R Notebook

Summary

In this project, we combined core value of 8 published articles related to Earnings Release and improve it with machine learning strategy. Finally, we got two models: long-before earning release and short-after earning release. Our model could tell you, if you long the stock before earning announcement, whether you could earn a positive return. If you short the stock after earning announcement, whether you could get a positive return.

Step 0: Related Papers and Projects

We read more than 8 papers including Post-Earnings Announcement Effect, Momentum and Trend Following in Global Asset Allocation, and etc. Details and links of papers can be found at data/paper/ListofStrategyPapers.xlsx. We then implemented methods and algorithms described in those papers.

Step 1: Data Collection And Processing

Part a: Using Matlab to process data

- **Data source: Using Bloomberg terminal to get stock related data(including EBIT_growth,Price to book value and 7 other indicators) and 06-17 earning announcement data. Original data can be found in data/original.**

- **Size: 9 files with a total size of 150MB**

- **Data Processing: For each announcement, we need to find the correspoding entry with same sticker and date in 9 indicators files respectively. We also calculate 1-year-Momentum = (Closing Price of Date a / Closing Price of Data(a-252) * 100). All the processing process took place in Matlab. Codes can be found at lib/data_process_matlab**

- **Result: We got 10 processed CSV files and you can find them at data/observation**

Part b: Using R to run regression and find the most important indicators

- **Data source: data/observation_processed/**

- **Size: 4 files**

- **Data Processing: Using StepWise AIC in R**

- **Result: Momentum and PS are two signiciant indicators**

Load library

```
library(DAAG)
```

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

```
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:DAAG':
##
##     hills
```

```
library(leaps)
library(car)
```

```
##
## Attaching package: 'car'

## The following object is masked from 'package:DAAG':
##
##     vif
```

Process Data

```
setwd("~/Spr2017-proj5-grp15/data/original/")
longBefore <- read.csv("../observation_processed/long_before_earning.csv")
longAfter <- read.csv("../observation_processed/long_after_earning.csv")
shortBefore <- read.csv("../observation_processed/short_before_earning.csv")
shortAfter <- read.csv("../observation_processed/short_after_earning.csv")
```

Using stepwise AIC to find the most inportant indicators

```
longBefore <- longBefore[,-1]
longAfter <- longAfter[,-1]
shortAfter <- shortAfter[,-1]
shortBefore <- shortBefore[,-1]
fitLongAfter <- lm(RETURN~DY+EBITG+EV2EBITDA+M2B+MOMENTUM+PB+PE+PF+PS+days+surprise,data=longAfter)
stepLA <- stepAIC(fitLongAfter, direction="both")
```

```
## Start:  AIC=-127781.8
## RETURN ~ DY + EBITG + EV2EBITDA + M2B + MOMENTUM + PB + PE +
##     PF + PS + days + surprise
##
##             Df Sum of Sq    RSS     AIC
## - EBITG      1   0.00138 88.481 -127783
## - PE         1   0.00202 88.481 -127783
## <none>                   88.479 -127782
## - M2B        1   0.00943 88.489 -127781
## - PB         1   0.01151 88.491 -127781
## - PF         1   0.01251 88.492 -127781
## - EV2EBITDA  1   0.01905 88.498 -127779
## - days       1   0.02045 88.500 -127779
## - DY         1   0.04665 88.526 -127772
## - surprise   1   0.10680 88.586 -127756
## - PS          1   0.28966 88.769 -127709
## - MOMENTUM    1   1.11561 89.595 -127496
##
## Step:  AIC=-127783.5
## RETURN ~ DY + EV2EBITDA + M2B + MOMENTUM + PB + PE + PF + PS +
##     days + surprise
##
##             Df Sum of Sq    RSS     AIC
## - PE         1   0.00194 88.482 -127785
## <none>                   88.481 -127783
## - M2B        1   0.00945 88.490 -127783
## - PB         1   0.01152 88.492 -127782
## - PF         1   0.01257 88.493 -127782
## + EBITG      1   0.00138 88.479 -127782
## - EV2EBITDA  1   0.01892 88.499 -127781
## - days       1   0.02045 88.501 -127780
## - DY         1   0.04656 88.527 -127773
## - surprise   1   0.10669 88.587 -127758
## - PS         1   0.28955 88.770 -127710
## - MOMENTUM   1   1.11514 89.596 -127498
##
## Step:  AIC=-127784.9
## RETURN ~ DY + EV2EBITDA + M2B + MOMENTUM + PB + PF + PS + days +
##     surprise
##
##             Df Sum of Sq    RSS     AIC
## <none>                   88.482 -127785
## - M2B        1   0.00924 88.492 -127785
## - PB         1   0.01129 88.494 -127784
## - PF         1   0.01274 88.495 -127784
## + PE         1   0.00194 88.481 -127783
## + EBITG      1   0.00130 88.481 -127783
## - EV2EBITDA  1   0.01718 88.500 -127782
## - days       1   0.02045 88.503 -127782
## - DY         1   0.04646 88.529 -127775
## - surprise   1   0.10639 88.589 -127759
## - PS         1   0.28777 88.770 -127712
## - MOMENTUM   1   1.11793 89.600 -127498
```

```r
summary(stepLA)
```

```
##
## Call:
## lm(formula = RETURN ~ DY + EV2EBITDA + M2B + MOMENTUM + PB +
##     PF + PS + days + surprise, data = longAfter)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.49333 -0.03223 -0.00191  0.03097  0.72365
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.534e-03  1.069e-03  -3.307 0.000945 ***
## DY          -2.222e-04  6.397e-05  -3.473 0.000515 ***
## EV2EBITDA    5.801e-05  2.747e-05   2.112 0.034704 *
## M2B         -3.295e-04  2.128e-04  -1.549 0.121491
## MOMENTUM    -1.851e-02  1.087e-03 -17.038  < 2e-16 ***
## PB           4.441e-04  2.594e-04   1.712 0.086880 .
## PF          -4.691e-06  2.579e-06  -1.819 0.068895 .
## PS           1.882e-03  2.177e-04   8.645  < 2e-16 ***
## days         5.523e-04  2.397e-04   2.304 0.021210 *
## surprise     1.696e-05  3.226e-06   5.256 1.48e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06206 on 22977 degrees of freedom
## Multiple R-squared:  0.02536,    Adjusted R-squared:  0.02498
## F-statistic: 66.43 on 9 and 22977 DF,  p-value: < 2.2e-16
```

```
fitLongBefore <- lm(RETURN~DY+EBITG+EV2EBITDA+M2B+MOMENTUM+PB+PE+PF+PS+days,data=longBefore)
stepLB <- stepAIC(fitLongBefore, direction="both")
```

```
## Start:  AIC=-152475.9
## RETURN ~ DY + EBITG + EV2EBITDA + M2B + MOMENTUM + PB + PE +
##     PF + PS + days
##
##             Df Sum of Sq    RSS     AIC
## - PE         1  0.000006 31.064 -152478
## - EV2EBITDA  1  0.000035 31.064 -152478
## - PF         1  0.000587 31.065 -152477
## - DY         1  0.002031 31.066 -152476
## <none>                   31.064 -152476
## - PB         1  0.003344 31.068 -152475
## - M2B        1  0.003686 31.068 -152475
## - EBITG      1  0.007261 31.072 -152473
## - PS         1  0.014931 31.079 -152467
## - days       1  0.037792 31.102 -152450
## - MOMENTUM   1  0.060704 31.125 -152433
##
## Step:  AIC=-152477.9
## RETURN ~ DY + EBITG + EV2EBITDA + M2B + MOMENTUM + PB + PF +
##     PS + days
##
##             Df Sum of Sq    RSS     AIC
## - EV2EBITDA  1  0.000044 31.064 -152480
## - PF         1  0.000585 31.065 -152479
## - DY         1  0.002033 31.066 -152478
## <none>                   31.064 -152478
## - PB         1  0.003339 31.068 -152477
## - M2B        1  0.003680 31.068 -152477
## + PE         1  0.000006 31.064 -152476
## - EBITG      1  0.007276 31.072 -152475
## - PS         1  0.014944 31.079 -152469
## - days       1  0.037792 31.102 -152452
## - MOMENTUM   1  0.060761 31.125 -152435
##
## Step:  AIC=-152479.9
## RETURN ~ DY + EBITG + M2B + MOMENTUM + PB + PF + PS + days
##
##             Df Sum of Sq    RSS     AIC
## - PF         1  0.000572 31.065 -152481
## - DY         1  0.002007 31.066 -152480
## <none>                   31.064 -152480
## - PB         1  0.003319 31.068 -152479
## - M2B        1  0.003660 31.068 -152479
## + EV2EBITDA  1  0.000044 31.064 -152478
## + PE         1  0.000014 31.064 -152478
## - EBITG      1  0.007267 31.072 -152476
## - PS         1  0.015887 31.080 -152470
## - days       1  0.037791 31.102 -152454
## - MOMENTUM   1  0.060745 31.125 -152437
##
## Step:  AIC=-152481.5
## RETURN ~ DY + EBITG + M2B + MOMENTUM + PB + PS + days
##
##             Df Sum of Sq    RSS     AIC
## - DY         1  0.001962 31.067 -152482
## <none>                   31.065 -152481
## - PB         1  0.003349 31.068 -152481
## - M2B        1  0.003692 31.069 -152481
## + PF         1  0.000572 31.064 -152480
## + EV2EBITDA  1  0.000031 31.065 -152479
## + PE         1  0.000010 31.065 -152479
## - EBITG      1  0.007298 31.072 -152478
## - PS         1  0.016196 31.081 -152471
## - days       1  0.037790 31.103 -152455
## - MOMENTUM   1  0.060682 31.126 -152438
##
## Step:  AIC=-152482
## RETURN ~ EBITG + M2B + MOMENTUM + PB + PS + days
##
##             Df Sum of Sq    RSS     AIC
## <none>                   31.067 -152482
## - PB         1  0.002845 31.070 -152482
## - M2B        1  0.003165 31.070 -152482
## + DY         1  0.001962 31.065 -152481
## + PF         1  0.000528 31.066 -152480
## + EV2EBITDA  1  0.000011 31.067 -152480
## + PE         1  0.000008 31.067 -152480
## - EBITG      1  0.007251 31.074 -152479
## - PS         1  0.015349 31.082 -152473
## - days       1  0.037788 31.105 -152456
## - MOMENTUM   1  0.060008 31.127 -152439
```

```
summary(stepLB)
```

```
##
## Call:
## lm(formula = RETURN ~ EBITG + M2B + MOMENTUM + PB + PS + days,
##     data = longBefore)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.40715 -0.01595  0.00052  0.01645  0.38948
##
```

```
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.343e-03  6.061e-04   2.216 0.026731 *
## EBITG        4.717e-07  2.033e-07   2.320 0.020347 *
## M2B          1.987e-04  1.296e-04   1.533 0.125307
## MOMENTUM     4.168e-03  6.245e-04   6.674 2.54e-11 ***
## PB          -2.315e-04  1.593e-04  -1.453 0.146124
## PS          -4.197e-04  1.243e-04  -3.376 0.000738 ***
## days         7.494e-04  1.415e-04   5.296 1.19e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0367 on 23063 degrees of freedom
## Multiple R-squared:  0.004778,   Adjusted R-squared:  0.004519
## F-statistic: 18.45 on 6 and 23063 DF,  p-value: < 2.2e-16
```

```r
fitShortBefore <- lm(RETURN~DY+EBITG+EV2EBITDA+M2B+MOMENTUM+PB+PE+PF+PS+days,data=shortBefore)
stepSB <- stepAIC(fitShortBefore, direction="both")
```

```
## Start:  AIC=-152475.9
## RETURN ~ DY + EBITG + EV2EBITDA + M2B + MOMENTUM + PB + PE +
##     PF + PS + days
##
##             Df Sum of Sq    RSS     AIC
## - PE         1  0.000006 31.064 -152478
## - EV2EBITDA  1  0.000035 31.064 -152478
## - PF         1  0.000587 31.065 -152477
## - DY         1  0.002031 31.066 -152476
## <none>                   31.064 -152476
## - PB         1  0.003344 31.068 -152475
## - M2B        1  0.003686 31.068 -152475
## - EBITG      1  0.007261 31.072 -152473
## - PS         1  0.014931 31.079 -152467
## - days       1  0.037792 31.102 -152450
## - MOMENTUM   1  0.060704 31.125 -152433
##
## Step:  AIC=-152477.9
## RETURN ~ DY + EBITG + EV2EBITDA + M2B + MOMENTUM + PB + PF +
##     PS + days
##
##             Df Sum of Sq    RSS     AIC
## - EV2EBITDA  1  0.000044 31.064 -152480
## - PF         1  0.000585 31.065 -152479
## - DY         1  0.002033 31.066 -152478
## <none>                   31.064 -152478
## - PB         1  0.003339 31.068 -152477
## - M2B        1  0.003680 31.068 -152477
## + PE         1  0.000006 31.064 -152476
## - EBITG      1  0.007276 31.072 -152475
## - PS         1  0.014944 31.079 -152469
## - days       1  0.037792 31.102 -152452
## - MOMENTUM   1  0.060761 31.125 -152435
##
## Step:  AIC=-152479.9
## RETURN ~ DY + EBITG + M2B + MOMENTUM + PB + PF + PS + days
```

```
##
##              Df Sum of Sq    RSS     AIC
## - PF         1  0.000572 31.065 -152481
## - DY         1  0.002007 31.066 -152480
## <none>                   31.064 -152480
## - PB         1  0.003319 31.068 -152479
## - M2B        1  0.003660 31.068 -152479
## + EV2EBITDA  1  0.000044 31.064 -152478
## + PE         1  0.000014 31.064 -152478
## - EBITG      1  0.007267 31.072 -152476
## - PS         1  0.015887 31.080 -152470
## - days       1  0.037791 31.102 -152454
## - MOMENTUM   1  0.060745 31.125 -152437
##
## Step:  AIC=-152481.5
## RETURN ~ DY + EBITG + M2B + MOMENTUM + PB + PS + days
##
##              Df Sum of Sq    RSS     AIC
## - DY         1  0.001962 31.067 -152482
## <none>                   31.065 -152481
## - PB         1  0.003349 31.068 -152481
## - M2B        1  0.003692 31.069 -152481
## + PF         1  0.000572 31.064 -152480
## + EV2EBITDA  1  0.000031 31.065 -152479
## + PE         1  0.000010 31.065 -152479
## - EBITG      1  0.007298 31.072 -152478
## - PS         1  0.016196 31.081 -152471
## - days       1  0.037790 31.103 -152455
## - MOMENTUM   1  0.060682 31.126 -152438
##
## Step:  AIC=-152482
## RETURN ~ EBITG + M2B + MOMENTUM + PB + PS + days
##
##              Df Sum of Sq    RSS     AIC
## <none>                   31.067 -152482
## - PB         1  0.002845 31.070 -152482
## - M2B        1  0.003165 31.070 -152482
## + DY         1  0.001962 31.065 -152481
## + PF         1  0.000528 31.066 -152480
## + EV2EBITDA  1  0.000011 31.067 -152480
## + PE         1  0.000008 31.067 -152480
## - EBITG      1  0.007251 31.074 -152479
## - PS         1  0.015349 31.082 -152473
## - days       1  0.037788 31.105 -152456
## - MOMENTUM   1  0.060008 31.127 -152439
```

```
summary(stepSB)
```

```
##
## Call:
## lm(formula = RETURN ~ EBITG + M2B + MOMENTUM + PB + PS + days,
##     data = shortBefore)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.38948 -0.01645 -0.00052  0.01595  0.40715
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.343e-03  6.061e-04  -2.216 0.026731 *
## EBITG       -4.717e-07  2.033e-07  -2.320 0.020347 *
## M2B         -1.987e-04  1.296e-04  -1.533 0.125307
## MOMENTUM    -4.168e-03  6.245e-04  -6.674 2.54e-11 ***
## PB           2.315e-04  1.593e-04   1.453 0.146124
## PS           4.197e-04  1.243e-04   3.376 0.000738 ***
## days        -7.494e-04  1.415e-04  -5.296 1.19e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0367 on 23063 degrees of freedom
## Multiple R-squared:  0.004778,   Adjusted R-squared:  0.004519
## F-statistic: 18.45 on 6 and 23063 DF,  p-value: < 2.2e-16
```

```r
fitShortAfter <- lm(RETURN~DY+EBITG+EV2EBITDA+M2B+MOMENTUM+PB+PE+PF+PS+days,data=shortAfter)
stepSA <- stepAIC(fitShortAfter, direction="both")
```

```
## Start:  AIC=-127756.1
## RETURN ~ DY + EBITG + EV2EBITDA + M2B + MOMENTUM + PB + PE +
##     PF + PS + days
##
##             Df Sum of Sq    RSS     AIC
## - EBITG      1   0.00126 88.587 -127758
## - PE         1   0.00171 88.588 -127758
## <none>                   88.586 -127756
## - M2B        1   0.00995 88.596 -127756
## - PB         1   0.01207 88.598 -127755
## - PF         1   0.01260 88.599 -127755
## - EV2EBITDA  1   0.01954 88.605 -127753
## - days       1   0.02045 88.606 -127753
## - DY         1   0.04871 88.635 -127745
## - PS         1   0.28393 88.870 -127685
## - MOMENTUM   1   1.11912 89.705 -127470
##
## Step:  AIC=-127757.8
## RETURN ~ DY + EV2EBITDA + M2B + MOMENTUM + PB + PE + PF + PS +
##     days
##
##             Df Sum of Sq    RSS     AIC
## - PE         1   0.00164 88.589 -127759
## <none>                   88.587 -127758
## - M2B        1   0.00996 88.597 -127757
## - PB         1   0.01209 88.599 -127757
## - PF         1   0.01266 88.600 -127756
## + EBITG      1   0.00126 88.586 -127756
## - EV2EBITDA  1   0.01942 88.607 -127755
## - days       1   0.02045 88.608 -127754
## - DY         1   0.04862 88.636 -127747
## - PS         1   0.28383 88.871 -127686
## - MOMENTUM   1   1.11867 89.706 -127471
##
```

```
## Step:  AIC=-127759.3
## RETURN ~ DY + EV2EBITDA + M2B + MOMENTUM + PB + PF + PS + days
##
##              Df Sum of Sq    RSS     AIC
## <none>                    88.589 -127759
## - M2B        1   0.00976 88.599 -127759
## - PB         1   0.01187 88.601 -127758
## - PF         1   0.01282 88.602 -127758
## + PE         1   0.00164 88.587 -127758
## + EBITG      1   0.00119 88.588 -127758
## - EV2EBITDA  1   0.01787 88.607 -127757
## - days       1   0.02045 88.609 -127756
## - DY         1   0.04852 88.637 -127749
## - PS         1   0.28228 88.871 -127688
## - MOMENTUM   1   1.12129 89.710 -127472
```

```
summary(stepSA)
```

```
##
## Call:
## lm(formula = RETURN ~ DY + EV2EBITDA + M2B + MOMENTUM + PB +
##     PF + PS + days, data = shortAfter)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.71892 -0.03096  0.00200  0.03229  0.49368
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.316e-03  1.069e-03   3.103  0.00192 **
## DY           2.270e-04  6.400e-05   3.547  0.00039 ***
## EV2EBITDA   -5.917e-05  2.748e-05  -2.153  0.03132 *
## M2B          3.388e-04  2.129e-04   1.591  0.11156
## MOMENTUM    -1.854e-02  1.087e-03 -17.054  < 2e-16 ***
## PB          -4.553e-04  2.595e-04  -1.754  0.07936 .
## PF           4.705e-06  2.580e-06   1.824  0.06822 .
## PS          -1.863e-03  2.178e-04  -8.557  < 2e-16 ***
## days        -5.523e-04  2.398e-04  -2.303  0.02128 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06209 on 22978 degrees of freedom
## Multiple R-squared:  0.02419,    Adjusted R-squared:  0.02385
## F-statistic:  71.2 on 8 and 22978 DF,  p-value: < 2.2e-16
```

Step 2: Select the stocks based on their Momentum

- **Data source: data/observation**

- **Size: 17 files with a total size of 70MB**

- **Data Processing: Using R to select 1500 observations with highest Momentum and use Python to reorganize data and store them in a proper way. Python code can be found at**

lib/reorganized.py.

- **Result: We got 2 models: Long-Before-Model and Short-After-Model**

```r
### Selected 1500 observations with highest Momentum
setwd("~/Spr2017-proj5-grp15/data/original/")
mom <- read.csv("../observation/momentum39034.csv",header = FALSE)
n_rows=nrow(mom)

## Long_before_earning.csv and Short_after_earning.csv are written using Python
long_before <- read.csv('../observation_processed/long_before_earning.csv')
short_after <- read.csv('../observation_processed/short_after_earning.csv')
nrow_longb=nrow(long_before)
nrow_shorta=nrow(short_after)

mom_sa=c()
for (i in 1:nrow_shorta){
  inter=max(short_after[i,'MOMENTUM'])
  mom_sa=c(mom_sa,inter)
}
mom_sa_selected=sort(mom_sa,decreasing=TRUE,index.return=TRUE)
sa_selected=mom_sa_selected$ix[1:1500]
sa_data_selected=short_after[sa_selected,]

mom_lb=c()
for (i in 1:nrow_longb){
  inter=max(long_before[i,'MOMENTUM'])
  mom_lb=c(mom_lb,inter)
}
mom_lb_selected=sort(mom_lb,decreasing=TRUE,index.return=TRUE)
lb_selected=mom_lb_selected$ix[1:1500]
lb_data_selected=long_before[lb_selected,]

# write.csv(lb_data_selected,'../observation_processed/long_before.csv')
# write.csv(sa_data_selected,'../observation_processed/short_after.csv')
```

Step 3: Classification model

- **Data source: data/obesrvation_processed**

- **Size: 2 files with a total size of 140kb**

- **Data Processing: Devidide data into training set(1200 observations) and test set(300 obervations)**

- **Classification method: GBM, SVM, NNET, Random Forest, and Logistic. We also taken into account majority vote(GBM,SVM and Random Forest)**

- **Result: as following codes will show**

```r
packages.used=c("gbm", "caret","DMwR" ,"nnet","randomForest","e1071")

# check packages that need to be installed.
packages.needed=setdiff(packages.used,
                        intersect(installed.packages()[,1],
                                  packages.used))
# install additional packages
if(length(packages.needed)>0){
  install.packages(packages.needed, dependencies = TRUE)
}
# install packages
if (!require("pacman")) install.packages("pacman")
```

```
## Loading required package: pacman
```

```r
pacman::p_load(text2vec, plyr,qlcMatrix, kernlab, knitr)
```

Load and Process Data

```r
short_after_selected <- sa_data_selected
long_after_selected <- lb_data_selected

colnames(long_after_selected)[1] <- "y"
long_after_selected$y <- ifelse(long_after_selected$RETURN >0, 1, 0)
long_after_selected <- long_after_selected[,-11]

colnames(short_after_selected)[1] <- "y"
short_after_selected$y <- ifelse(short_after_selected$RETURN >0, 1,0)
short_after_selected <- short_after_selected[,-11]

test.index <- sample(1:1500,300,replace = F)

test.sas <- short_after_selected[test.index,]
test.lbs <- long_after_selected[test.index,]
test.sas.x <- test.sas[,-1]
test.lbs.x <- test.lbs[,-1]
train.sas <- short_after_selected[-test.index,]
train.lbs <- long_after_selected[-test.index,]
```

Load requred functions

```r
source("../lib/evaluation_measures.R")
source("../lib/train.R")
source("../lib/test.R")
source("../lib/cross_validation.R")
```

Short After Model

GBM

```r
# GBM
start.time <- Sys.time()
res_gbm = train.gbm(train.sas)
pred.gbm = test.gbm(res_gbm,test.sas.x)
sas.gbm.sum = table(pred.gbm,test.sas$y)
end.time <- Sys.time()
gbm.sas.time <- end.time-start.time
perf_sas_gbm <- performance_statistics(sas.gbm.sum)
perf_sas_gbm
```

```
## $precision
## [1] 0.7552743
##
## $recall
## [1] 0.9274611
##
## $f1
## [1] 0.8325581
##
## $accuracy
## [1] 0.76
```

SVM

```r
# model.svm <- svm(y ~ ., data = train.sas, cost = 256, gamma = 0.3)
# Tune svm
start.time <- Sys.time()
model.svm.sas <- train.svm(train.sas)
pre.svm <- test.svm(model.svm.sas,test.sas.x)
svm.sas <- table(pre.svm,test.sas$y)
end.time <- Sys.time()
svm.sas.time <- end.time-start.time
perf_sas_svm <- performance_statistics(svm.sas)
perf_sas_svm
```

```
## $precision
## [1] 0.8167539
##
## $recall
## [1] 0.8082902
##
## $f1
## [1] 0.8125
##
## $accuracy
## [1] 0.76
```

BPNN

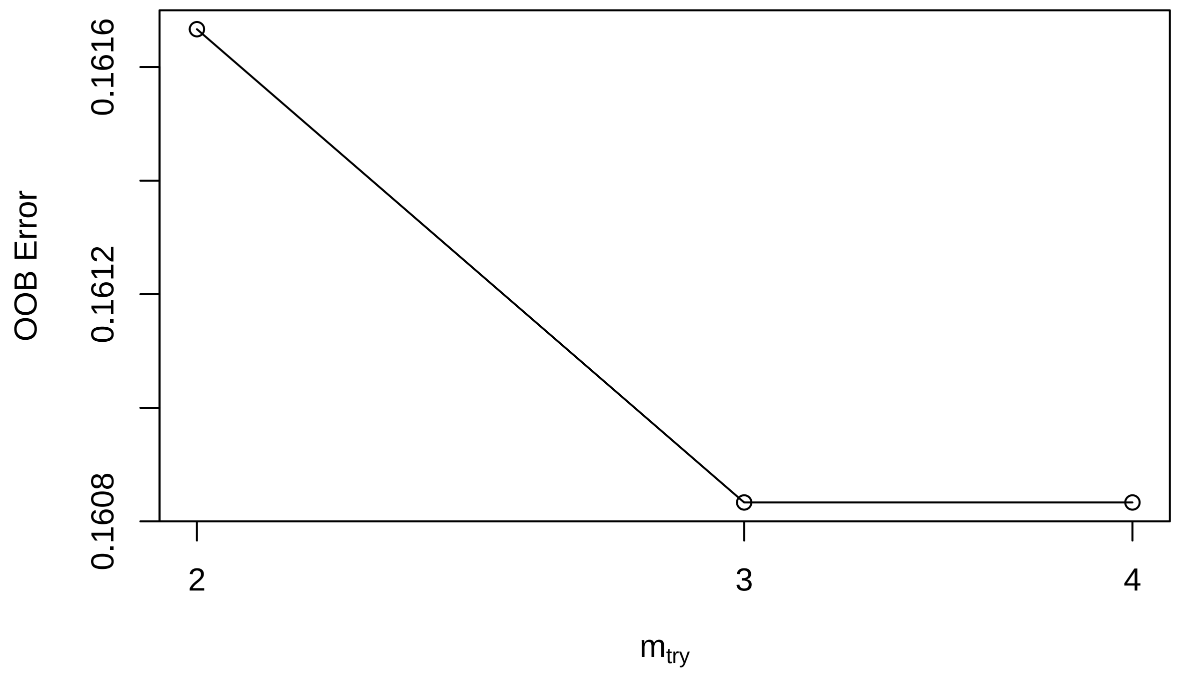
```r
# netural network
start.time <- Sys.time()
# model.nnet <- nnet(y ~ ., data = train.sas, linout = F, size = 10, decay = 0.001, maxit = 200, trace
# Tune bpnn
model.nnet <- train.bp(train.sas)
pre.nnet <- test.bp(model.nnet,test.sas.x)
nnet.sas <- table(pre.nnet,test.sas$y)
end.time <- Sys.time()
nnet.sas.time <- end.time-start.time
perf_sas_nnet <- performance_statistics(nnet.sas)
perf_sas_nnet
```

```
## $precision
## [1] 0.7324561
##
## $recall
## [1] 0.865285
##
## $f1
## [1] 0.7933492
##
## $accuracy
## [1] 0.71
```

Random Forest

```r
# Random Forest
start.time <- Sys.time()
model.rf <- train.rf(train.sas)
```

```
## mtry = 3  OOB error = 16.08%
## Searching left ...
## mtry = 2     OOB error = 16.17%
## -0.005181347 1e-05
## Searching right ...
## mtry = 4     OOB error = 16.08%
## 0 1e-05
```



```
pre.rf <- test.rf(model.rf,test.sas.x)
rf.sas <- table(pre.rf,test.sas$y)
end.time <- Sys.time()
rf.sas.time <- end.time-start.time
perf_sas_rf <- performance_statistics(rf.sas)
perf_sas_rf
```

```
## $precision
## [1] 0.8860104
##
## $recall
## [1] 0.8860104
##
## $f1
## [1] 0.8860104
##
## $accuracy
## [1] 0.8533333
```

Logistic

```
start.time <- Sys.time()
res_logi = train.log(train.sas)
pred.logi = test.log(res_logi,test.sas.x)
log.sas <- table(pred.logi,test.sas$y)
end.time <- Sys.time()
log.sas.time <- end.time-start.time
perf_sas_log <- performance_statistics(log.sas)
perf_sas_log
```

```
## $precision
## [1] 0.6430976
```

```
##
## $recall
## [1] 0.9896373
##
## $f1
## [1] 0.7795918
##
## $accuracy
## [1] 0.64
```

Majority Vote(Equal Weight)

```
# Majority Vote
pre=(as.numeric(as.character(pre.svm))+as.numeric(as.character(pred.gbm))+as.numeric(as.character(pre.r
pre=ifelse(pre>=2,1,0)
mv <- table(pre,test.sas$y)
perf_sas_mv <- performance_statistics(mv)
perf_sas_mv
```

```
## $precision
## [1] 0.8240741
##
## $recall
## [1] 0.9222798
##
## $f1
## [1] 0.8704156
##
## $accuracy
## [1] 0.8233333
```

Long - Before Model

SVM

```
# model.svm <- svm(y ~ ., data = train.lbs, cost = 256, gamma = 0.3)
# Tune svm
start.time <- Sys.time()
model.svm.lbs <- train.svm(train.lbs)
pre.svm <- test.svm(model.svm.lbs,test.lbs.x)
svm.lbs <- table(pre.svm,test.lbs$y)
end.time <- Sys.time()
svm.lbs.time <- end.time-start.time
perf_lbs_svm <- performance_statistics(svm.lbs)
perf_lbs_svm
```

```
## $precision
## [1] 0.6666667
##
## $recall
## [1] 0.7323944
```

```
##
## $f1
## [1] 0.6979866
##
## $accuracy
## [1] 0.7
```

BPNN

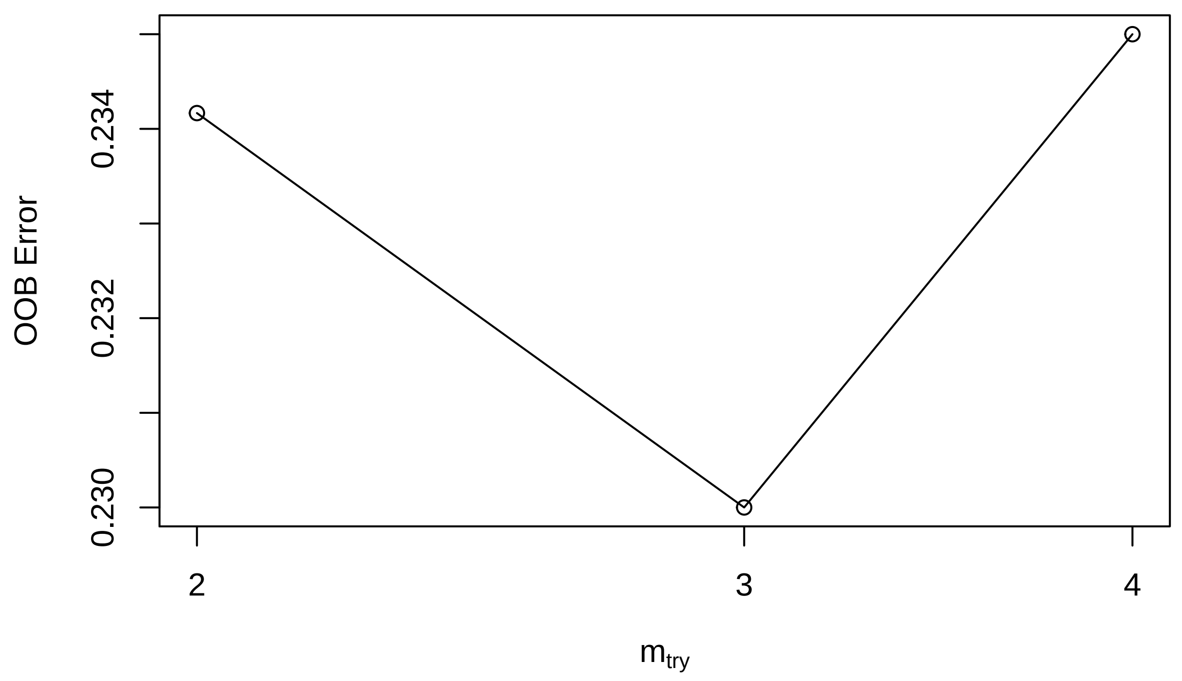
```
# netural network
start.time <- Sys.time()
# model.nnet <- nnet(y ~ ., data = train.sas, linout = F, size = 10, decay = 0.001, maxit = 200, trace
# Tune bpnn
model.nnet <- train.bp(train.lbs)
pre.nnet <- test.bp(model.nnet,test.lbs.x)
nnet.lbs <- table(pre.nnet,test.lbs$y)
end.time <- Sys.time()
nnet.lbs.time <- end.time-start.time
perf_lbs_nnet <- performance_statistics(nnet.lbs)
perf_lbs_nnet
```

```
## $precision
## [1] 0.5416667
##
## $recall
## [1] 0.7323944
##
## $f1
## [1] 0.6227545
##
## $accuracy
## [1] 0.58
```

Random Forest

```
# Random Forest
start.time <- Sys.time()
model.rf <- train.rf(train.lbs)
```

```
## mtry = 3  OOB error = 23%
## Searching left ...
## mtry = 2     OOB error = 23.42%
## -0.01811594 1e-05
## Searching right ...
## mtry = 4     OOB error = 23.5%
## -0.02173913 1e-05
```



```
pre.rf <- test.rf(model.rf,test.lbs.x)
rf.lbs <- table(pre.rf,test.lbs$y)
end.time <- Sys.time()
rf.lbs.time <- end.time-start.time
perf_lbs_rf <- performance_statistics(rf.lbs)
perf_lbs_rf
```

```
## $precision
## [1] 0.7132353
##
## $recall
## [1] 0.6830986
##
## $f1
## [1] 0.6978417
##
## $accuracy
## [1] 0.72
```

Logistic

```
start.time <- Sys.time()
res_logi = train.log(train.lbs)
pred.logi = test.log(res_logi,test.lbs.x)
log.lbs <- table(pred.logi,test.lbs$y)
end.time <- Sys.time()
log.lbs.time <- end.time-start.time
perf_lbs_log <- performance_statistics(log.lbs)
perf_lbs_log
```

```
## $precision
## [1] 0.4755245
```

```
##
## $recall
## [1] 0.4788732
##
## $f1
## [1] 0.477193
##
## $accuracy
## [1] 0.5033333
```

GBM

```
# GBM
start.time <- Sys.time()

res_gbm = train.gbm(train.lbs)
pred.gbm = test.gbm(res_gbm,test.lbs.x)
lbs.gbm.sum = table(pred.gbm,test.lbs$y)

end.time <- Sys.time()
gbm.lbs.time <- end.time-start.time

perf_lbs_gbm <- performance_statistics(lbs.gbm.sum)
perf_lbs_gbm
```

```
## $precision
## [1] 0.5588235
##
## $recall
## [1] 0.5352113
##
## $f1
## [1] 0.5467626
##
## $accuracy
## [1] 0.58
```

Short-After-Model Summary

```
compare_df <- data.frame(method=c("GBM","SVM","NNET","RF","LOGISTIC","MV"),
                         precision=c(perf_sas_gbm$precision,perf_sas_svm$precision,perf_sas_nnet$precisi
                         recall=c(perf_sas_gbm$recall,perf_sas_svm$recall,perf_sas_nnet$recall,perf_sas_
                         f1=c(perf_sas_gbm$f1,perf_sas_svm$f1,perf_sas_nnet$f1,perf_sas_rf$f1,perf_sas_l
                         accuracy=c(perf_sas_gbm$accuracy,perf_sas_svm$accuracy,perf_sas_nnet$accuracy,p
                         time=c(gbm.sas.time,svm.sas.time,nnet.sas.time,rf.sas.time,log.sas.time,"NA"))
kable(compare_df,caption="Comparision of performance for different methods(Short-After-Model)", digits=2
```

Table 1: Comparision of performance for different methods(Short-After-Model)

method	precision	recall	f1	accuracy	time
GBM	0.76	0.93	0.83	0.76	0.13741112 secs

method	precision	recall	f1	accuracy	time
SVM	0.82	0.81	0.81	0.76	33.64355111 secs
NNET	0.73	0.87	0.79	0.71	90.17032194 secs
RF	0.89	0.89	0.89	0.85	4.37895298 secs
LOGISTIC	0.64	0.99	0.78	0.64	0.01925707 secs
MV	0.82	0.92	0.87	0.82	NA

Long-Before-Model Summary

```
compare_df <- data.frame(method=c("GBM","SVM","NNET","RF","LOGISTIC"),
                         precision=c(perf_lbs_gbm$precision,perf_lbs_svm$precision,perf_lbs_nnet$precisi
                         recall=c(perf_lbs_gbm$recall,perf_lbs_svm$recall,perf_lbs_nnet$recall,perf_lbs_
                         f1=c(perf_lbs_gbm$f1,perf_lbs_svm$f1,perf_lbs_nnet$f1,perf_lbs_rf$f1,perf_lbs_l
                         time=c(gbm.lbs.time,svm.lbs.time,nnet.lbs.time,rf.lbs.time,log.lbs.time))

kable(compare_df,caption="Comparision of performance for different methods(Long-Before-Model)", digits=2
```

Table 2: Comparision of performance for different methods(Long-Before-Model)

method	precision	recall	f1	accuracy	time
GBM	0.56	0.54	0.55	0.58	0.42110300 secs
SVM	0.67	0.73	0.70	0.70	37.87903690 secs
NNET	0.54	0.73	0.62	0.58	79.63956594 secs
RF	0.71	0.68	0.70	0.72	4.33613610 secs
LOGISTIC	0.48	0.48	0.48	0.50	0.02247691 secs

Step 4: Including news features from NLP

Part a: Get Data

- **Data source: Bloomberg and WSJ news and headlines**

- **Purpose: Web scrapying using Python from WSJ and Bloomberg. You can find the code at lib/wsj_news_scrape_python/run.py and extract_news.ipynb.**

- **Data Processing: Using scrapy and python package "selenium"**

- **Result: WSJ news stored at data/wsjnews and Bloomberg headlines stored at data/news_Bloomberg**

Part b: NLP

- **Data source: from previous step**

- **Purpose: We would like to analyze if these articles from mainstream media could influence stock market and investor's trading strategie**

- **Data Processing: Extracting features using NLP. Specifically, we use R built-in pacakge “syuzhetand” and word_count.**

- **Result: As follows**

```
# source("../lib/sentimental.r")
```

Part c: Used R to match and clean news data

- **Data source: From previous steps**

- **Purpose: For every news data we found, we use R to find the corresponding indicators(previous financial features) with the same date and ticker. We reorganized them and save them as RData file.**

- **Data Processing: We use R, and you can find the code at lib/features_matching.R**

- **Result: RData file for futher processing.**

```
# source("../lib/features_matching.r")
```

Step 5:

- **Data source: data/all_features.csv**

- **Purpose: Do classification and model evluation based on new features extracted from news headlines**

- **Data Processing: Using R to run classification method including GBM, SVM, NNET, Random Forest, and Logistic as in step 3.**

- **Result: As below**

```
setwd("~/Spr2017-proj5-grp15/data/original/")
all.features <- read.csv("../all_features.csv")
```

Data Process

```
all.features[is.na(all.features)] <- 0
all.features$X <- all.features$y.return01
all.features <- all.features[,-ncol(all.features)]
colnames(all.features)[1] <- "y"
all.features <- all.features[,-c(2,3)]

n <- nrow(all.features)

test.ind <- sample(1:n,n/3,replace = F)
```

```r
test.feature <- all.features[test.ind,]
test.feature.x <- test.feature[,-1]
train.feature <- all.features[-test.ind,]
```

GBM

```r
# GBM
start.time <- Sys.time()
res_gbm = train.gbm(train.feature)
pred.gbm = test.gbm(res_gbm,test.feature.x)
feature.gbm.sum = table(pred.gbm,test.feature$y)
end.time <- Sys.time()
gbm.feature.time <- end.time-start.time
perf_feature_gbm <- performance_statistics(feature.gbm.sum)
perf_feature_gbm
```

```
## $precision
## [1] 0.6
##
## $recall
## [1] 0.75
##
## $f1
## [1] 0.6666667
##
## $accuracy
## [1] 0.64
```

SVM

```r
# model.svm <- svm(y ~ ., data = train.sas, cost = 256, gamma = 0.3)
# Tune svm
start.time <- Sys.time()
model.svm.feature <- train.svm(train.feature)
pre.svm <- test.svm(model.svm.feature,test.feature.x)
svm.feature <- table(pre.svm,test.feature$y)
end.time <- Sys.time()
svm.feature.time <- end.time-start.time
perf_feature_svm <- performance_statistics(svm.feature)
perf_feature_svm
```

```
## $precision
## [1] 0.5
##
## $recall
## [1] 0.7083333
##
## $f1
## [1] 0.5862069
##
## $accuracy
## [1] 0.52
```

NNET

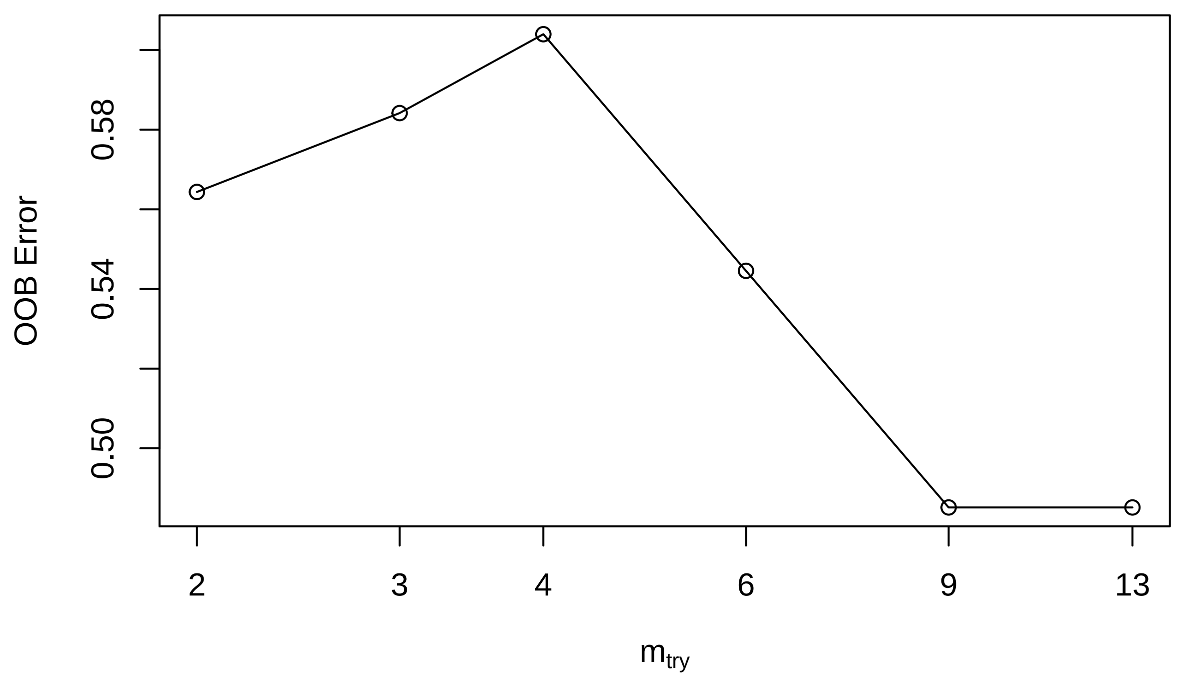
```
# netural network
start.time <- Sys.time()
# model.nnet <- nnet(y ~ ., data = train.sas, linout = F, size = 10, decay = 0.001, maxit = 200, trace
# Tune bpnn
model.nnet <- train.bp(train.feature)
pre.nnet <- test.bp(model.nnet,test.feature.x)
nnet.feature <- table(pre.nnet,test.feature$y)
end.time <- Sys.time()
nnet.feature.time <- end.time-start.time
perf_feature_nnet <- performance_statistics(nnet.feature)
perf_feature_nnet
```

```
## $precision
## [1] 0.5263158
##
## $recall
## [1] 0.8333333
##
## $f1
## [1] 0.6451613
##
## $accuracy
## [1] 0.56
```

Random Forest

```
# Random Forest
start.time <- Sys.time()
model.rf <- train.rf(train.feature)
```

```
## mtry = 4  OOB error = 60.4%
## Searching left ...
## mtry = 3     OOB error = 58.42%
## 0.03278689 1e-05
## mtry = 2     OOB error = 56.44%
## 0.03389831 1e-05
## Searching right ...
## mtry = 6     OOB error = 54.46%
## 0.03508772 1e-05
## mtry = 9     OOB error = 48.51%
## 0.1090909 1e-05
## mtry = 13    OOB error = 48.51%
## 0 1e-05
```



```
pre.rf <- test.rf(model.rf,test.feature.x)
rf.feature <- table(pre.rf,test.feature$y)
end.time <- Sys.time()
rf.feature.time <- end.time-start.time
perf_feature_rf <- performance_statistics(rf.feature)
perf_feature_rf
```

```
## $precision
## [1] 0.5428571
##
## $recall
## [1] 0.7916667
##
## $f1
## [1] 0.6440678
##
## $accuracy
## [1] 0.58
```

Logistic

```
start.time <- Sys.time()
res_logi = train.log(train.feature)
pred.logi = test.log(res_logi,test.feature.x)
log.feature <- table(pred.logi,test.feature$y)
end.time <- Sys.time()
log.feature.time <- end.time-start.time
perf_feature_log <- performance_statistics(log.feature)
perf_feature_log
```

```
## $precision
## [1] 0.5769231
```

```
##
## $recall
## [1] 0.625
##
## $f1
## [1] 0.6
##
## $accuracy
## [1] 0.6
```

Model Summary

```
compare_df <- data.frame(method=c("GBM","SVM","NNET","RF","LOGISTIC"),
                         precision=c(perf_feature_gbm$precision,perf_feature_svm$precision,perf_feature_
                         recall=c(perf_feature_gbm$recall,perf_feature_svm$recall,perf_feature_nnet$reca
                         f1=c(perf_feature_gbm$f1,perf_feature_svm$f1,perf_feature_nnet$f1,perf_feature_
                         time=c(gbm.feature.time,svm.feature.time,nnet.feature.time,rf.feature.time,log

kable(compare_df,caption="Comparision of performance for different methods", digits=2)
```

Table 3: Comparision of performance for different methods

method	precision	recall	f1	accuracy	time
GBM	0.60	0.75	0.67	0.64	0.037508965 secs
SVM	0.50	0.71	0.59	0.52	2.232439041 secs
NNET	0.53	0.83	0.65	0.56	22.089481115 secs
RF	0.54	0.79	0.64	0.58	0.631005049 secs
LOGISTIC	0.58	0.62	0.60	0.60	0.008542061 secs

Two Models Comparison

```
compare_df <- data.frame(model=c("Short-After-Model","NLP"),
                         precision=c(perf_sas_rf$precision,perf_feature_nnet$precision),
                         recall=c(perf_sas_rf$recall,perf_feature_nnet$recall),
                         f1=c(perf_sas_rf$f1,perf_feature_nnet$f1),                  accuracy=c(perf_
                         time=c(rf.sas.time,nnet.feature.time))

kable(compare_df,caption="Comparision of performance for two different models", digits=2)
```

Table 4: Comparision of performance for two different models

model	precision	recall	f1	accuracy	time
Short-After-Model	0.89	0.89	0.89	0.85	4.378953 secs
NLP	0.53	0.83	0.65	0.56	22.089481 secs

Conclusion

- Stock market is known as a chaotic system. We get more information on news by limiting our scope to earning release day, and are able to build the prediction model of more than 85% accuracy. However, with the help of NLP our model does not improve but worse off. We may need to reconsider the weight of each features and do better feature selection.

- Models and features: GBM, SVM, NNET, RF, Logistic, and Random forest. Random forest gives us the best among all of them

- Further, it is observed that data visualization from different companies shows that data set from a certain company is much more distinguishable than mixing data (portfolio and index)

- Compared with our goal, we just made it. We may check whether the strategy still works with recent 10 years data, improve the model with statistical technique and redict whether to long or short before & after Earnings Release and so on.

- We use their core value: long before Earnings Release, Short after Earnings Release and use machine learning model to improve it

- NLP analysis: The sentiment result from toolkit is too general to apply for financial news and our word vector list is quite limited

- Model Improvement: high dimension can be handled better