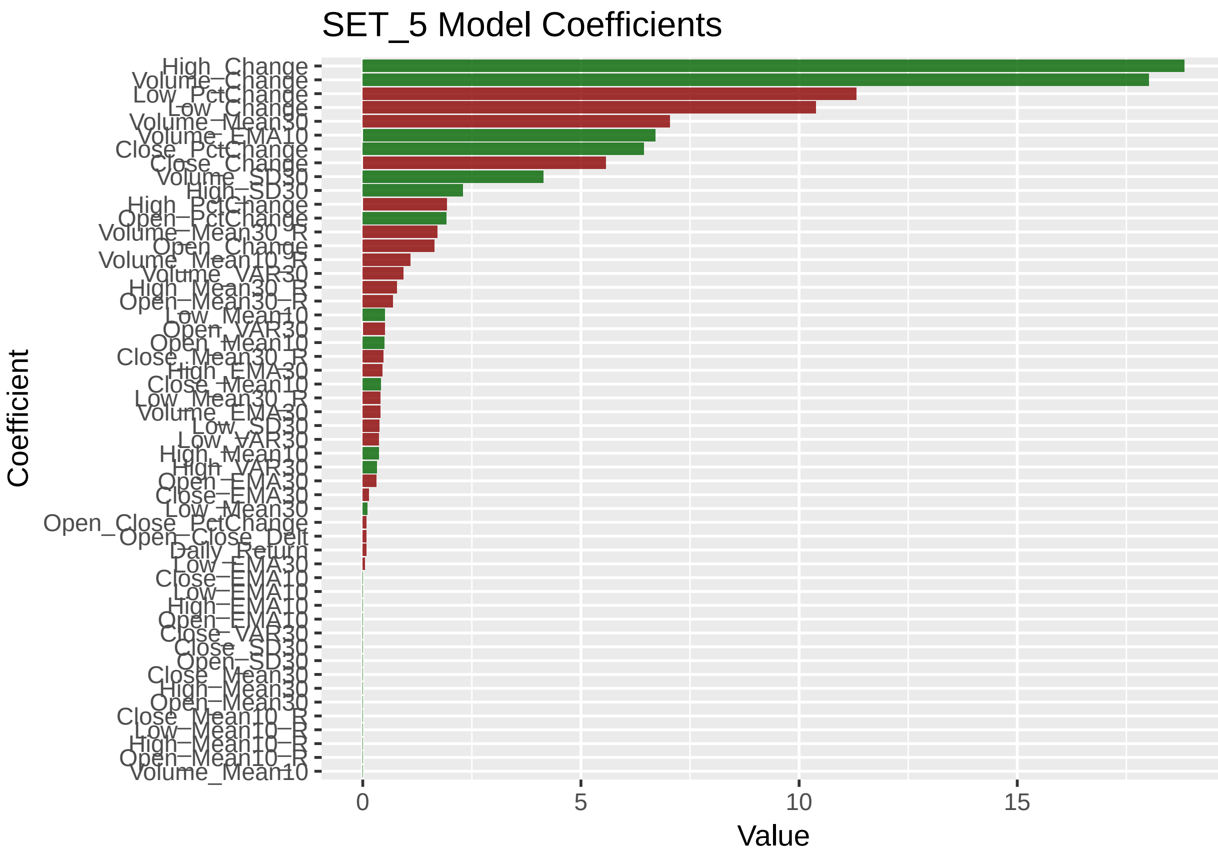

```
## [1] "SET_2 Model RMSE: 58.5796447975623"
```



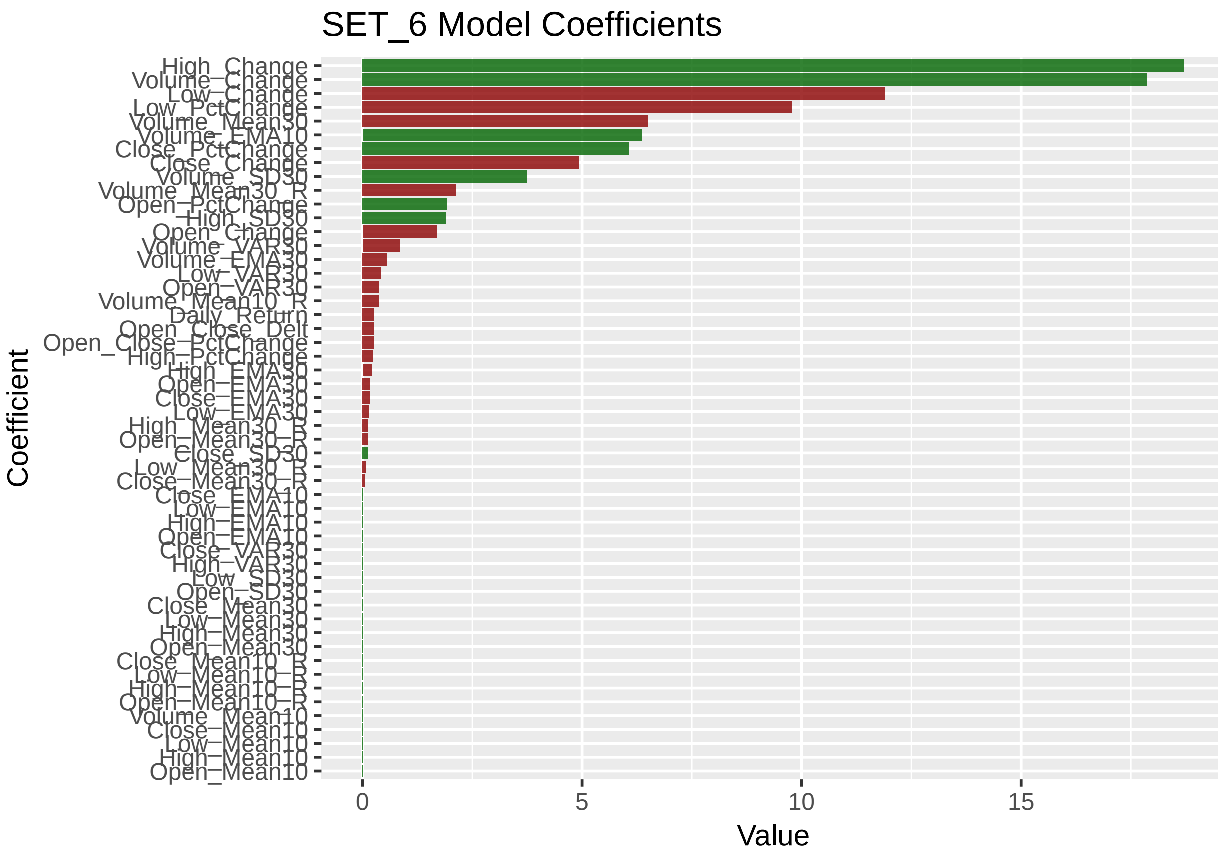

```
## [1] "SET_3 Model RMSE: 49.3713649004864"
```



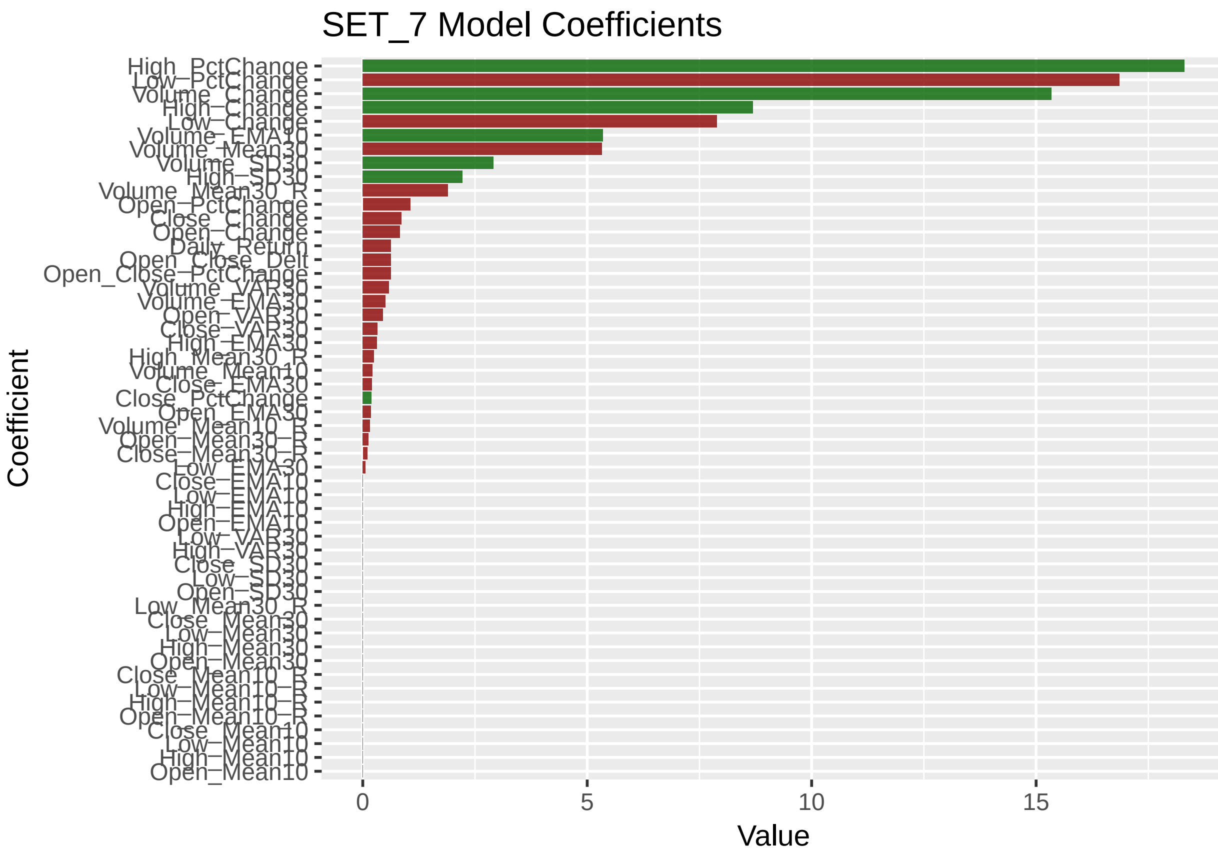

```
## [1] "SET_4 Model RMSE: 46.4744949580306"
```



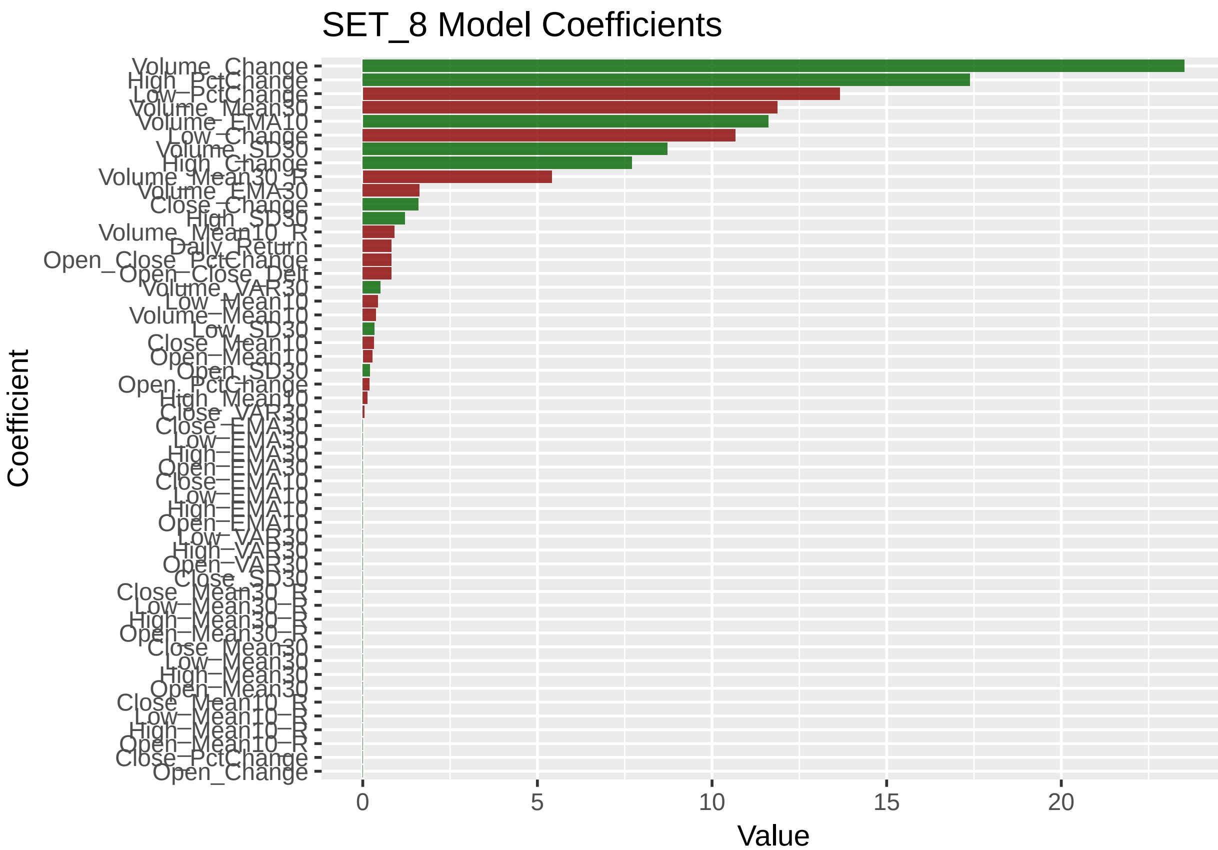

```
## [1] "SET_5 Model RMSE: 53.7597971088195"
```




```
## [1] "SET_6 Model RMSE: 54.2150841003213"
```




```
## [1] "SET_7 Model RMSE: 84.5398039202926"
```




```
## [1] "SET_8 Model RMSE: 567.724918266198"
```


```
## [1] "SET_9 Model RMSE: 62.4036070351572"
```

```
#View RMSE for each test set
#[1] "SET_1 Model RMSE: 62.1731609558704"
#[1] "SET_2 Model RMSE: 58.5796447975623"
#[1] "SET_3 Model RMSE: 49.3713649004864"
#[1] "SET_4 Model RMSE: 46.4744949580306"
#[1] "SET_5 Model RMSE: 53.7597971088195"
#[1] "SET_6 Model RMSE: 54.2150841003213"
#[1] "SET_7 Model RMSE: 84.5398039202926"
#[1] "SET_8 Model RMSE: 567.724918266198"
#[1] "SET_9 Model RMSE: 62.4036070351572"
#model appears unstable with Set_8 having a much larger RMSE

#Most important variables based on frequency across test sets
imp.var <- c(
"Low_Change",
"Volume_Change",
"Volume_EMA10",
"High_Change",
"Volume_Mean30",
"Low_PctChange",
"Open_Change",
"High_PctChange",
"Volume_SD30"
)
```

```
#change testSet sets to dataframes
test.set1 <- as.data.frame(testSet[["SET_1"]])
test.set2 <- as.data.frame(testSet[["SET_2"]])
test.set3 <- as.data.frame(testSet[["SET_3"]])
test.set4 <- as.data.frame(testSet[["SET_4"]])
test.set5 <- as.data.frame(testSet[["SET_5"]])
test.set6 <- as.data.frame(testSet[["SET_6"]])
test.set7 <- as.data.frame(testSet[["SET_7"]])
test.set8 <- as.data.frame(testSet[["SET_8"]])
test.set9 <- as.data.frame(testSet[["SET_9"]])

#change testList sets to dataframes
pred.set1 = as.data.frame(testList[["SET_1"]])
pred.set2 = as.data.frame(testList[["SET_2"]])
pred.set3 = as.data.frame(testList[["SET_3"]])
pred.set4 = as.data.frame(testList[["SET_4"]])
pred.set5 = as.data.frame(testList[["SET_5"]])
pred.set6 = as.data.frame(testList[["SET_6"]])
pred.set7 = as.data.frame(testList[["SET_7"]])
pred.set8 = as.data.frame(testList[["SET_8"]])
pred.set9 = as.data.frame(testList[["SET_9"]])

#attached predicted values to test dataframe
temp.1 <- cbind(test.set1, pred.set1)
temp.2 <- cbind(test.set2, pred.set2)
temp.3 <- cbind(test.set3, pred.set3)
temp.4 <- cbind(test.set4, pred.set4)
temp.5 <- cbind(test.set5, pred.set5)
temp.6 <- cbind(test.set6, pred.set6)
temp.7 <- cbind(test.set7, pred.set7)
temp.8 <- cbind(test.set8, pred.set8)
temp.9 <- cbind(test.set9, pred.set9)

pred.4 <-
rbind(temp.1,
temp.2,
temp.3,
temp.4,
temp.5,
temp.6,
temp.7,
temp.8,
temp.9)

#remove NA's
pred.4 <- pred.4[!is.na(pred.4$Volume_PctChange), ]

#ape computes the elementwise absolute percent difference
#between two numeric vectors
pred.4$APE <- ape(pred.4$Volume_PctChange, pred.4$predict)

# use the function to identify outliers
outliers <- FindOutliers(pred.4$APE)
```

```r
# remove non outliers
GLM.Outliers.4 <- pred.4[outliers, ]

#remove rows with APE of Inf
GLM.Outliers.4 <- GLM.Outliers.4[is.finite(GLM.Outliers.4$APE),]
```

6.1.5 Response = Volume % Change, Predictors = 9 Most Imp Variables

```r
#Response = Volume_PctChange
#Predictors = 9 Most Imp Variables from above model

# response and predictors to use
resp <- "Volume_PctChange"
pred <- imp.var

# iterating over the keys
modelList <- list()
testList <- list()

for (aKey in names(trainSet)) {
aTrain <- trainSet[[aKey]]
aTest <- testSet[[aKey]]

# make h2o data.frame, loads into H2O service
train.hex <- as.h2o(aTrain)
test.hex <- as.h2o(aTest)

# Summary
summary(train.hex, exact_quantiles = TRUE)
summary(test.hex, exact_quantiles = TRUE)

# train a model with that data
loop.glm = h2o.glm(
x = pred,
y = resp,
training_frame = train.hex,
validation_frame = test.hex,
family = "gaussian",
alpha = 0.5
)

# save the model
modelList[[aKey]] <- loop.glm

# predict response variable
glm.pred = h2o.predict(object = loop.glm, newdata = test.hex)

# save the prediction
testList[[aKey]] <- glm.pred
}
```

```r
#plot important variables for training set, list test set model RMSE
for (aKey in names(modelList)) {
mm <- modelList[[aKey]]

pp <- plotGLMVariableImportance(mm@model) +
labs(title = sprintf('%s Model Coefficients', aKey))
print(pp)

print(sprintf("%s Model RMSE: %s", aKey, h2o.rmse(mm, valid = T)))
}
```

```r
#View RMSE for each test set
#[1] "SET_1 Model RMSE: 62.0412626766534"
#[1] "SET_2 Model RMSE: 58.0499092924219"
#[1] "SET_3 Model RMSE: 48.9394510222991"
#[1] "SET_4 Model RMSE: 46.1845048377001"
#[1] "SET_5 Model RMSE: 53.995859242722"
#[1] "SET_6 Model RMSE: 51.4598686448757"
#[1] "SET_7 Model RMSE: 84.5250432394661"
#[1] "SET_8 Model RMSE: 544.773467068406"
#[1] "SET_9 Model RMSE: 61.7290832268858"
#model appears unstable with Set_8 having a much larger RMSE

#change testSet sets to dataframes
test.set1 <- as.data.frame(testSet[["SET_1"]])
test.set2 <- as.data.frame(testSet[["SET_2"]])
test.set3 <- as.data.frame(testSet[["SET_3"]])
test.set4 <- as.data.frame(testSet[["SET_4"]])
test.set5 <- as.data.frame(testSet[["SET_5"]])
test.set6 <- as.data.frame(testSet[["SET_6"]])
test.set7 <- as.data.frame(testSet[["SET_7"]])
test.set8 <- as.data.frame(testSet[["SET_8"]])
test.set9 <- as.data.frame(testSet[["SET_9"]])

#change testList sets to dataframes
pred.set1 = as.data.frame(testList[["SET_1"]])
pred.set2 = as.data.frame(testList[["SET_2"]])
pred.set3 = as.data.frame(testList[["SET_3"]])
pred.set4 = as.data.frame(testList[["SET_4"]])
pred.set5 = as.data.frame(testList[["SET_5"]])
pred.set6 = as.data.frame(testList[["SET_6"]])
pred.set7 = as.data.frame(testList[["SET_7"]])
pred.set8 = as.data.frame(testList[["SET_8"]])
pred.set9 = as.data.frame(testList[["SET_9"]])

#attached predicted values to test dataframe
temp.1 <- cbind(test.set1, pred.set1)
temp.2 <- cbind(test.set2, pred.set2)
temp.3 <- cbind(test.set3, pred.set3)
temp.4 <- cbind(test.set4, pred.set4)
temp.5 <- cbind(test.set5, pred.set5)
temp.6 <- cbind(test.set6, pred.set6)
temp.7 <- cbind(test.set7, pred.set7)
temp.8 <- cbind(test.set8, pred.set8)
```

```r
temp.9 <- cbind(test.set9, pred.set9)

pred.5 <-
rbind(temp.1,
temp.2,
temp.3,
temp.4,
temp.5,
temp.6,
temp.7,
temp.8,
temp.9)

#remove NA's
pred.5 <- pred.5[!is.na(pred.5$Volume_PctChange), ]

#ape computes the elementwise absolute percent difference
#between two numeric vectors
pred.5$APE <- ape(pred.5$Volume_PctChange, pred.5$predict)

# use the function to identify outliers
outliers <- FindOutliers(pred.5$APE)

# remove non outliers
GLM.Outliers.5 <- pred.5[outliers, ]

#remove rows with APE of Inf
GLM.Outliers.5 <- GLM.Outliers.5[is.finite(GLM.Outliers.5$APE),]
```

6.1.6 Response = Volume % Change, Predictors = 4 Most Imp Variables

```r
#Response = Volume_PctChange
#Predictors = 9 Most Imp Variables from above model

# response and predictors to use
resp <- "Volume_PctChange"
pred <- c("Low_Change",
"Volume_Change",
"Volume_EMA10",
"High_Change")

# iterating over the keys
modelList <- list()
testList <- list()

for (aKey in names(trainSet)) {
aTrain <- trainSet[[aKey]]
aTest <- testSet[[aKey]]

# make h2o data.frame, loads into H2O service
train.hex <- as.h2o(aTrain)
test.hex <- as.h2o(aTest)
```

```r
# Summary
summary(train.hex, exact_quantiles = TRUE)
summary(test.hex, exact_quantiles = TRUE)

# train a model with that data
loop.glm = h2o.glm(
x = pred,
y = resp,
training_frame = train.hex,
validation_frame = test.hex,
family = "gaussian",
alpha = 0.5
)

# save the model
modelList[[aKey]] <- loop.glm

# predict response variable
glm.pred = h2o.predict(object = loop.glm, newdata = test.hex)

# save the prediction
testList[[aKey]] <- glm.pred
}
```

```r
#plot important variables for training set, list test set model RMSE
for (aKey in names(modelList)) {
mm <- modelList[[aKey]]

pp <- plotGLMVariableImportance(mm@model) +
labs(title = sprintf('%s Model Coefficients', aKey))
print(pp)

print(sprintf("%s Model RMSE: %s", aKey, h2o.rmse(mm, valid = T)))
}
```

```r
#View RMSE for each test set
#[1] "SET_1 Model RMSE: 61.5955219449105"
#[1] "SET_2 Model RMSE: 58.029107585805"
#[1] "SET_3 Model RMSE: 48.823898111323"
#[1] "SET_4 Model RMSE: 45.9619200497954"
#[1] "SET_5 Model RMSE: 53.8579424135553"
#[1] "SET_6 Model RMSE: 51.5711715439961"
#[1] "SET_7 Model RMSE: 84.7546230270245"
#[1] "SET_8 Model RMSE: 51.8634475930439"
#[1] "SET_9 Model RMSE: 60.1618304678315"
#model appears stable with sith similar values of RMSE across test sets

#change testSet sets to dataframes
test.set1 <- as.data.frame(testSet[["SET_1"]])
test.set2 <- as.data.frame(testSet[["SET_2"]])
test.set3 <- as.data.frame(testSet[["SET_3"]])
test.set4 <- as.data.frame(testSet[["SET_4"]])
test.set5 <- as.data.frame(testSet[["SET_5"]])
test.set6 <- as.data.frame(testSet[["SET_6"]])
```

```r
test.set7 <- as.data.frame(testSet[["SET_7"]])
test.set8 <- as.data.frame(testSet[["SET_8"]])
test.set9 <- as.data.frame(testSet[["SET_9"]])

#change testList sets to dataframes
pred.set1 = as.data.frame(testList[["SET_1"]])
pred.set2 = as.data.frame(testList[["SET_2"]])
pred.set3 = as.data.frame(testList[["SET_3"]])
pred.set4 = as.data.frame(testList[["SET_4"]])
pred.set5 = as.data.frame(testList[["SET_5"]])
pred.set6 = as.data.frame(testList[["SET_6"]])
pred.set7 = as.data.frame(testList[["SET_7"]])
pred.set8 = as.data.frame(testList[["SET_8"]])
pred.set9 = as.data.frame(testList[["SET_9"]])

#attached predicted values to test dataframe
temp.1 <- cbind(test.set1, pred.set1)
temp.2 <- cbind(test.set2, pred.set2)
temp.3 <- cbind(test.set3, pred.set3)
temp.4 <- cbind(test.set4, pred.set4)
temp.5 <- cbind(test.set5, pred.set5)
temp.6 <- cbind(test.set6, pred.set6)
temp.7 <- cbind(test.set7, pred.set7)
temp.8 <- cbind(test.set8, pred.set8)
temp.9 <- cbind(test.set9, pred.set9)

pred.6 <-
rbind(temp.1,
temp.2,
temp.3,
temp.4,
temp.5,
temp.6,
temp.7,
temp.8,
temp.9)

#remove NA's
pred.6 <- pred.6[!is.na(pred.6$Volume_PctChange), ]

#ape computes the elementwise absolute percent difference
#between two numeric vectors
pred.6$APE <- ape(pred.6$Volume_PctChange, pred.6$predict)

# use the function to identify outliers
outliers <- FindOutliers(pred.6$APE)

# remove non outliers
GLM.Outliers.6 <- pred.6[outliers, ]

#remove rows with APE of Inf
GLM.Outliers.6 <- GLM.Outliers.6[is.finite(GLM.Outliers.6$APE),]
```

6.1.7 Outliers

```
#Best model for Open Close % Change used 4 variables
kable(head(GLM.Outliers.3[, 1:7])) %>% kable_styling(latex_options = "scale_down")
```

	Symbol	Date	Open	High	Low	Close	Volume
1988	AMG	2010-10-06	82.048	82.288	81.143	82.038	584040
3420	AZO	2010-11-09	243.050	243.660	242.050	243.000	328569
6030	CMI	2010-10-27	76.931	77.777	76.029	76.923	4684832
27998	WHR	2010-10-22	73.327	73.620	72.344	73.322	1061414
34893	CME	2011-02-09	45.368	46.444	44.954	45.370	5585956
35567	COST	2011-02-04	60.012	60.257	59.487	60.020	2314592

```
#Best model for Volume % Change used 4 variables
kable(head(GLM.Outliers.6[, 1:7])) %>% kable_styling(latex_options = "scale_down")
```

	Symbol	Date	Open	High	Low	Close	Volume
1042	AFL	2010-10-26	46.543	46.973	46.400	46.728	3450756
2178	AMT	2010-10-04	46.959	47.187	46.293	46.476	3219485
2349	ANDV	2010-12-03	15.277	15.916	15.277	15.636	6115589
2672	APA	2010-12-08	106.060	106.260	102.880	103.740	3643849
2745	APC	2010-12-21	63.264	64.963	63.084	64.906	3070781
3876	BEN	2010-11-19	33.825	34.105	33.406	33.734	2600877

```
#remove predict and APE columns
GLM.Price <- GLM.Outliers.3
GLM.Price$predict <- NULL
GLM.Price$APE <- NULL
GLM.Volume <- GLM.Outliers.6
GLM.Volume$predict <- NULL
GLM.Volume$APE <- NULL

#see if the outliers found with the best models for both
#Open Close % Change and Volume % Change overlap
common <- inner_join(GLM.Price, GLM.Volume)  #no rows in common
```

In order to detect possible outliers, I trained several GLM models for normal using Open Close % Change and Volume % Change as my response variables. Open Close % Change: Modeling for Open Close % Change resulted in 49 possible outliers as the actual Open Close % Change differed significantly from the value predicted by the GLM model. These data points required additional analysis to determine if the anomaly detection was correct. After, viewing the data points it is difficult to determine that any of the data points are true anomalies. For example, on 1/12/12 CMG's closing price was flat versus its opening price. However, my model predicted a 12% decline. It's difficult to say that my model is correct and that the stock should have traded down significantly instead of flat. Particularly given that the stock traded in a narrow band the week prior to 1/12/12 and the week after (-2% - +3%).

Volume % Change: Modeling for Volume % Change resulted in 203 possible outliers as the actual Volume % Change differed significantly from the value predicted by the GLM model. These data points required additional analysis to determine if the anomaly detection was correct. After, viewing the original data, I

found it difficult to determine that these data points were true outliers. For example, on 5/21/12 ISRG's volume was basically flat versus the prior day. However, my model predicted a 67% change. In the week prior to and after 5/21/12 ISRG's volume percent change ranged from -26% to +49%.
Interestingly, while modeling for normal there was no possible outlier data points in common between modeling for Open Close % Change and Volume % Change.

6.2 RF

```
#Start H2O
#h2o.init(nthreads = -1, max_mem_size = '8G')

# clean slate in case the cluster was already running
#h2o.removeAll()

# update outlier function as range used for GLM resulted
#in too few outliers
FindOutliers <- function(data) {
lowerq = quantile(data)[2]
upperq = quantile(data)[4]
iqr = upperq - lowerq
extreme.threshold.upper = (iqr * 200) + upperq
extreme.threshold.lower = lowerq - (iqr * 200)
result <-
which(data > extreme.threshold.upper |
data < extreme.threshold.lower)
}
```

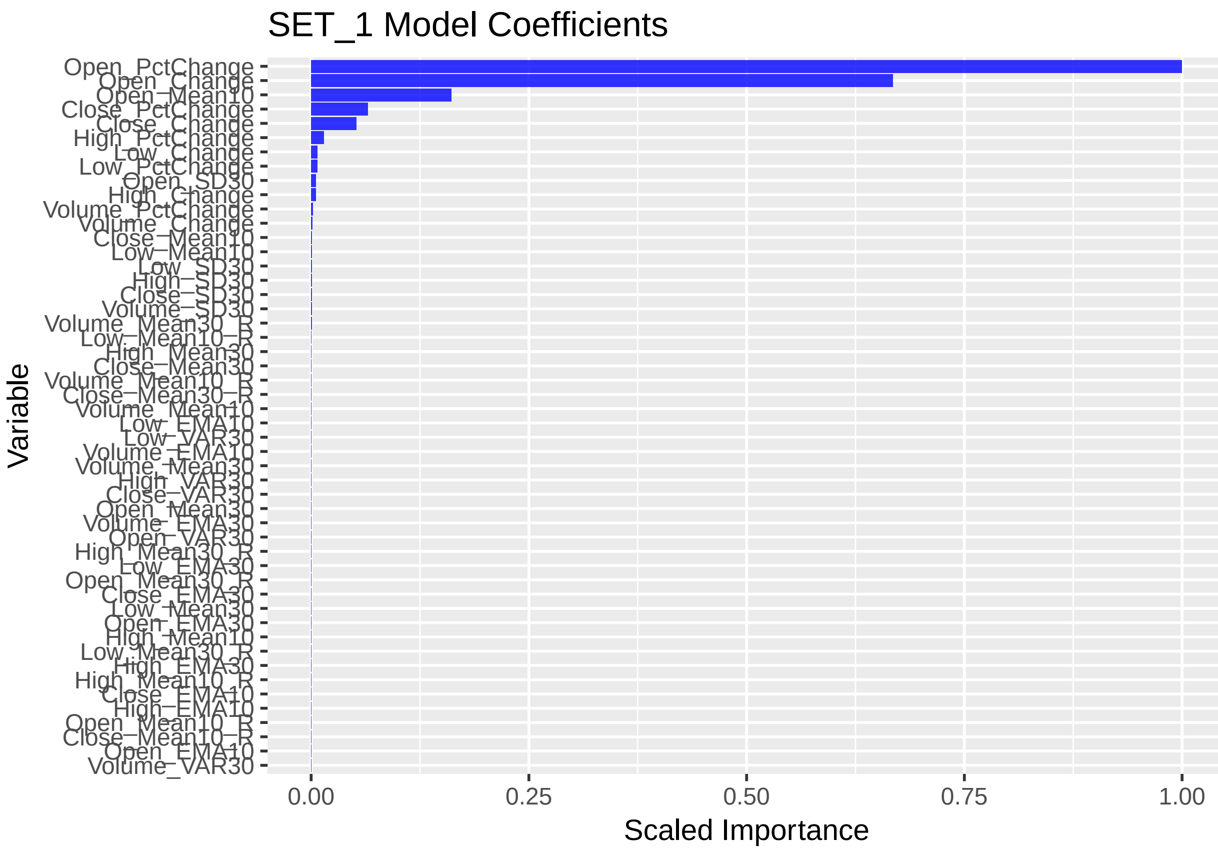
6.2.1 Response = Open Close % Change, Predictors = Most Variables

```
#Response = Open_Close_PctChange
#Predictors = All variables excluding Daily_Return,
#Open_Close_Delt ( due to high correlation).
# Also excludes Quarter, Year, Symbol, Sector, QtrYear,
#DOW, Name, which were dropped by the model.

# response and predictors to use
resp <- "Open_Close_PctChange"
pred <-
c(
"Open_Change",
"High_Change",
"Low_Change",
"Close_Change",
"Volume_Change",
"Open_PctChange",
"High_PctChange",
"Low_PctChange",
"Close_PctChange",
"Volume_PctChange",
"Open_Mean10",
"High_Mean10",
```

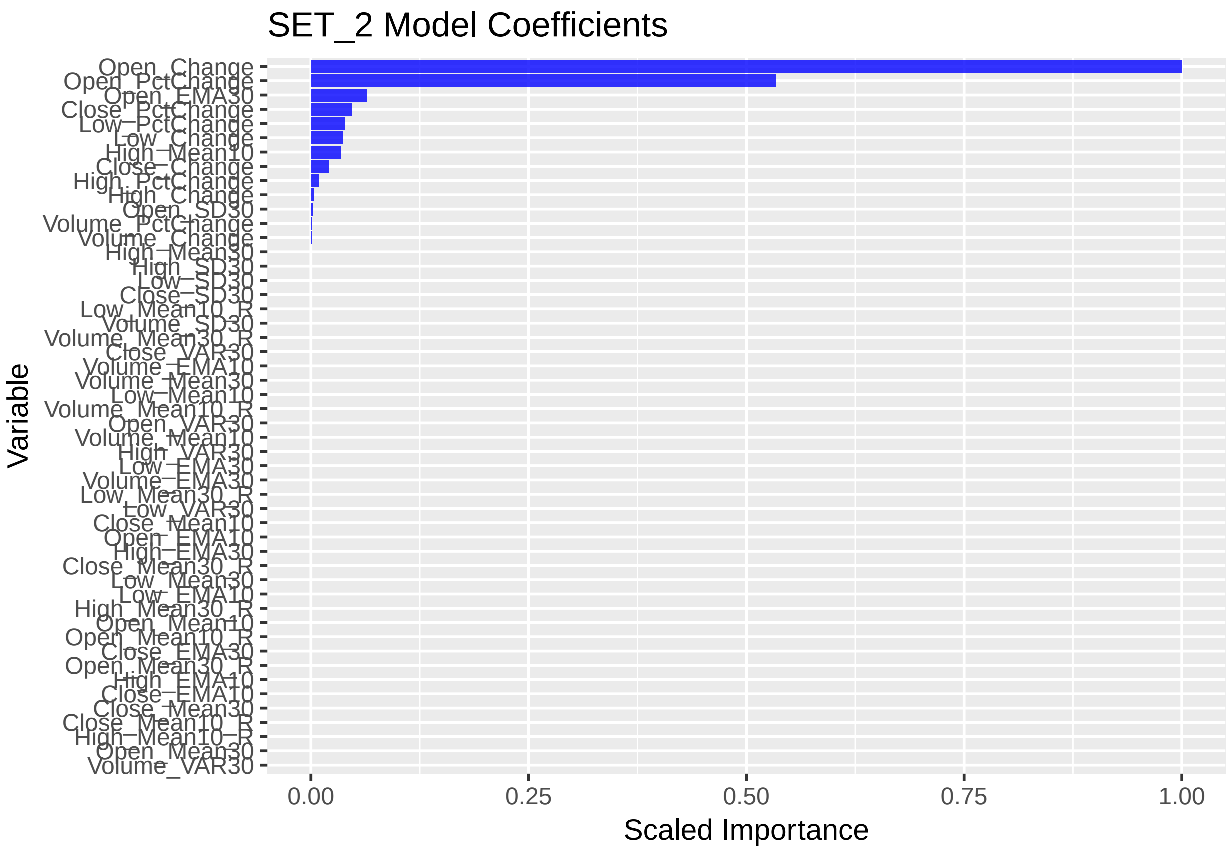
```r
"Low_Mean10",
"Close_Mean10",
"Volume_Mean10",
"Open_Mean10_R",
"High_Mean10_R",
"Low_Mean10_R",
"Close_Mean10_R",
"Volume_Mean10_R",
"Open_Mean30",
"High_Mean30",
"Low_Mean30",
"Close_Mean30",
"Volume_Mean30",
"Open_Mean30_R",
"High_Mean30_R",
"Low_Mean30_R",
"Close_Mean30_R",
"Volume_Mean30_R",
"Open_SD30",
"High_SD30",
"Low_SD30",
"Close_SD30",
"Volume_SD30",
"Open_VAR30",
"High_VAR30",
"Low_VAR30",
"Close_VAR30",
"Volume_VAR30",
"Open_EMA10",
"High_EMA10",
"Low_EMA10",
"Close_EMA10",
"Volume_EMA10",
"Open_EMA30",
"High_EMA30",
"Low_EMA30",
"Close_EMA30",
"Volume_EMA30"
)

# iterating over the keys
modelList <- list()
testList <- list()

for (aKey in names(trainSet)) {
aTrain <- trainSet[[aKey]]
aTest <- testSet[[aKey]]

# make h2o data.frame, loads into H2O service
train.hex <- as.h2o(aTrain)
test.hex <- as.h2o(aTest)

# Summary
```

```
summary(train.hex, exact_quantiles = TRUE)
summary(test.hex, exact_quantiles = TRUE)

# train a model with that data
loop.rf = h2o.randomForest(
x = pred,
y = resp,
training_frame = train.hex,
validation_frame = test.hex,
model_id = "rf_v1",
## name the model in H2O, helps use Flow
ntrees = 200,
## use a maximum of 200 trees to create the
##  random forest model. Will let
##  the early stopping criteria decide when
##  the random forest is sufficiently accurate
max_depth = 30,
stopping_rounds = 2,
## Stop fitting new trees when the 2-tree
##  average is within 0.001 (default) of
##  the prior two 2-tree averages.
##  Can be thought of as a convergence setting
seed = 123 ## Set the random seed so that this can be reproduced
)

# save the model
modelList[[aKey]] <- loop.rf

# predict response variable
rf.pred = h2o.predict(object = loop.rf, newdata = test.hex)

# save the prediction
testList[[aKey]] <- rf.pred
}
```

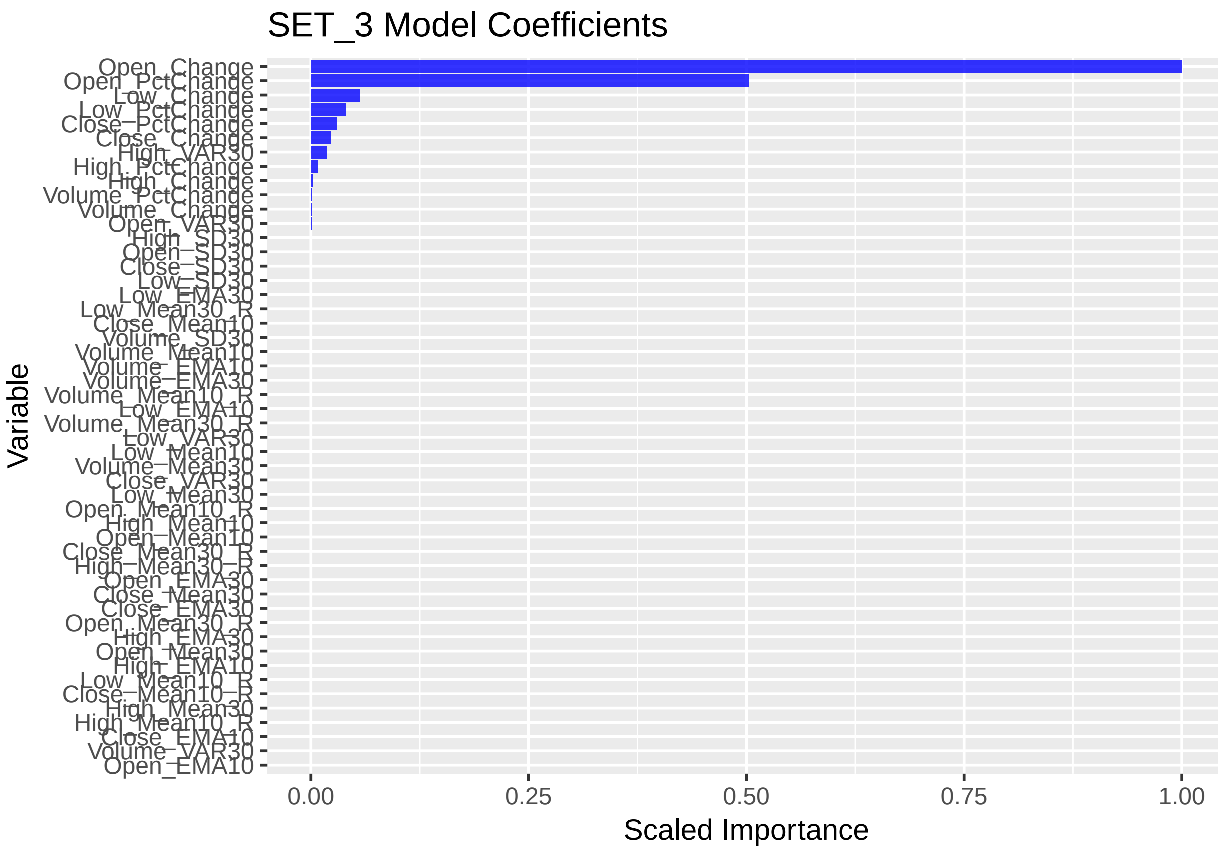
```
#plot important variables for training set, list test set model RMSE
for (aKey in names(modelList)) {
mm <- modelList[[aKey]]

pp <- plotRFVariableImportance(mm@model) +
labs(title = sprintf('%s Model Coefficients', aKey))
print(pp)

print(sprintf("%s Model RMSE: %s", aKey, h2o.rmse(mm, valid = T)))

}
```



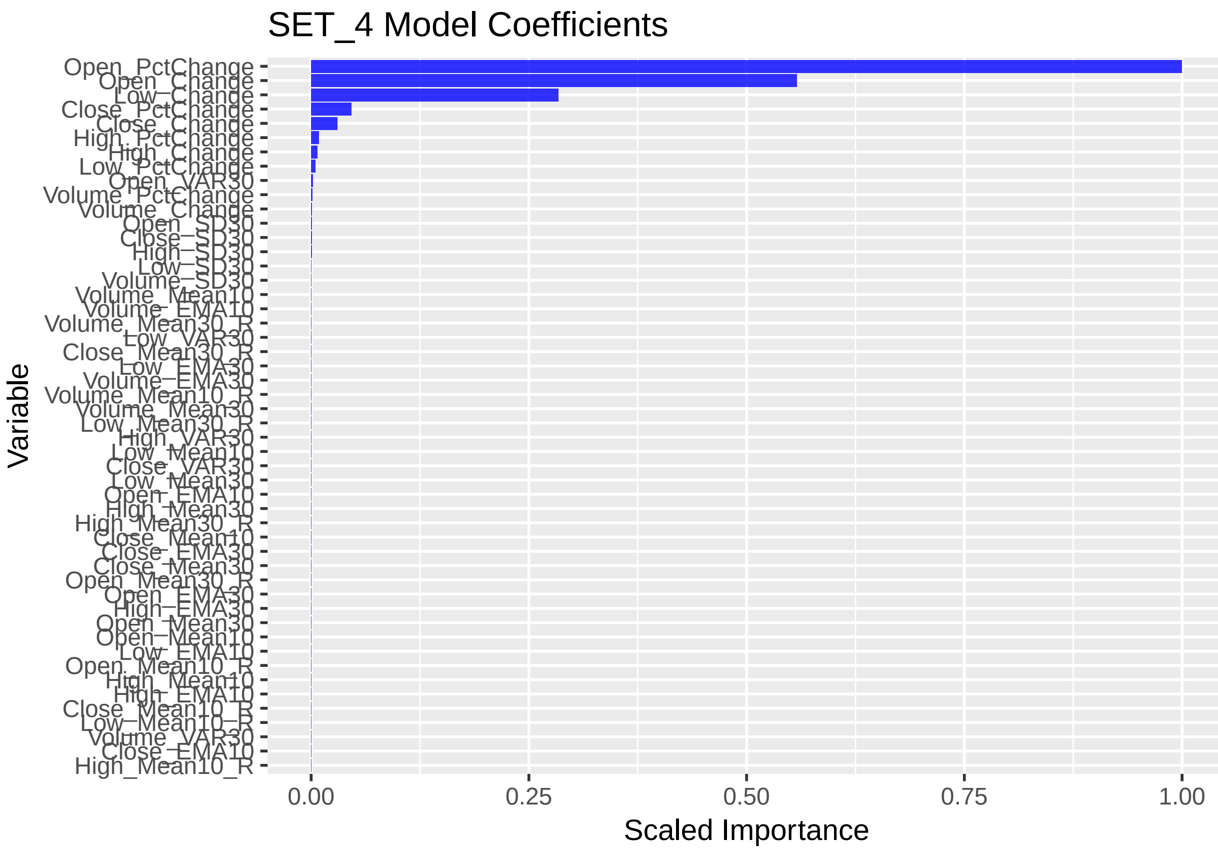

```
## [1] "SET_1 Model RMSE: 0.820505123412724"
```



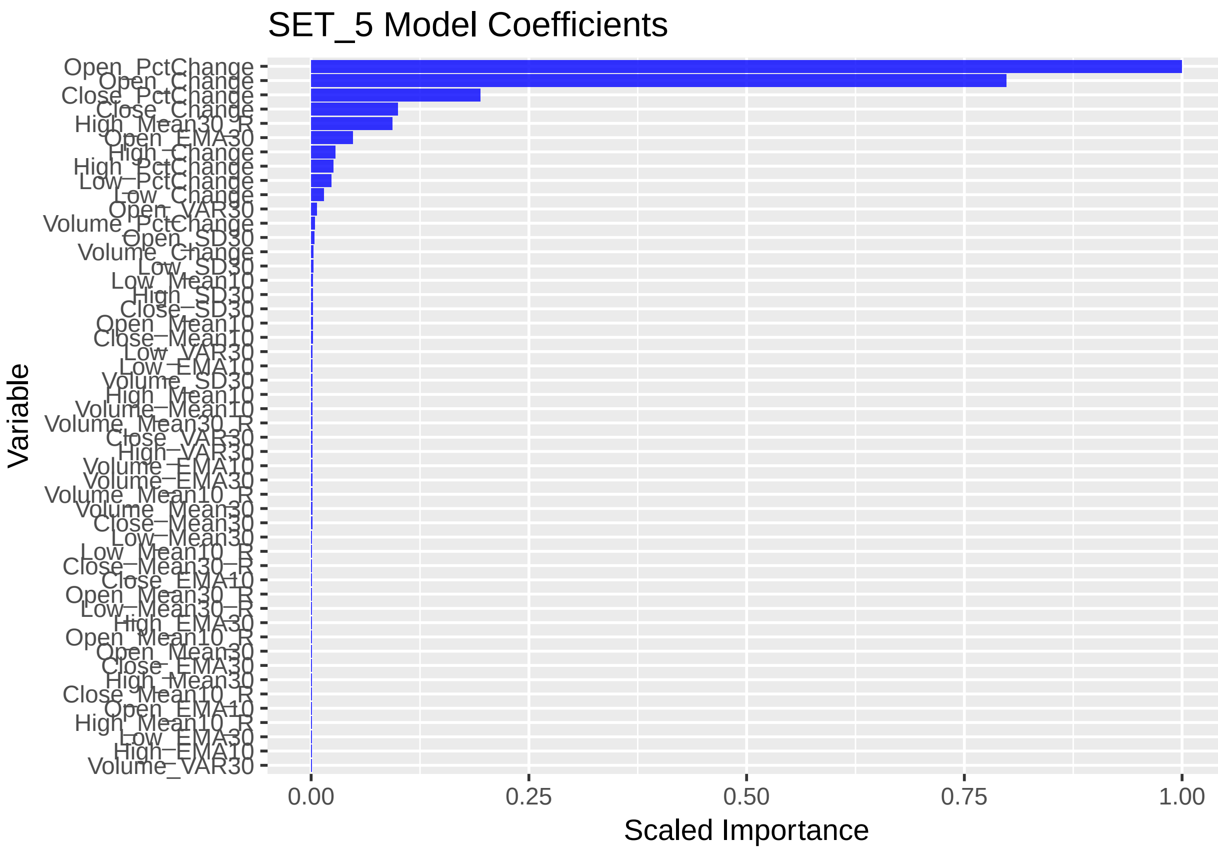

```
## [1] "SET_2 Model RMSE: 3.08372209382762"
```



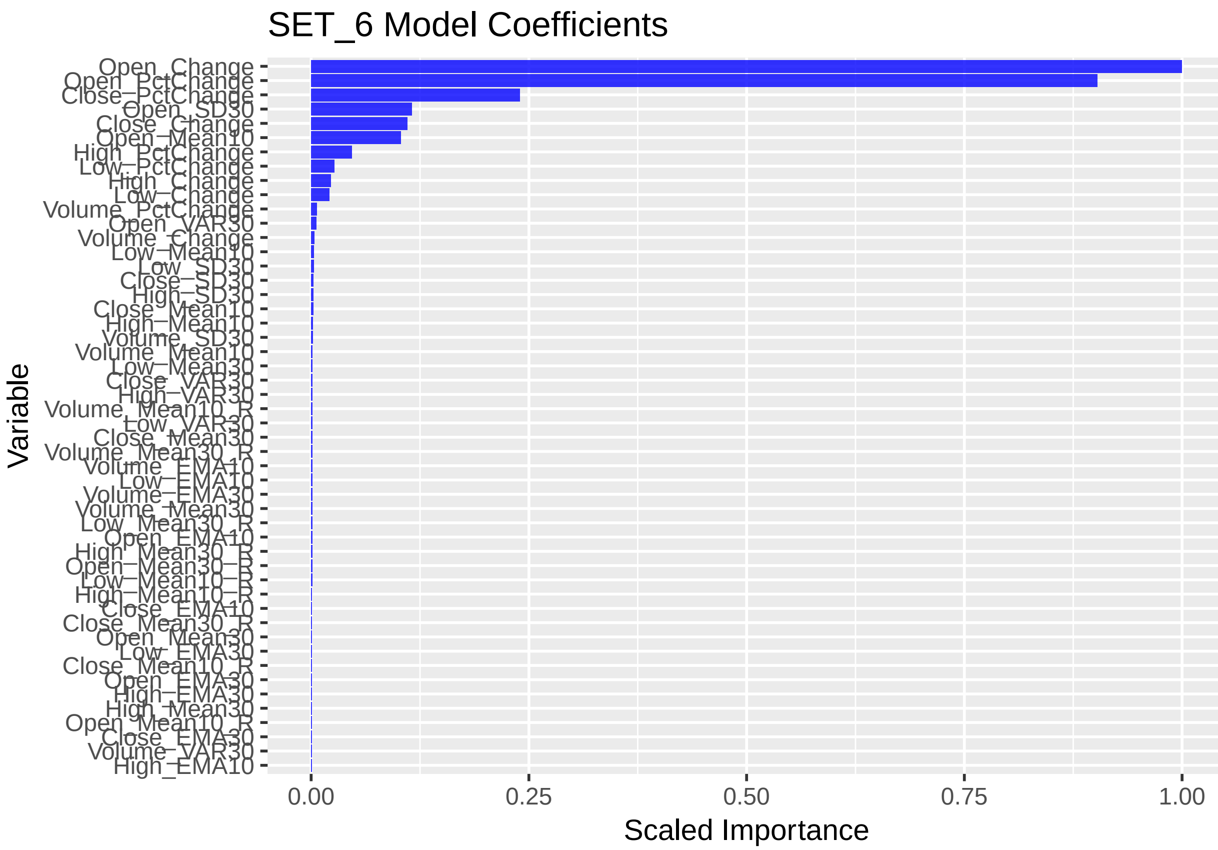

```
## [1] "SET_3 Model RMSE: 0.630740301508796"
```



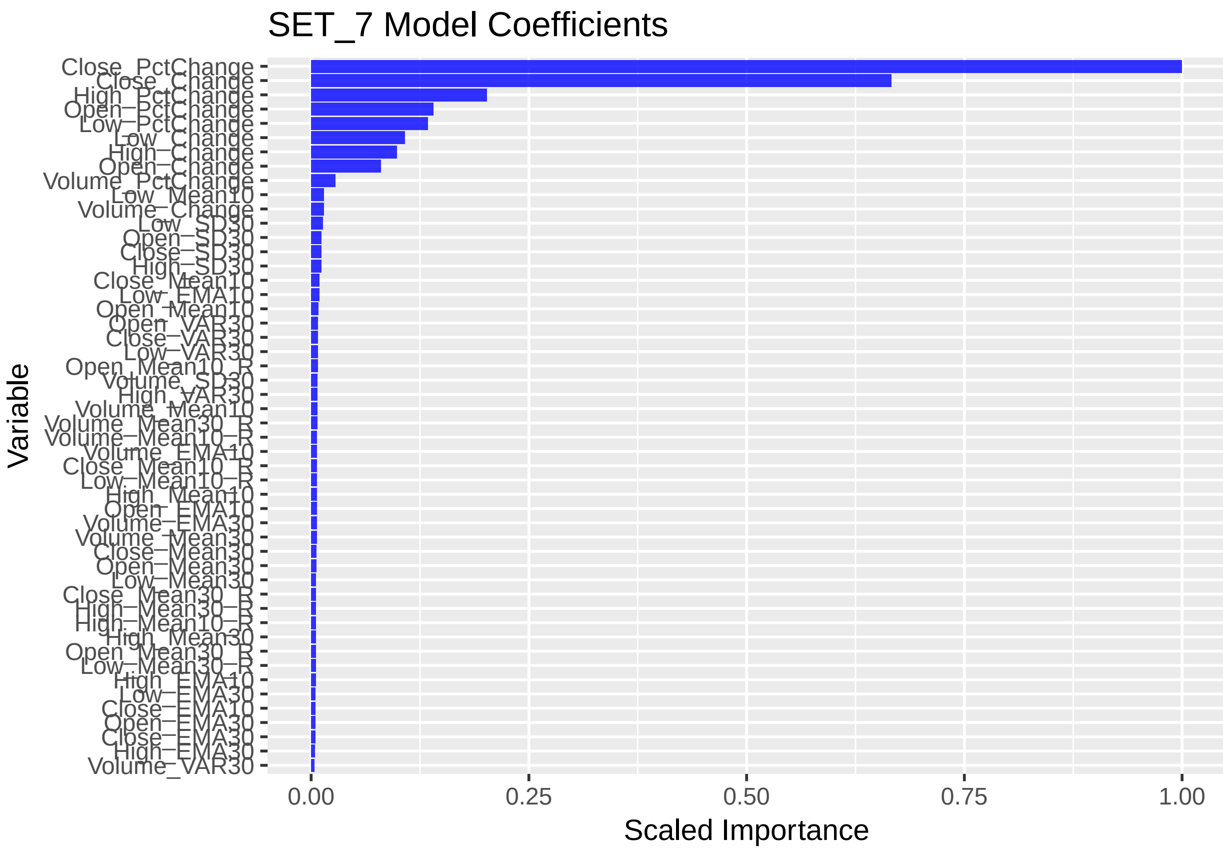

```
## [1] "SET_4 Model RMSE: 1.00874706953839"
```



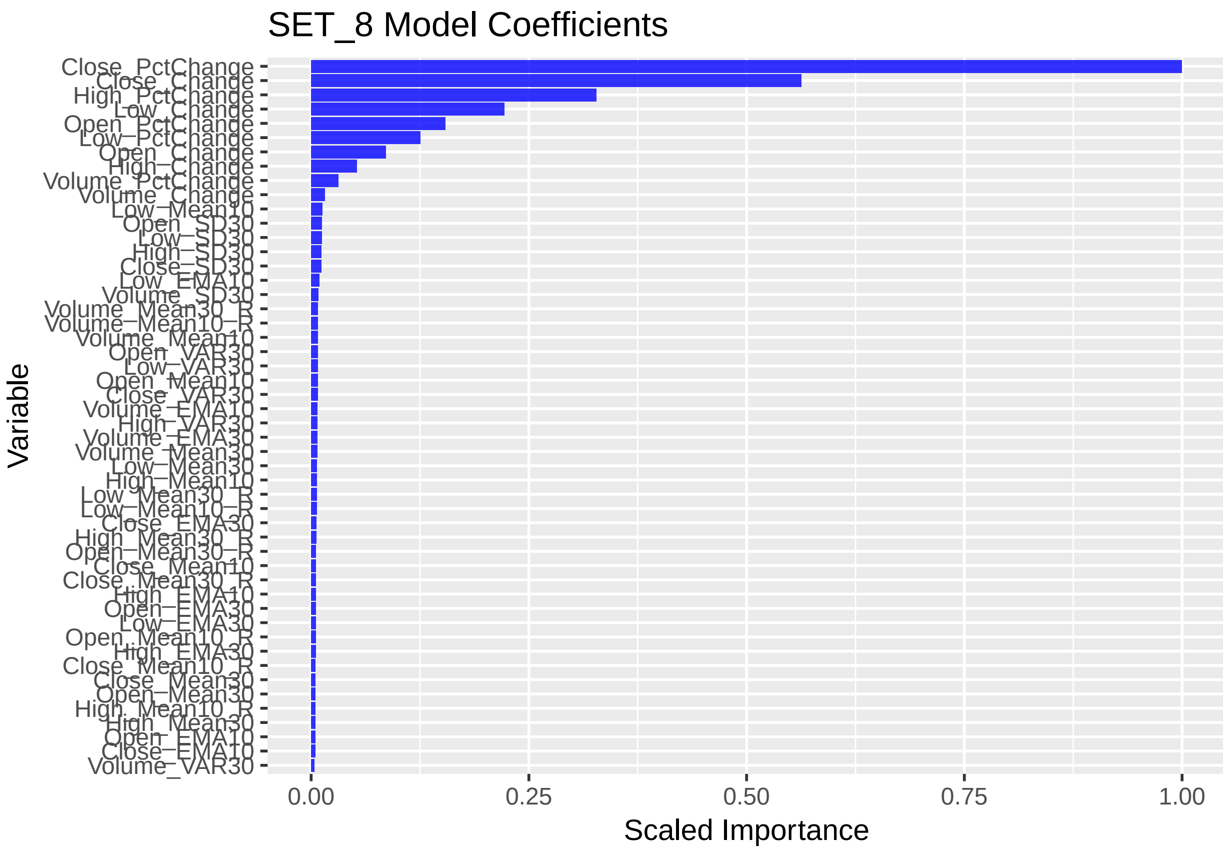

```
## [1] "SET_5 Model RMSE: 0.87697035443454"
```



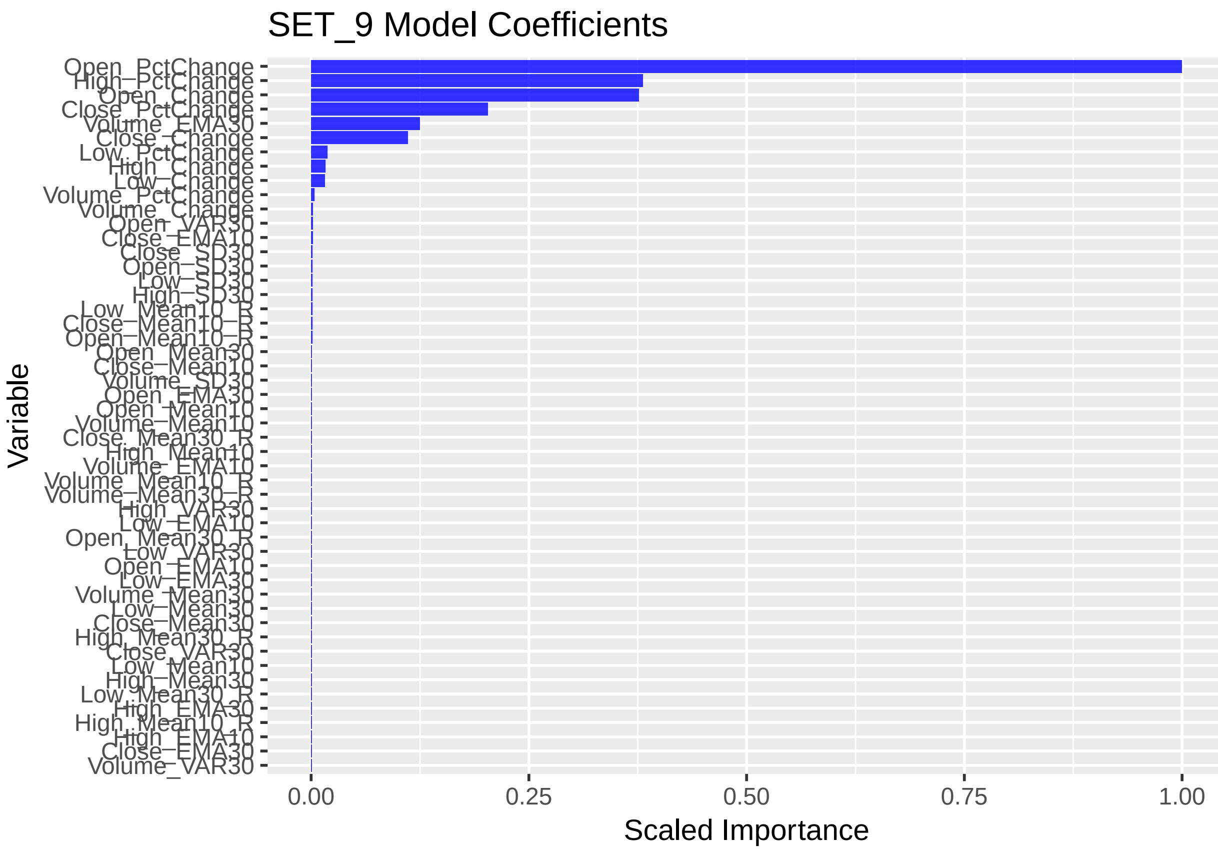

```
## [1] "SET_6 Model RMSE: 0.905195699792568"
```




```
## [1] "SET_7 Model RMSE: 0.637285048313381"
```




```
## [1] "SET_8 Model RMSE: 5.21842058153798"
```




```
## [1] "SET_9 Model RMSE: 2.43825778966936"
```

```r
#View RMSE for each test set
#[1] "SET_1 Model RMSE: 0.820505123412724"
#[1] "SET_2 Model RMSE: 3.08372209382762"
#[1] "SET_3 Model RMSE: 0.747484634264602"
#[1] "SET_4 Model RMSE: 0.99991652105645"
#[1] "SET_5 Model RMSE: 0.87697035443454"
#[1] "SET_6 Model RMSE: 0.905195699792568"
#[1] "SET_7 Model RMSE: 0.63339027514153"
#[1] "SET_8 Model RMSE: 5.21245650339379"
#[1] "SET_9 Model RMSE: 2.43825778966936"
#model appears relatively stable with RMSE range not too broad

#Most important variables based on frequency across test sets
imp.var <- c(
"Close_PctChange",
"Open_PctChange",
"Open_Change",
"Close_Change",
"Low_PctChange",
"High_PctChange",
"High_Change",
"Low_Change",
"Open_EMA30",
"Open_Mean10",
"Close_EMA10",
```

```r
"High_Mean30_R",
"High_SD30",
"High_VAR30",
"Open_SD30",
"Volume_EMA30"
)

#change testSet sets to dataframes
test.set1 <- as.data.frame(testSet[["SET_1"]])
test.set2 <- as.data.frame(testSet[["SET_2"]])
test.set3 <- as.data.frame(testSet[["SET_3"]])
test.set4 <- as.data.frame(testSet[["SET_4"]])
test.set5 <- as.data.frame(testSet[["SET_5"]])
test.set6 <- as.data.frame(testSet[["SET_6"]])
test.set7 <- as.data.frame(testSet[["SET_7"]])
test.set8 <- as.data.frame(testSet[["SET_8"]])
test.set9 <- as.data.frame(testSet[["SET_9"]])

#change testList sets to dataframes
pred.set1 = as.data.frame(testList[["SET_1"]])
pred.set2 = as.data.frame(testList[["SET_2"]])
pred.set3 = as.data.frame(testList[["SET_3"]])
pred.set4 = as.data.frame(testList[["SET_4"]])
pred.set5 = as.data.frame(testList[["SET_5"]])
pred.set6 = as.data.frame(testList[["SET_6"]])
pred.set7 = as.data.frame(testList[["SET_7"]])
pred.set8 = as.data.frame(testList[["SET_8"]])
pred.set9 = as.data.frame(testList[["SET_9"]])

#attached predicted values to test dataframe
temp.1 <- cbind(test.set1, pred.set1)
temp.2 <- cbind(test.set2, pred.set2)
temp.3 <- cbind(test.set3, pred.set3)
temp.4 <- cbind(test.set4, pred.set4)
temp.5 <- cbind(test.set5, pred.set5)
temp.6 <- cbind(test.set6, pred.set6)
temp.7 <- cbind(test.set7, pred.set7)
temp.8 <- cbind(test.set8, pred.set8)
temp.9 <- cbind(test.set9, pred.set9)

pred.rf.1 <-
rbind(temp.1,
temp.2,
temp.3,
temp.4,
temp.5,
temp.6,
temp.7,
temp.8,
temp.9)

#ape computes the elementwise absolute percent difference
#between two numeric vectors
```

```
pred.rf.1$APE <-
ape(pred.rf.1$Open_Close_PctChange, pred.rf.1$predict)

# use the function to identify outliers
outliers <- FindOutliers(pred.rf.1$APE)

# remove non outliers
RF.Outliers.1 <- pred.rf.1[outliers,]

#remove rows with APE of Inf
RF.Outliers.1 <- RF.Outliers.1[is.finite(RF.Outliers.1$APE), ]
```

6.2.2 Response = Open Close % Change, Predictors = Top 16 Most Imp Variables

```
#Response = Open_Close_PctChange
#Predictors = Top 16 most important variables found in model above.

# response and predictors to use
resp <- "Open_Close_PctChange"
pred <- imp.var

# iterating over the keys
modelList <- list()
testList <- list()

for (aKey in names(trainSet)) {
aTrain <- trainSet[[aKey]]
aTest <- testSet[[aKey]]

# make h2o data.frame, loads into H2O service
train.hex <- as.h2o(aTrain)
test.hex <- as.h2o(aTest)

# Summary
summary(train.hex, exact_quantiles = TRUE)
summary(test.hex, exact_quantiles = TRUE)

# train a model with that data
loop.rf = h2o.randomForest(
x = pred,
y = resp,
training_frame = train.hex,
validation_frame = test.hex,
model_id = "rf_v1",
## name the model in H2O, helps use Flow
ntrees = 200,
## use a maximum of 200 trees to create the
##  random forest model. Will let
##  the early stopping criteria decide when
##  the random forest is sufficiently accurate
max_depth = 30,
stopping_rounds = 2,
```

```r
## Stop fitting new trees when the 2-tree
##  average is within 0.001 (default) of
##  the prior two 2-tree averages.
##  Can be thought of as a convergence setting
seed = 123 ## Set the random seed so that this can be reproduced
)

# save the model
modelList[[aKey]] <- loop.rf

# predict response variable
rf.pred = h2o.predict(object = loop.rf, newdata = test.hex)

# save the prediction
testList[[aKey]] <- rf.pred
}
```

```r
#plot important variables for training set, list test set model RMSE
for (aKey in names(modelList)) {
mm <- modelList[[aKey]]

pp <- plotRFVariableImportance(mm@model) +
labs(title = sprintf('%s Model Coefficients', aKey))
print(pp)

print(sprintf("%s Model RMSE: %s", aKey, h2o.rmse(mm, valid = T)))

}
```

```r
#View RMSE for each test set
#[1] "SET_1 Model RMSE: 0.713180922319812"
#[1] "SET_2 Model RMSE: 2.08084422290107"
#[1] "SET_3 Model RMSE: 0.545479268766056"
#[1] "SET_4 Model RMSE: 1.00041860740511"
#[1] "SET_5 Model RMSE: 0.883301272237984"
#[1] "SET_6 Model RMSE: 0.912221906926535"
#[1] "SET_7 Model RMSE: 0.634151514111552"
#[1] "SET_8 Model RMSE: 5.21937302987135"
#[1] "SET_9 Model RMSE: 3.06858817551119"
#model appears relatively stable with RMSE range not too broad

#change testSet sets to dataframes
test.set1 <- as.data.frame(testSet[["SET_1"]])
test.set2 <- as.data.frame(testSet[["SET_2"]])
test.set3 <- as.data.frame(testSet[["SET_3"]])
test.set4 <- as.data.frame(testSet[["SET_4"]])
test.set5 <- as.data.frame(testSet[["SET_5"]])
test.set6 <- as.data.frame(testSet[["SET_6"]])
test.set7 <- as.data.frame(testSet[["SET_7"]])
test.set8 <- as.data.frame(testSet[["SET_8"]])
test.set9 <- as.data.frame(testSet[["SET_9"]])

#change testList sets to dataframes
pred.set1 = as.data.frame(testList[["SET_1"]])
```

```
pred.set2 = as.data.frame(testList[["SET_2"]])
pred.set3 = as.data.frame(testList[["SET_3"]])
pred.set4 = as.data.frame(testList[["SET_4"]])
pred.set5 = as.data.frame(testList[["SET_5"]])
pred.set6 = as.data.frame(testList[["SET_6"]])
pred.set7 = as.data.frame(testList[["SET_7"]])
pred.set8 = as.data.frame(testList[["SET_8"]])
pred.set9 = as.data.frame(testList[["SET_9"]])

#attached predicted values to test dataframe
temp.1 <- cbind(test.set1, pred.set1)
temp.2 <- cbind(test.set2, pred.set2)
temp.3 <- cbind(test.set3, pred.set3)
temp.4 <- cbind(test.set4, pred.set4)
temp.5 <- cbind(test.set5, pred.set5)
temp.6 <- cbind(test.set6, pred.set6)
temp.7 <- cbind(test.set7, pred.set7)
temp.8 <- cbind(test.set8, pred.set8)
temp.9 <- cbind(test.set9, pred.set9)

pred.rf.2 <-
rbind(temp.1,
temp.2,
temp.3,
temp.4,
temp.5,
temp.6,
temp.7,
temp.8,
temp.9)

#ape computes the elementwise absolute percent difference
#between two numeric vectors
pred.rf.2$APE <-
ape(pred.rf.2$Open_Close_PctChange, pred.rf.2$predict)

# use the function to identify outliers
outliers <- FindOutliers(pred.rf.2$APE)

# remove non outliers
RF.Outliers.2 <- pred.rf.2[outliers,]

#remove rows with APE of Inf
RF.Outliers.2 <- RF.Outliers.2[is.finite(RF.Outliers.2$APE), ]
```

6.2.3 Response = Open Close % Change, Predictors = 4 Most Imp Variables

```
#Response = Open_Close_PctChange
#Predictors = 4 most important variables found in the first model above.

# response and predictors to use
resp <- "Open_Close_PctChange"
```

```r
pred <- c("Close_PctChange",
"Open_PctChange",
"Open_Change",
"Close_Change")

# iterating over the keys
modelList <- list()
testList <- list()

for (aKey in names(trainSet)) {
aTrain <- trainSet[[aKey]]
aTest <- testSet[[aKey]]

# make h2o data.frame, loads into H2O service
train.hex <- as.h2o(aTrain)
test.hex <- as.h2o(aTest)

# Summary
summary(train.hex, exact_quantiles = TRUE)
summary(test.hex, exact_quantiles = TRUE)

# train a model with that data
loop.rf = h2o.randomForest(
x = pred,
y = resp,
training_frame = train.hex,
validation_frame = test.hex,
model_id = "rf_v1",
## name the model in H2O, helps use Flow
ntrees = 200,
## use a maximum of 200 trees to create the
##  random forest model. Will let
##  the early stopping criteria decide when
##  the random forest is sufficiently accurate
max_depth = 30,
stopping_rounds = 2,
## Stop fitting new trees when the 2-tree
##  average is within 0.001 (default) of
##  the prior two 2-tree averages.
##  Can be thought of as a convergence setting
seed = 123 ## Set the random seed so that this can be reproduced
)

# save the model
modelList[[aKey]] <- loop.rf

# predict response variable
rf.pred = h2o.predict(object = loop.rf, newdata = test.hex)

# save the prediction
testList[[aKey]] <- rf.pred
}
```

```r
#plot important variables for training set, list test set model RMSE
for (aKey in names(modelList)) {
mm <- modelList[[aKey]]

pp <- plotRFVariableImportance(mm@model) +
labs(title = sprintf('%s Model Coefficients', aKey))
print(pp)

print(sprintf("%s Model RMSE: %s", aKey, h2o.rmse(mm, valid = T)))

}
```

```
## [1] "SET_1 Model RMSE: 0.843478431113042"

## [1] "SET_2 Model RMSE: 3.14756053447266"

## [1] "SET_3 Model RMSE: 0.903396314235912"

## [1] "SET_4 Model RMSE: 1.1373127459668"

## [1] "SET_5 Model RMSE: 1.01549989961399"

## [1] "SET_6 Model RMSE: 0.979603261804701"

## [1] "SET_7 Model RMSE: 0.844338564039552"

## [1] "SET_8 Model RMSE: 5.20300161369571"

## [1] "SET_9 Model RMSE: 3.09994348631331"
```

```r
#View RMSE for each test set
#[1] "SET_1 Model RMSE: 0.843478431672689"
#[1] "SET_2 Model RMSE: 3.14756053447266"
#[1] "SET_3 Model RMSE: 0.89608677376429"
#[1] "SET_4 Model RMSE: 1.1373127459668"
#[1] "SET_5 Model RMSE: 1.01574403012868"
#[1] "SET_6 Model RMSE: 0.979603261804701"
#[1] "SET_7 Model RMSE: 0.844338564039552"
#[1] "SET_8 Model RMSE: 5.20300161369571"
#[1] "SET_9 Model RMSE: 3.09994348631331"
#model appears relatively stable with RMSE range not too broad

#change testSet sets to dataframes
test.set1 <- as.data.frame(testSet[["SET_1"]])
test.set2 <- as.data.frame(testSet[["SET_2"]])
test.set3 <- as.data.frame(testSet[["SET_3"]])
test.set4 <- as.data.frame(testSet[["SET_4"]])
test.set5 <- as.data.frame(testSet[["SET_5"]])
test.set6 <- as.data.frame(testSet[["SET_6"]])
test.set7 <- as.data.frame(testSet[["SET_7"]])
test.set8 <- as.data.frame(testSet[["SET_8"]])
test.set9 <- as.data.frame(testSet[["SET_9"]])

#change testList sets to dataframes
pred.set1 = as.data.frame(testList[["SET_1"]])
pred.set2 = as.data.frame(testList[["SET_2"]])
pred.set3 = as.data.frame(testList[["SET_3"]])
pred.set4 = as.data.frame(testList[["SET_4"]])
```

```r
pred.set5 = as.data.frame(testList[["SET_5"]])
pred.set6 = as.data.frame(testList[["SET_6"]])
pred.set7 = as.data.frame(testList[["SET_7"]])
pred.set8 = as.data.frame(testList[["SET_8"]])
pred.set9 = as.data.frame(testList[["SET_9"]])

#attached predicted values to test dataframe
temp.1 <- cbind(test.set1, pred.set1)
temp.2 <- cbind(test.set2, pred.set2)
temp.3 <- cbind(test.set3, pred.set3)
temp.4 <- cbind(test.set4, pred.set4)
temp.5 <- cbind(test.set5, pred.set5)
temp.6 <- cbind(test.set6, pred.set6)
temp.7 <- cbind(test.set7, pred.set7)
temp.8 <- cbind(test.set8, pred.set8)
temp.9 <- cbind(test.set9, pred.set9)

pred.rf.3 <-
rbind(temp.1,
temp.2,
temp.3,
temp.4,
temp.5,
temp.6,
temp.7,
temp.8,
temp.9)

#ape computes the elementwise absolute percent difference
#between two numeric vectors
pred.rf.3$APE <-
ape(pred.rf.3$Open_Close_PctChange, pred.rf.3$predict)

# use the function to identify outliers
outliers <- FindOutliers(pred.rf.3$APE)

# remove non outliers
RF.Outliers.3 <- pred.rf.3[outliers,]

#remove rows with APE of Inf
RF.Outliers.3 <- RF.Outliers.3[is.finite(RF.Outliers.3$APE), ]
```

6.2.4 Response = Volume % Change, Predictors = Most Variables

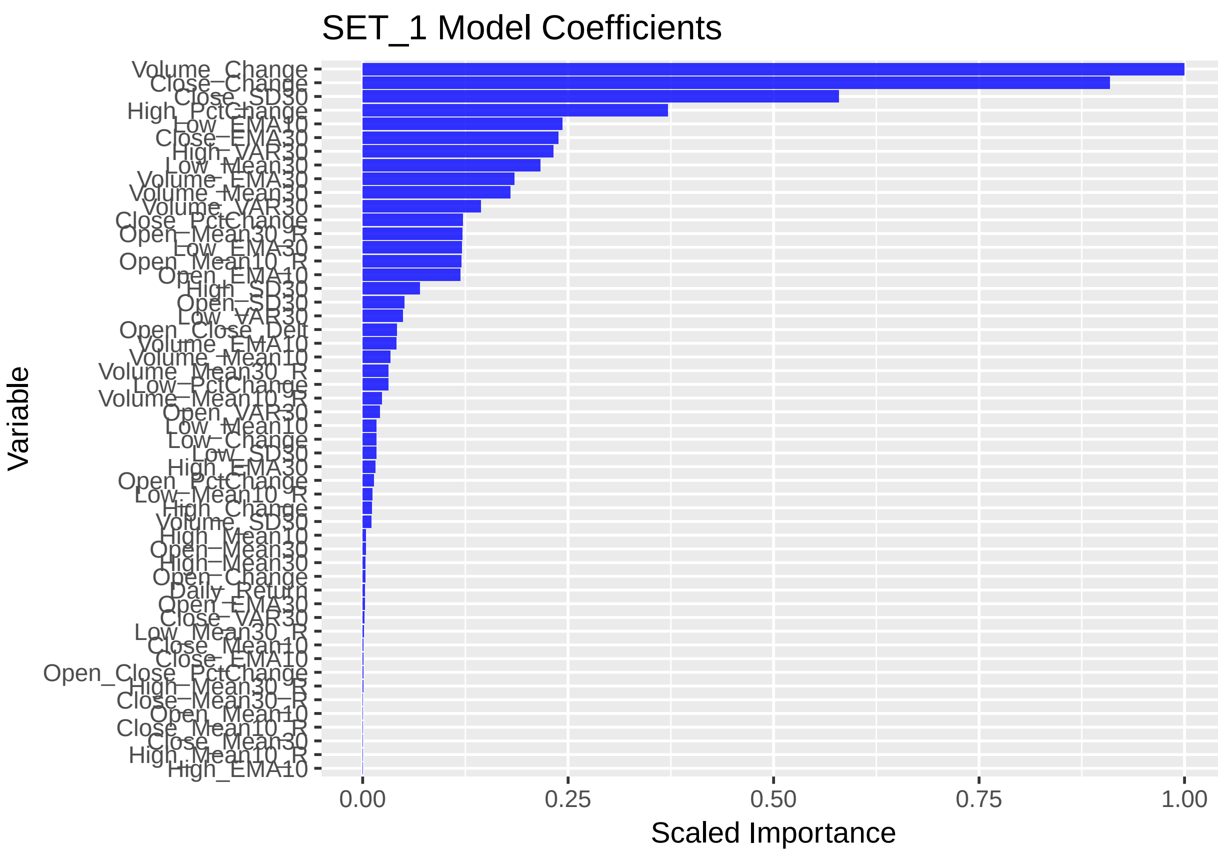
```r
#Response = Volume_PctChange
#Predictors = All variables excluding Quarter, Year,
#Symbol, Sector, QtrYear, DOW, Name, which were dropped by the model.

# response and predictors to use
resp <- "Volume_PctChange"
pred <-
c(
```

```
"Open_Close_PctChange",
"Daily_Return",
"Open_Close_Delt",
"Open_Change",
"High_Change",
"Low_Change",
"Close_Change",
"Volume_Change",
"Open_PctChange",
"High_PctChange",
"Low_PctChange",
"Close_PctChange",
"Volume_PctChange",
"Open_Mean10",
"High_Mean10",
"Low_Mean10",
"Close_Mean10",
"Volume_Mean10",
"Open_Mean10_R",
"High_Mean10_R",
"Low_Mean10_R",
"Close_Mean10_R",
"Volume_Mean10_R",
"Open_Mean30",
"High_Mean30",
"Low_Mean30",
"Close_Mean30",
"Volume_Mean30",
"Open_Mean30_R",
"High_Mean30_R",
"Low_Mean30_R",
"Close_Mean30_R",
"Volume_Mean30_R",
"Open_SD30",
"High_SD30",
"Low_SD30",
"Close_SD30",
"Volume_SD30",
"Open_VAR30",
"High_VAR30",
"Low_VAR30",
"Close_VAR30",
"Volume_VAR30",
"Open_EMA10",
"High_EMA10",
"Low_EMA10",
"Close_EMA10",
"Volume_EMA10",
"Open_EMA30",
"High_EMA30",
"Low_EMA30",
"Close_EMA30",
"Volume_EMA30"
```

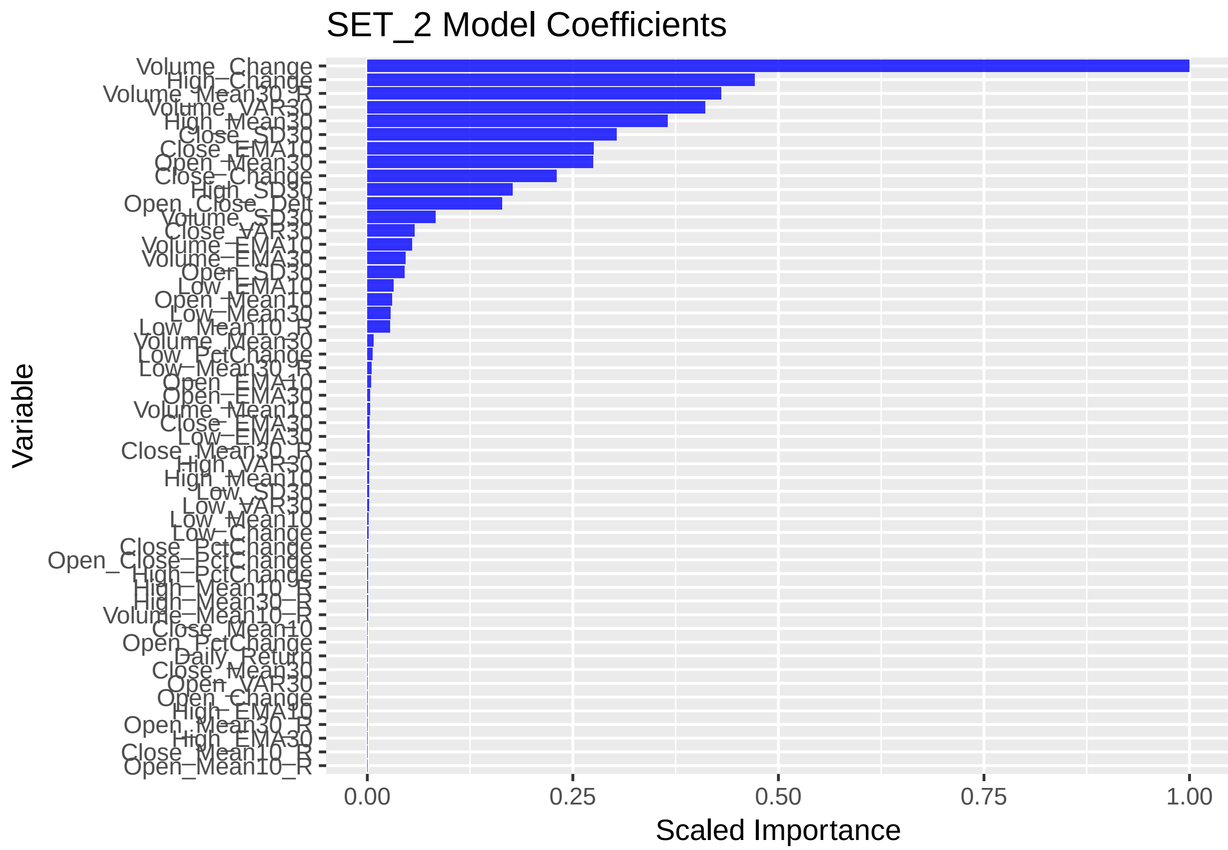
```
)

# iterating over the keys
modelList <- list()
testList <- list()

for (aKey in names(trainSet)) {
aTrain <- trainSet[[aKey]]
aTest <- testSet[[aKey]]

# make h2o data.frame, loads into H2O service
train.hex <- as.h2o(aTrain)
test.hex <- as.h2o(aTest)

# Summary
summary(train.hex, exact_quantiles = TRUE)
summary(test.hex, exact_quantiles = TRUE)

# train a model with that data
loop.rf = h2o.randomForest(
x = pred,
y = resp,
training_frame = train.hex,
validation_frame = test.hex,
model_id = "rf_v1",
## name the model in H2O, helps use Flow
ntrees = 200,
## use a maximum of 200 trees to create the
##  random forest model. Will let
##  the early stopping criteria decide when
##  the random forest is sufficiently accurate
max_depth = 30,
stopping_rounds = 2,
## Stop fitting new trees when the 2-tree
##  average is within 0.001 (default) of
##  the prior two 2-tree averages.
##  Can be thought of as a convergence setting
seed = 123 ## Set the random seed so that this can be reproduced
)

# save the model
modelList[[aKey]] <- loop.rf

# predict response variable
rf.pred = h2o.predict(object = loop.rf, newdata = test.hex)

# save the prediction
testList[[aKey]] <- rf.pred
}
```

```
#plot important variables for training set, list test set model RMSE
for (aKey in names(modelList)) {
mm <- modelList[[aKey]]
```

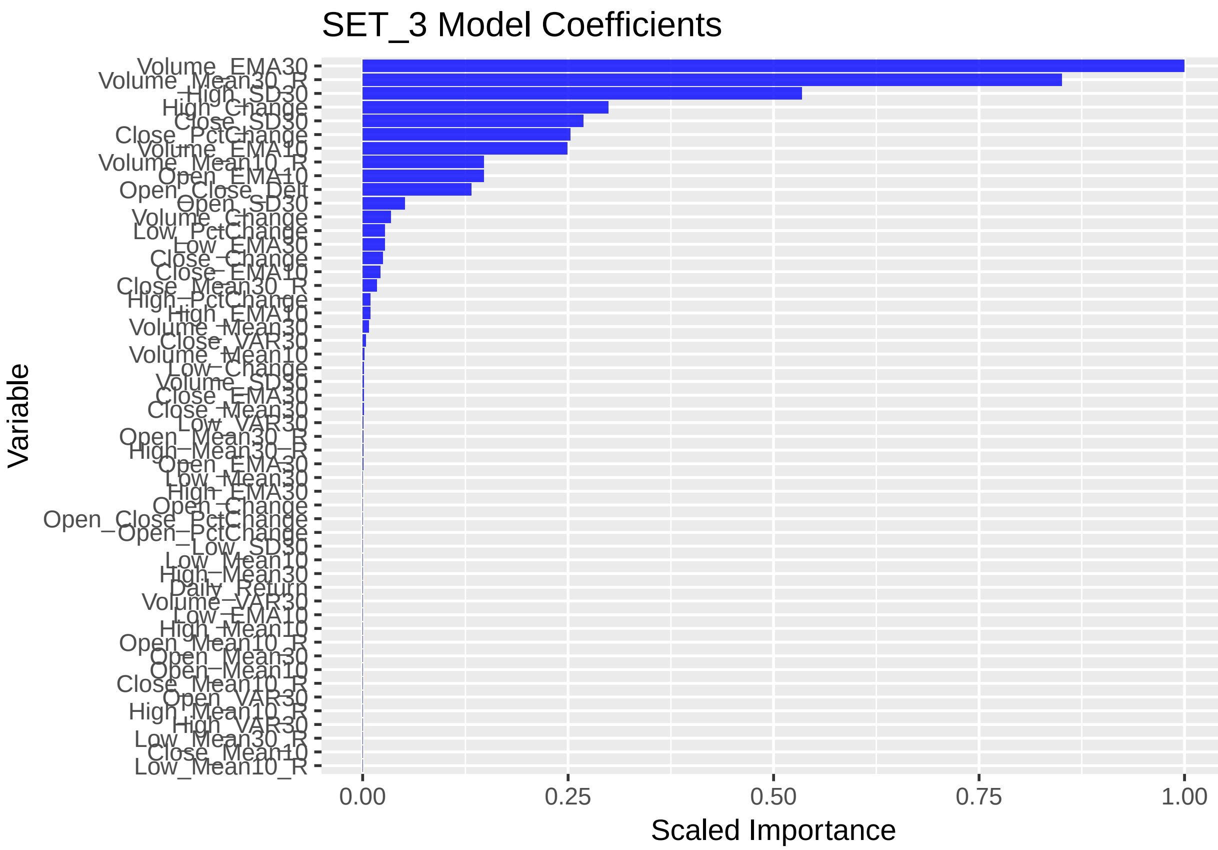
```r
pp <- plotRFVariableImportance(mm@model) +
labs(title = sprintf('%s Model Coefficients', aKey))
print(pp)

print(sprintf("%s Model RMSE: %s", aKey, h2o.rmse(mm, valid = T)))

}
```



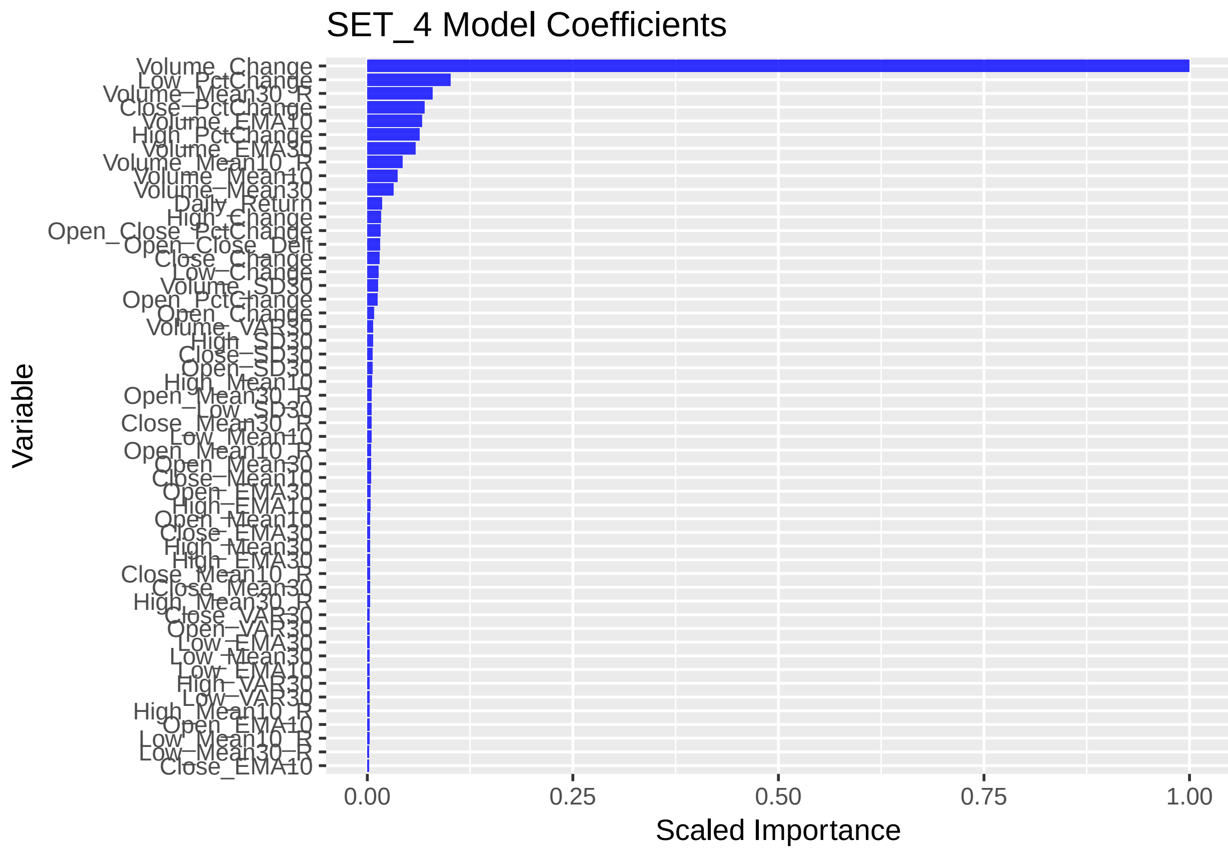

```
## [1] "SET_1 Model RMSE: 576.692237061663"
```



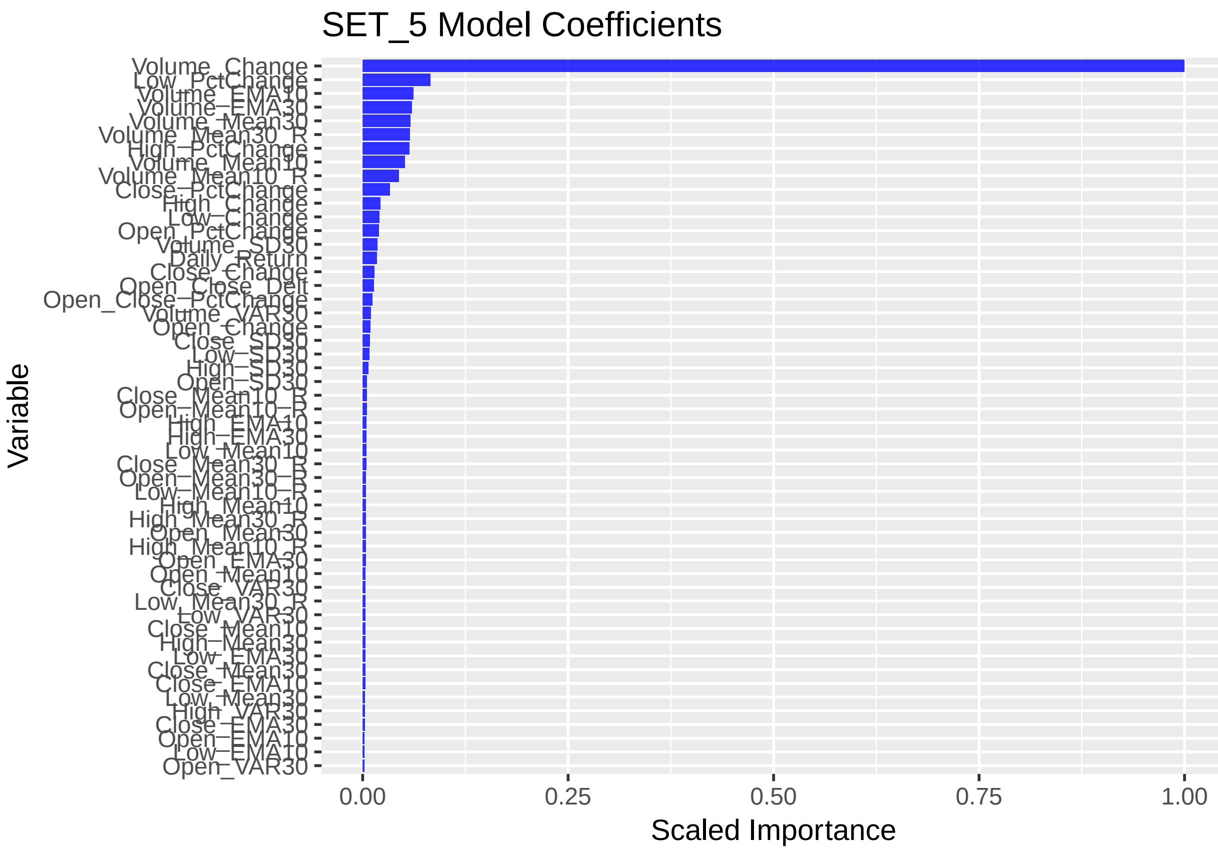

```
## [1] "SET_2 Model RMSE: 110.902571585891"
```



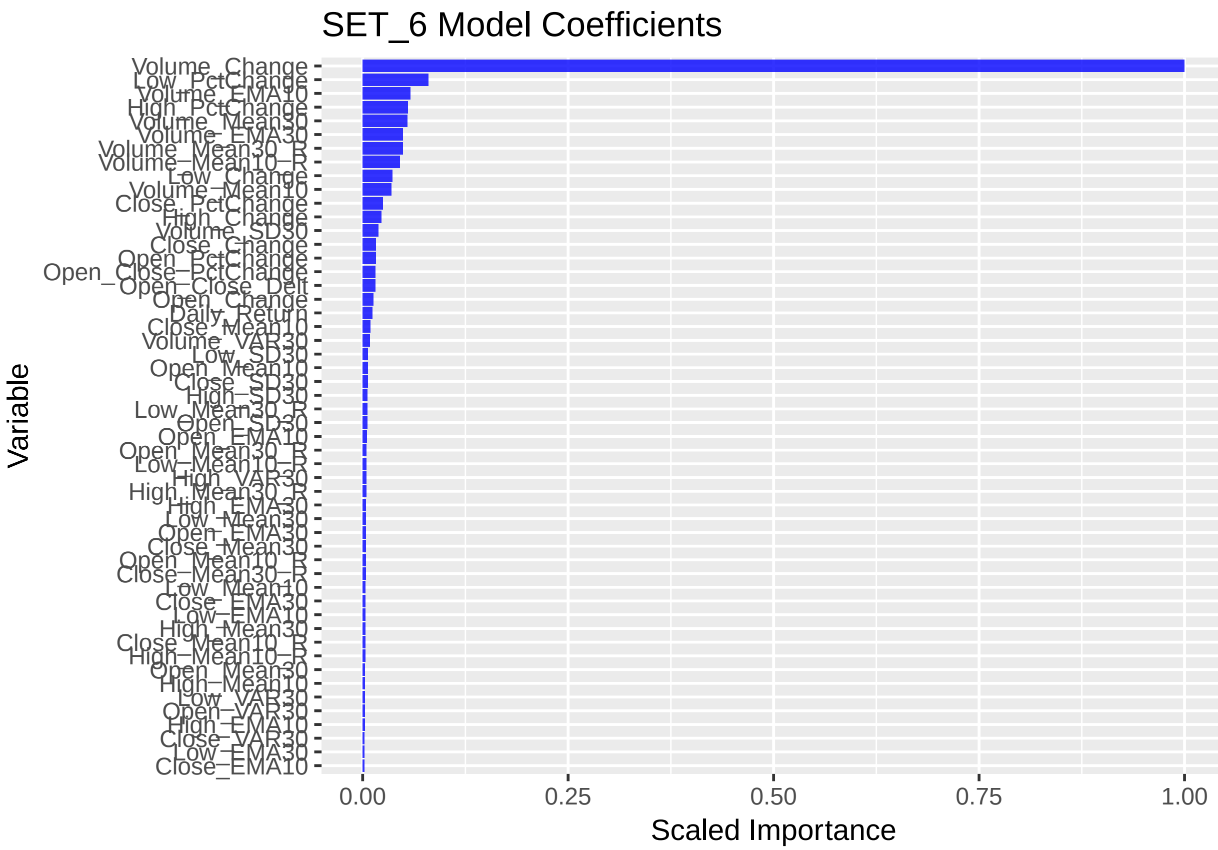

```
## [1] "SET_3 Model RMSE: 25.2511196082064"
```




```
## [1] "SET_4 Model RMSE: 32.3291983290633"
```

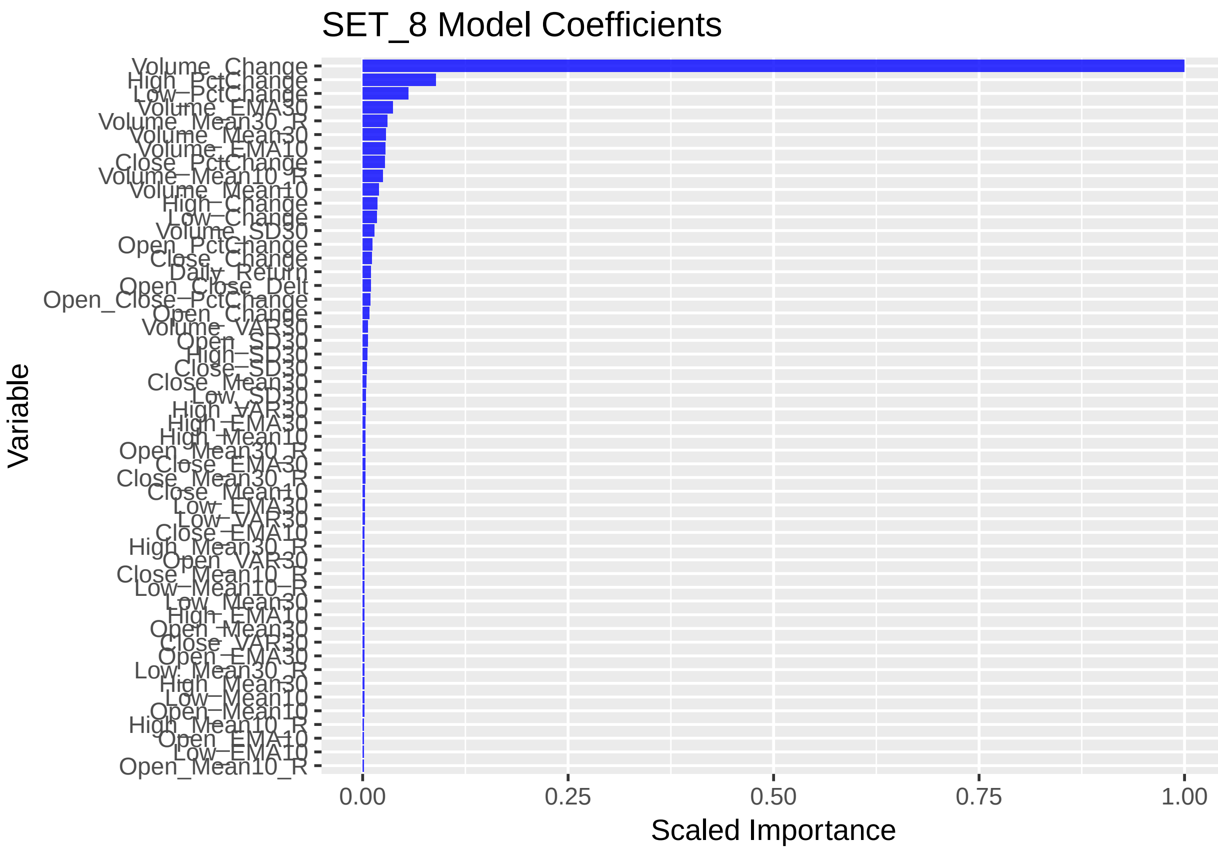


```
## [1] "SET_5 Model RMSE: 30.6235828652931"
```




```
## [1] "SET_6 Model RMSE: 24.0871172052513"
```


```
## [1] "SET_7 Model RMSE: 72.9803820641586"
```




```
## [1] "SET_8 Model RMSE: 22.4690534197918"
```


```
## [1] "SET_9 Model RMSE: 30.5628613749089"
```

```
#View RMSE for each test set
#[1] "SET_1 Model RMSE: 576.692237061663"
#[1] "SET_2 Model RMSE: 110.902571585891"
#[1] "SET_3 Model RMSE: 25.2511196082064"
#[1] "SET_4 Model RMSE: 32.3291983290633"
#[1] "SET_5 Model RMSE: 30.5608903784423"
#[1] "SET_6 Model RMSE: 24.0871172052513"
#[1] "SET_7 Model RMSE: 72.9803820641586"
#[1] "SET_8 Model RMSE: 22.4690534197918"
#[1] "SET_9 Model RMSE: 30.3120425213452"
#model appears unstable with large RMSE range

#Most important variables based on frequency across test sets
imp.var <- c(
"Volume_EMA10",
"Volume_Change",
"Volume_Mean30_R",
"High_PctChange",
"Low_PctChange",
"Volume_Mean30",
"Close_PctChange",
"Close_SD30",
"High_Change",
"Close_Change",
"Close_EMA30",
```

```r
"High_Mean30",
"High_SD30",
"Low_EMA10",
"Volume_SD30",
"Volume_VAR30"
)

#change testSet sets to dataframes
test.set1 <- as.data.frame(testSet[["SET_1"]])
test.set2 <- as.data.frame(testSet[["SET_2"]])
test.set3 <- as.data.frame(testSet[["SET_3"]])
test.set4 <- as.data.frame(testSet[["SET_4"]])
test.set5 <- as.data.frame(testSet[["SET_5"]])
test.set6 <- as.data.frame(testSet[["SET_6"]])
test.set7 <- as.data.frame(testSet[["SET_7"]])
test.set8 <- as.data.frame(testSet[["SET_8"]])
test.set9 <- as.data.frame(testSet[["SET_9"]])

#change testList sets to dataframes
pred.set1 = as.data.frame(testList[["SET_1"]])
pred.set2 = as.data.frame(testList[["SET_2"]])
pred.set3 = as.data.frame(testList[["SET_3"]])
pred.set4 = as.data.frame(testList[["SET_4"]])
pred.set5 = as.data.frame(testList[["SET_5"]])
pred.set6 = as.data.frame(testList[["SET_6"]])
pred.set7 = as.data.frame(testList[["SET_7"]])
pred.set8 = as.data.frame(testList[["SET_8"]])
pred.set9 = as.data.frame(testList[["SET_9"]])

#attached predicted values to test dataframe
temp.1 <- cbind(test.set1, pred.set1)
temp.2 <- cbind(test.set2, pred.set2)
temp.3 <- cbind(test.set3, pred.set3)
temp.4 <- cbind(test.set4, pred.set4)
temp.5 <- cbind(test.set5, pred.set5)
temp.6 <- cbind(test.set6, pred.set6)
temp.7 <- cbind(test.set7, pred.set7)
temp.8 <- cbind(test.set8, pred.set8)
temp.9 <- cbind(test.set9, pred.set9)

pred.rf.4 <-
rbind(temp.1,
temp.2,
temp.3,
temp.4,
temp.5,
temp.6,
temp.7,
temp.8,
temp.9)

#ape computes the elementwise absolute percent difference
#between two numeric vectors
```

```
pred.rf.4$APE <-
ape(pred.rf.4$Open_Close_PctChange, pred.rf.4$predict)

# use the function to identify outliers
outliers <- FindOutliers(pred.rf.4$APE)

# remove non outliers
RF.Outliers.4 <- pred.rf.4[outliers,]

#remove rows with APE of Inf
RF.Outliers.4 <- RF.Outliers.4[is.finite(RF.Outliers.4$APE), ]
```

6.2.5 Response = Volume % Change, Predictors = 16 Most Imp Variables

```
#Response = Volume_PctChange
#Predictors = 16 most important variables from above model

# response and predictors to use
resp <- "Volume_PctChange"
pred <- imp.var

# iterating over the keys
modelList <- list()
testList <- list()

for (aKey in names(trainSet)) {
aTrain <- trainSet[[aKey]]
aTest <- testSet[[aKey]]

# make h2o data.frame, loads into H2O service
train.hex <- as.h2o(aTrain)
test.hex <- as.h2o(aTest)

# Summary
summary(train.hex, exact_quantiles = TRUE)
summary(test.hex, exact_quantiles = TRUE)

# train a model with that data
loop.rf = h2o.randomForest(
x = pred,
y = resp,
training_frame = train.hex,
validation_frame = test.hex,
model_id = "rf_v1",
## name the model in H2O, helps use Flow
ntrees = 200,
## use a maximum of 200 trees to create the
##  random forest model. Will let
##  the early stopping criteria decide when
##  the random forest is sufficiently accurate
max_depth = 30,
stopping_rounds = 2,
```

```
## Stop fitting new trees when the 2-tree
##  average is within 0.001 (default) of
##  the prior two 2-tree averages.
##  Can be thought of as a convergence setting
seed = 123 ## Set the random seed so that this can be reproduced
)

# save the model
modelList[[aKey]] <- loop.rf

# predict response variable
rf.pred = h2o.predict(object = loop.rf, newdata = test.hex)

# save the prediction
testList[[aKey]] <- rf.pred
}
```

```
#plot important variables for training set, list test set model RMSE
for (aKey in names(modelList)) {
mm <- modelList[[aKey]]

pp <- plotRFVariableImportance(mm@model) +
labs(title = sprintf('%s Model Coefficients', aKey))
print(pp)

print(sprintf("%s Model RMSE: %s", aKey, h2o.rmse(mm, valid = T)))

}
```

```
## [1] "SET_1 Model RMSE: 378.654264996058"

## [1] "SET_2 Model RMSE: 54.6655580560508"

## [1] "SET_3 Model RMSE: 43.1552758931998"

## [1] "SET_4 Model RMSE: 30.7571414430038"

## [1] "SET_5 Model RMSE: 30.6324531946655"

## [1] "SET_6 Model RMSE: 23.2334850408571"

## [1] "SET_7 Model RMSE: 72.7031683246301"

## [1] "SET_8 Model RMSE: 22.7140197448556"

## [1] "SET_9 Model RMSE: 43.2254666491596"
```

```
#View RMSE for each test set
#[1] "SET_1 Model RMSE: 393.538581877468"
#[1] "SET_2 Model RMSE: 61.4290006215392"
#[1] "SET_3 Model RMSE: 46.3983617020438"
#[1] "SET_4 Model RMSE: 30.7571414430038"
#[1] "SET_5 Model RMSE: 30.6324531946655"
#[1] "SET_6 Model RMSE: 23.2523143271395"
#[1] "SET_7 Model RMSE: 71.3298511208012"
#[1] "SET_8 Model RMSE: 22.6255653167132"
#[1] "SET_9 Model RMSE: 43.2254666491596"
#model appears unstable with large RMSE range
```

```r
#change testSet sets to dataframes
test.set1 <- as.data.frame(testSet[["SET_1"]])
test.set2 <- as.data.frame(testSet[["SET_2"]])
test.set3 <- as.data.frame(testSet[["SET_3"]])
test.set4 <- as.data.frame(testSet[["SET_4"]])
test.set5 <- as.data.frame(testSet[["SET_5"]])
test.set6 <- as.data.frame(testSet[["SET_6"]])
test.set7 <- as.data.frame(testSet[["SET_7"]])
test.set8 <- as.data.frame(testSet[["SET_8"]])
test.set9 <- as.data.frame(testSet[["SET_9"]])

#change testList sets to dataframes
pred.set1 = as.data.frame(testList[["SET_1"]])
pred.set2 = as.data.frame(testList[["SET_2"]])
pred.set3 = as.data.frame(testList[["SET_3"]])
pred.set4 = as.data.frame(testList[["SET_4"]])
pred.set5 = as.data.frame(testList[["SET_5"]])
pred.set6 = as.data.frame(testList[["SET_6"]])
pred.set7 = as.data.frame(testList[["SET_7"]])
pred.set8 = as.data.frame(testList[["SET_8"]])
pred.set9 = as.data.frame(testList[["SET_9"]])

#attached predicted values to test dataframe
temp.1 <- cbind(test.set1, pred.set1)
temp.2 <- cbind(test.set2, pred.set2)
temp.3 <- cbind(test.set3, pred.set3)
temp.4 <- cbind(test.set4, pred.set4)
temp.5 <- cbind(test.set5, pred.set5)
temp.6 <- cbind(test.set6, pred.set6)
temp.7 <- cbind(test.set7, pred.set7)
temp.8 <- cbind(test.set8, pred.set8)
temp.9 <- cbind(test.set9, pred.set9)

pred.rf.5 <-
rbind(temp.1,
temp.2,
temp.3,
temp.4,
temp.5,
temp.6,
temp.7,
temp.8,
temp.9)

#ape computes the elementwise absolute percent difference
#between two numeric vectors
pred.rf.5$APE <-
ape(pred.rf.5$Open_Close_PctChange, pred.rf.5$predict)

# use the function to identify outliers
outliers <- FindOutliers(pred.rf.5$APE)

# remove non outliers
```

```r
RF.Outliers.5 <- pred.rf.5[outliers,]

#remove rows with APE of Inf
RF.Outliers.5 <- RF.Outliers.5[is.finite(RF.Outliers.5$APE), ]
```

6.2.6 Response = Volume % Change, Predictors = 4 Most Imp Variables

```r
#Response = Volume_PctChange
#Predictors = 4 most important variables from above model

# response and predictors to use
resp <- "Volume_PctChange"
pred <- c("Volume_EMA10",
"Volume_Change",
"Volume_Mean30_R",
"High_PctChange")

# iterating over the keys
modelList <- list()
testList <- list()

for (aKey in names(trainSet)) {
aTrain <- trainSet[[aKey]]
aTest <- testSet[[aKey]]

# make h2o data.frame, loads into H2O service
train.hex <- as.h2o(aTrain)
test.hex <- as.h2o(aTest)

# Summary
summary(train.hex, exact_quantiles = TRUE)
summary(test.hex, exact_quantiles = TRUE)

# train a model with that data
loop.rf = h2o.randomForest(
x = pred,
y = resp,
training_frame = train.hex,
validation_frame = test.hex,
model_id = "rf_v1",
## name the model in H2O, helps use Flow
ntrees = 200,
## use a maximum of 200 trees to create the
##  random forest model. Will let
##  the early stopping criteria decide when
##  the random forest is sufficiently accurate
max_depth = 30,
stopping_rounds = 2,
## Stop fitting new trees when the 2-tree
##  average is within 0.001 (default) of
##  the prior two 2-tree averages.
##  Can be thought of as a convergence setting
```

```
seed = 123 ## Set the random seed so that this can be reproduced
)

# save the model
modelList[[aKey]] <- loop.rf

# predict response variable
rf.pred = h2o.predict(object = loop.rf, newdata = test.hex)

# save the prediction
testList[[aKey]] <- rf.pred
}
```

```
#plot important variables for training set, list test set model RMSE
for (aKey in names(modelList)) {
mm <- modelList[[aKey]]

pp <- plotRFVariableImportance(mm@model) +
labs(title = sprintf('%s Model Coefficients', aKey))
print(pp)

print(sprintf("%s Model RMSE: %s", aKey, h2o.rmse(mm, valid = T)))

}
```

```
## [1] "SET_1 Model RMSE: 57.6008523065804"

## [1] "SET_2 Model RMSE: 283.219566547425"

## [1] "SET_3 Model RMSE: 98.4701233077814"

## [1] "SET_4 Model RMSE: 28.0209827163222"

## [1] "SET_5 Model RMSE: 33.5335099105658"

## [1] "SET_6 Model RMSE: 27.3228808790625"

## [1] "SET_7 Model RMSE: 74.470905616025"

## [1] "SET_8 Model RMSE: 30.1222604896944"

## [1] "SET_9 Model RMSE: 38.811638702533"
```

```
#View RMSE for each test set
#[1] "SET_1 Model RMSE: 57.6008523065804"
#[1] "SET_2 Model RMSE: 283.219566547425"
#[1] "SET_3 Model RMSE: 98.4701233077814"
#[1] "SET_4 Model RMSE: 28.0209827163222"
#[1] "SET_5 Model RMSE: 33.5335099105658"
#[1] "SET_6 Model RMSE: 27.3228808790625"
#[1] "SET_7 Model RMSE: 74.7739503333878"
#[1] "SET_8 Model RMSE: 30.1222604896944"
#[1] "SET_9 Model RMSE: 38.811638702533"
#model appears unstable with large RMSE range

#change testSet sets to dataframes
test.set1 <- as.data.frame(testSet[["SET_1"]])
test.set2 <- as.data.frame(testSet[["SET_2"]])
test.set3 <- as.data.frame(testSet[["SET_3"]])
```

```r
test.set4 <- as.data.frame(testSet[["SET_4"]])
test.set5 <- as.data.frame(testSet[["SET_5"]])
test.set6 <- as.data.frame(testSet[["SET_6"]])
test.set7 <- as.data.frame(testSet[["SET_7"]])
test.set8 <- as.data.frame(testSet[["SET_8"]])
test.set9 <- as.data.frame(testSet[["SET_9"]])

#change testList sets to dataframes
pred.set1 = as.data.frame(testList[["SET_1"]])
pred.set2 = as.data.frame(testList[["SET_2"]])
pred.set3 = as.data.frame(testList[["SET_3"]])
pred.set4 = as.data.frame(testList[["SET_4"]])
pred.set5 = as.data.frame(testList[["SET_5"]])
pred.set6 = as.data.frame(testList[["SET_6"]])
pred.set7 = as.data.frame(testList[["SET_7"]])
pred.set8 = as.data.frame(testList[["SET_8"]])
pred.set9 = as.data.frame(testList[["SET_9"]])

#attached predicted values to test dataframe
temp.1 <- cbind(test.set1, pred.set1)
temp.2 <- cbind(test.set2, pred.set2)
temp.3 <- cbind(test.set3, pred.set3)
temp.4 <- cbind(test.set4, pred.set4)
temp.5 <- cbind(test.set5, pred.set5)
temp.6 <- cbind(test.set6, pred.set6)
temp.7 <- cbind(test.set7, pred.set7)
temp.8 <- cbind(test.set8, pred.set8)
temp.9 <- cbind(test.set9, pred.set9)

pred.rf.6 <-
rbind(temp.1,
temp.2,
temp.3,
temp.4,
temp.5,
temp.6,
temp.7,
temp.8,
temp.9)

#ape computes the elementwise absolute percent difference
#between two numeric vectors
pred.rf.6$APE <-
ape(pred.rf.6$Open_Close_PctChange, pred.rf.6$predict)

# use the function to identify outliers
outliers <- FindOutliers(pred.rf.6$APE)

# remove non outliers
RF.Outliers.6 <- pred.rf.6[outliers,]

#remove rows with APE of Inf
RF.Outliers.6 <- RF.Outliers.6[is.finite(RF.Outliers.6$APE), ]
```

```r
#tune hyperparameters to see if model becomes more stable
# iterating over the keys
modelList <- list()
testList <- list()

for (aKey in names(trainSet)) {
aTrain <- trainSet[[aKey]]
aTest <- testSet[[aKey]]

# make h2o data.frame, loads into H2O service
train.hex <- as.h2o(aTrain)
test.hex <- as.h2o(aTest)

# Summary
summary(train.hex, exact_quantiles = TRUE)
summary(test.hex, exact_quantiles = TRUE)

# train a model with that data
loop.rf = h2o.randomForest(
x = pred,
y = resp,
training_frame = train.hex,
validation_frame = test.hex,
model_id = "rf_v7",
ntrees = 50,
## change to 50
max_depth = 100,
## change to 50, remove stopping_rounds
seed = 123 ## Set the random seed so that this can be reproduced
)

# save the model
modelList[[aKey]] <- loop.rf

# predict response variable
rf.pred = h2o.predict(object = loop.rf, newdata = test.hex)

# save the prediction
testList[[aKey]] <- rf.pred
}
```

```r
#plot important variables for training set, list test set model RMSE
for (aKey in names(modelList)) {
mm <- modelList[[aKey]]

pp <- plotRFVariableImportance(mm@model) +
labs(title = sprintf('%s Model Coefficients', aKey))
print(pp)

print(sprintf("%s Model RMSE: %s", aKey, h2o.rmse(mm, valid = T)))

}
```

```r
#View RMSE for each test set
#[1] "SET_1 Model RMSE: 120.226605930769"
#[1] "SET_2 Model RMSE: 87.7838608149893"
#[1] "SET_3 Model RMSE: 40.5480887100986"
#[1] "SET_4 Model RMSE: 24.5603643489344"
#[1] "SET_5 Model RMSE: 32.666127615185"
#[1] "SET_6 Model RMSE: 27.10781137695"
#[1] "SET_7 Model RMSE: 73.2567814337984"
#[1] "SET_8 Model RMSE: 25.6022741909722"
#[1] "SET_9 Model RMSE: 36.5643091548129"
#model appears more stable with less range in RMSE

#change testSet sets to dataframes
test.set1 <- as.data.frame(testSet[["SET_1"]])
test.set2 <- as.data.frame(testSet[["SET_2"]])
test.set3 <- as.data.frame(testSet[["SET_3"]])
test.set4 <- as.data.frame(testSet[["SET_4"]])
test.set5 <- as.data.frame(testSet[["SET_5"]])
test.set6 <- as.data.frame(testSet[["SET_6"]])
test.set7 <- as.data.frame(testSet[["SET_7"]])
test.set8 <- as.data.frame(testSet[["SET_8"]])
test.set9 <- as.data.frame(testSet[["SET_9"]])

#change testList sets to dataframes
pred.set1 = as.data.frame(testList[["SET_1"]])
pred.set2 = as.data.frame(testList[["SET_2"]])
pred.set3 = as.data.frame(testList[["SET_3"]])
pred.set4 = as.data.frame(testList[["SET_4"]])
pred.set5 = as.data.frame(testList[["SET_5"]])
pred.set6 = as.data.frame(testList[["SET_6"]])
pred.set7 = as.data.frame(testList[["SET_7"]])
pred.set8 = as.data.frame(testList[["SET_8"]])
pred.set9 = as.data.frame(testList[["SET_9"]])

#attached predicted values to test dataframe
temp.1 <- cbind(test.set1, pred.set1)
temp.2 <- cbind(test.set2, pred.set2)
temp.3 <- cbind(test.set3, pred.set3)
temp.4 <- cbind(test.set4, pred.set4)
temp.5 <- cbind(test.set5, pred.set5)
temp.6 <- cbind(test.set6, pred.set6)
temp.7 <- cbind(test.set7, pred.set7)
temp.8 <- cbind(test.set8, pred.set8)
temp.9 <- cbind(test.set9, pred.set9)

pred.rf.7 <-
rbind(temp.1,
temp.2,
temp.3,
temp.4,
temp.5,
temp.6,
temp.7,
```

```
temp.8,
temp.9)

#ape computes the elementwise absolute percent difference
#between two numeric vectors
pred.rf.7$APE <-
ape(pred.rf.7$Open_Close_PctChange, pred.rf.7$predict)

# use the function to identify outliers
outliers <- FindOutliers(pred.rf.7$APE)

# remove non outliers
RF.Outliers.7 <- pred.rf.7[outliers,]

#remove rows with APE of Inf
RF.Outliers.7 <- RF.Outliers.7[is.finite(RF.Outliers.7$APE), ]
```

6.2.7 Outliers

```
#Best model for Open Close % Change used 4 variables
kable(head(RF.Outliers.3[, 1:7])) %>% kable_styling(latex_options = "scale_down")
```

	Symbol	Date	Open	High	Low	Close	Volume
39441	FFIV	2011-01-20	111.190	114.750	106.100	109.150	23321247
61439	BLK	2011-05-19	165.460	166.920	163.720	165.450	1097505
90891	BXP	2011-09-29	75.916	76.260	74.680	75.909	2032185
120657	CAT	2011-10-24	77.060	78.019	76.254	77.068	19475803
121839	CME	2011-10-03	37.843	38.544	37.216	37.845	4748905
151063	CMG	2012-01-12	347.610	348.510	343.800	347.620	384830

```
#Best model for Volume % Change used 4 variables
kable(head(RF.Outliers.7[, 1:7])) %>% kable_styling(latex_options = "scale_down")
```

	Symbol	Date	Open	High	Low	Close	Volume
5443	CHTR	2010-10-06	35.844	36.386	35.844	35.855	12500
5450	CHTR	2010-10-15	36.828	37.038	36.176	36.817	1114804
12646	HCP	2010-11-03	25.353	25.439	24.819	25.345	17815635
34681	CLX	2011-01-03	50.430	50.602	49.595	50.421	8434040
34893	CME	2011-02-09	45.368	46.444	44.954	45.370	5585956
41467	HCP	2011-03-23	25.952	26.051	25.749	25.947	33925723

```
#remove predict and APE columns
RF.Price <- RF.Outliers.3
RF.Price$predict <- NULL
RF.Price$APE <- NULL
RF.Volume <- RF.Outliers.7
```

```r
RF.Volume$predict <- NULL
RF.Volume$APE <- NULL

#see if the outliers found with the best models for both
#Open Close % Change and Volume % Change overlap
common <- inner_join(RF.Price, RF.Volume)  #2 rows in common
kable(common[,1:2])
```

Symbol	Date
BLK	2011-05-19
AMZN	2012-07-26

```r
#see if outliers found with GLM and RF overlap
#Volume % Change overlap
GLM.Volume.new <- GLM.Volume[,1:2]
RF.Volume.new <- RF.Volume[,1:2]
common.volume <- inner_join(GLM.Volume.new, RF.Volume.new)  #0 rows in common

#Open Close % Change overlap
GLM.Price.new <- GLM.Price[,1:2]
RF.Price.new <- RF.Price[,1:2]
common.price <- inner_join(GLM.Price.new, RF.Price.new)  #4 rows in common

#both RF and GLM models when modeling for Open Close % Change
#found 4 overlapping datapoints.
kable(common.price)
```

Symbol	Date
BLK	2011-05-19
CMG	2012-01-12
GOOGL	2012-03-27
GOOGL	2012-12-10

RF:

In order to detect possible outliers, I trained several RF models for normal using Open Close % Change and Volume % Change as my response variables.

Open Close % Change: Modeling for Open Close % Change resulted in 13 possible outliers as the actual Open Close % Change differed significantly from the value predicted by the RF model. These data points required additional analysis to determine if the anomaly detection was correct.

Volume % Change: Modeling for Volume % Change resulted in 16 possible outliers as the actual Volume % Change differed significantly from the value predicted by the RF model. These data points required additional analysis to determine if the anomaly detection was correct.

Overall, my RF models predicted greater change in Open Close % Change and Volume % Change than actually occurred. However, upon further inspection I was unable to definitively say that my model was more correct than the actual results.

Both RF and GLM models:

I then looked to see if my RF and GLM models came up with the same possible outliers. Both RF and GLM models when modeling for Open Close % Change found 4 overlapping data points:

BLK: 5/19/11 BLK's closing price was flat versus its opening price. However, my model predicted a 5% decline. It's difficult to say that my model is correct and that the stock should have traded down significantly instead of flat. Particularly given that the stock traded in a narrow band the week prior to 5/19/11 and the week after (-1% - +1%).

CMG: 1/12/12 CMG's closing price was flat versus its opening price. However, my model predicted a 12% decline. Again, it's difficult to say that my model is correct and that the stock should have traded down significantly instead of flat. Particularly given that the stock traded in a narrow band the week prior to 1/12/12 and the week after (-2% - +3%).

GOOGL: 3/27/12 GOOGL's closing price was flat versus its opening price. However, my model predicted a 3.4% increase. It's difficult to say that my model is correct and that the stock should have traded up. Particularly given that the stock traded in a narrow band the week prior to 3/27/12 and the week after (-2% - +1%). 12/10/12 GOOGL's closing price was flat versus its opening price. However, my model predicted a 2% increase. It's difficult to say that my model is correct and that the stock should have traded up instead of flat. Particularly given that the stock traded in a narrow band the week prior to 12/10/12 and the week after (-2% - +2%).