Appendix

Data description and preparation

The data are obtained from CNNpred: CNN-based stock market prediction using a diverse set of variables Data Set. There are 1985 rows (including header) and 84 columns in the data set.

The second column Close in this data set represents the index value S&P 500 at market close in each trading day. We create a new column called Close30. Close30 is the Close value stored in the second column and is shifted backwards by 30 rows. Therefore, it shows the Close after 30 trading days for each row and this is our response variable. Now, we have 85 columns in the data set. In these 85 columns, we have excluded Name, Date and Close30 as our explanatory variables. The reason is as follows. First, since all the values in the Name column are “S & P”, we can remove them to simplify the data set. Second, for simplicity, we use the number of consecutive trading days missed rather than consecutive calendar days missed for visualization in data cleansing sessions. Therefore, we can also remove the Date column to simplify the data set. Lastly, reminding that Close30 is our response variable, thus only 82 columns are used as predictor variables to estimate the price of S&P 500 in the near future, indicated by Close30.

Data Cleansing

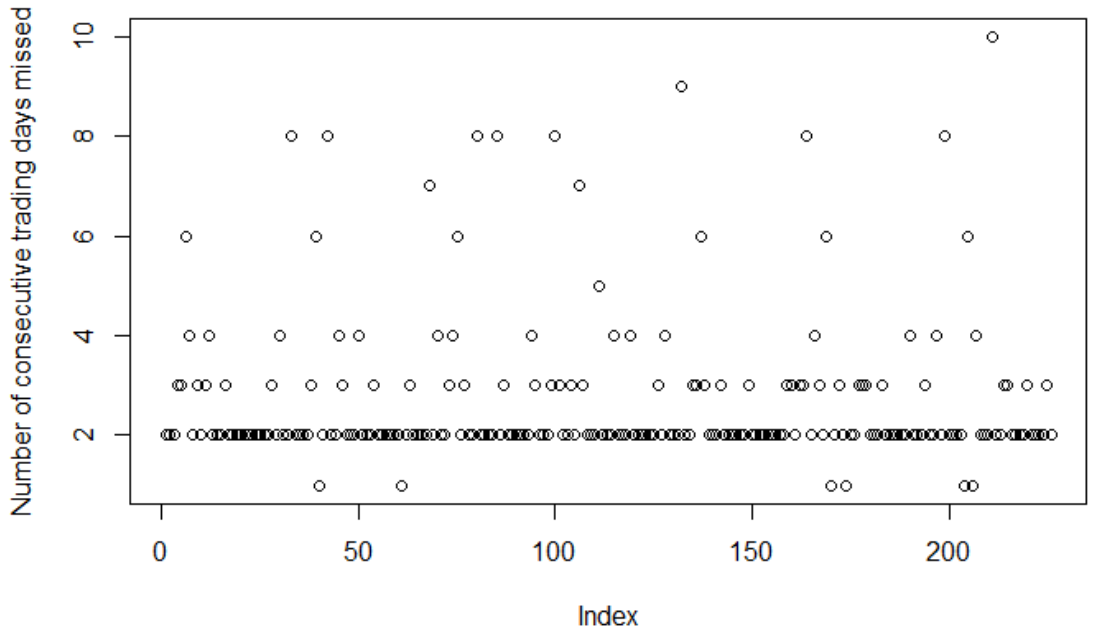


Figure 1

Number of consecutive trading days missed	1	2	3	4	5
Count	6	148	39	15	1

Number of consecutive trading days missed	6	7	8	9	10
Count	6	2	7	1	1

Table 1: Count data for Figure 1

Before data cleansing, a total of 870 trading days were missed due to missing values in the data set. It is worth noting that the total number of trading days missed in the above table is only 609. However, 201 and 60 trading days at the beginning and the end of the data set are also excluded respectively. Adding 201 and 60 to 609 makes the total number of missing trading days 870.

EMA_200 is one of the main reasons that the first 201 trading days cannot be used. EMA_200 is the 200 days exponential moving average of the index. Therefore, we cannot use the first 199 trading days in the data set if we would like to use this predictor for machine learning. Besides, the last 60 trading days also cannot be used. One of the main reasons is due to missing data of SSEC (Shanghai Composite Index) in this dataset. Therefore, we cannot use the last 60 trading days too.

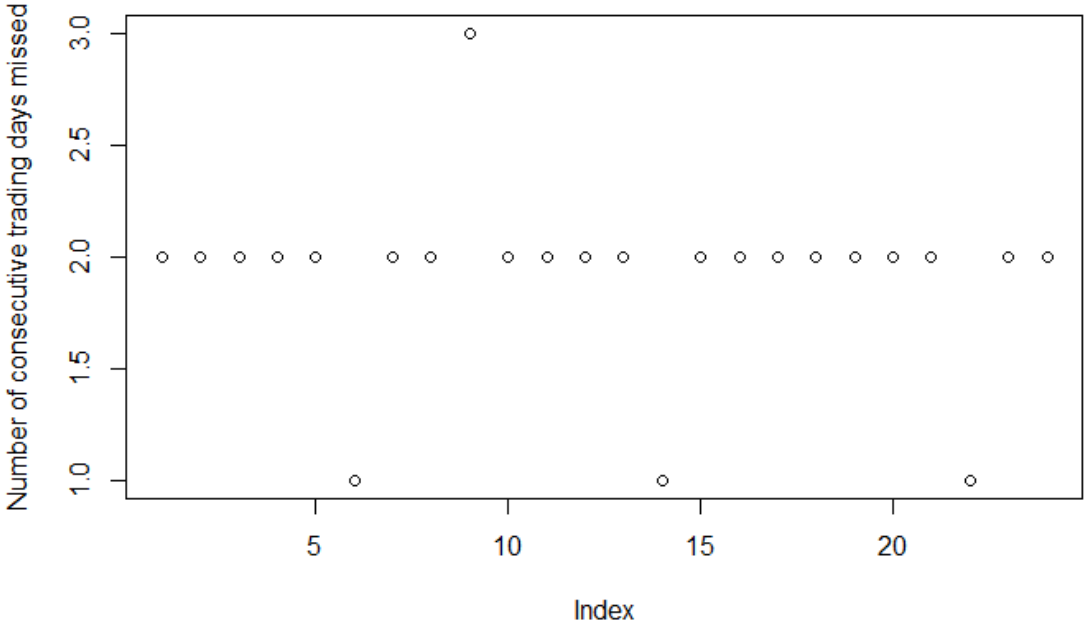


Figure 2

Number of consecutive trading days missed	1	2	3
Count	3	20	1

Table 2: Count data for Figure 2

Due to the fact that there are 870 missing trading days in Figure 1 which cannot be used in machine learning, we decided to eliminate the predictors with more than 50 missing values. After data cleansing, 11 predictors including but not limited to EMA_200 and SSEC, which is mentioned in Figure 1, are eliminated.

After data cleansing, only 125 trading days were missed due to missing values in the data set. The total number of trading days missed in the above table is only 46, which is greatly reduced compared with before data cleansing. However, the 49 and 30 trading days at the beginning and the end of the data set are also excluded respectively. The first 49 trading days cannot be used mainly due to EMA_50. EMA_50 is the 50 days exponential moving average of the index. The last 30 trading days cannot be used as the data set does not provide further Close value after 30 trading days. Therefore, we cannot use the first 49 trading days and the last 30 trading days in the data set for doing our analysis.

---R Code---

import necessary library

```
# glmnet for lasso and ridge
# leaps for subset selection
#install.packages("glmnet")
#install.packages("leaps")
require(glmnet)
```

```
## Loading required package: glmnet
```

```
## Warning: package 'glmnet' was built under R version 4.0.5
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-3
```

```
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 4.0.5
```

read dataset

```
file_path = "D:/HKU/Year3 Sem1/STAT3612/Project/Processed_S&P.csv" # replace
it with actual path

# SP500: original dataset
# Close_after30: vector of Close after 30 days
# SP500Num: SP500 with added column Close_after30 and without string columns,
i.e. Date and Name
# SPNONA: SP500Num without rows with missing data
SP500 <- read.csv(file_path)
SP500Num <- subset(SP500, select = -c(Date, Name) )
Close_after30 = SP500Num[31:1984, 1]
SP500Num["Close30"] <- c(Close_after30, rep(NA,
nrow(SP500Num)-length(Close_after30)))
SP500NONA <- SP500Num[complete.cases(SP500Num), ]
head(SP500NONA)
```

```
##       Close       Volume           mom          mom1          mom2
mom3
## 202 1165.90  0.258439793 -1.587725e-02  7.243745e-03  2.027485e-03
-0.003641386
## 203 1178.17 -0.102183525  1.052408e-02 -1.587725e-02  7.243745e-03
0.002027485
## 204 1180.26 -0.080035721  1.773909e-03  1.052408e-02 -1.587725e-02
0.007243745
## 207 1185.64 -0.004192942  1.688568e-05  2.146971e-03  2.389258e-03
0.001773909
## 208 1182.45  0.031398679 -2.690584e-03  1.688568e-05  2.146971e-03
```

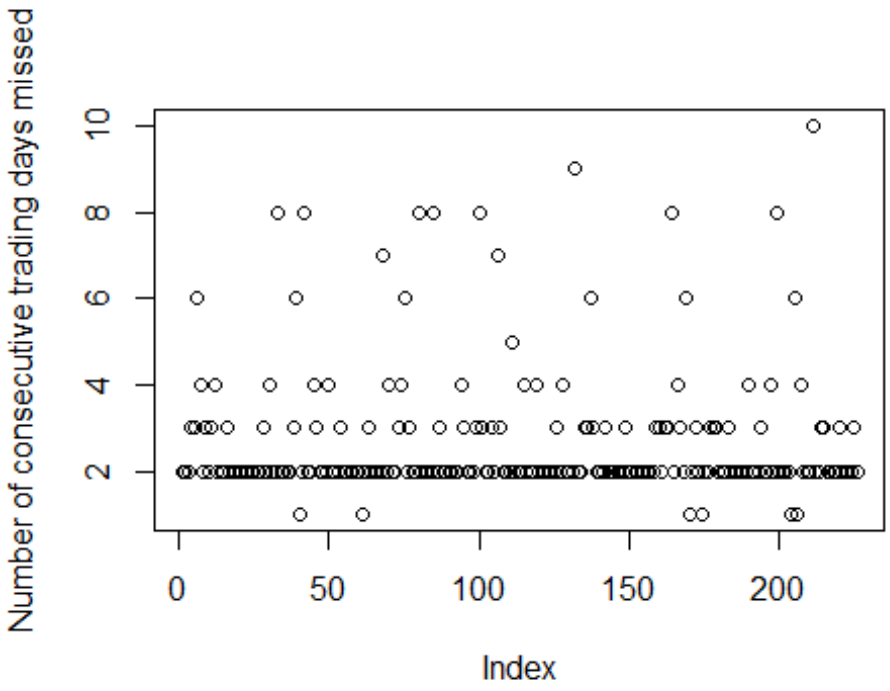
```
0.002389258
## 209 1183.78 -0.012041968  1.124849e-03 -2.690584e-03  1.688568e-05
0.002146971
##            ROC_5    ROC_10   ROC_15   ROC_20   EMA_10   EMA_20   EMA_50
## 202 -0.330833919 0.4436807 1.585787 2.291670 1168.304 1156.447 1132.269
## 203  0.005947543 1.5690124 2.921219 3.869416 1170.097 1158.516 1134.069
## 204  0.549488476 1.9169948 3.422718 4.927861 1171.945 1160.586 1135.881
## 207  1.693111810 1.3566765 2.144305 3.305748 1177.825 1166.883 1141.415
## 208  0.363267342 0.3692365 1.937979 3.295098 1178.666 1168.366 1143.024
## 209  0.298240978 0.8493683 2.220953 3.731167 1179.596 1169.834 1144.622
##      EMA_200 DTB4WK DTB3 DTB6 DGS5 DGS10          Oil         Gold DAAA
DBAA
## 202 1121.503   0.14 0.14 0.18 1.11  2.50 -0.042017818 -0.020661913 4.70
5.75
## 203 1122.066   0.14 0.14 0.17 1.11  2.51  0.029659419  0.000000000 4.69
5.74
## 204 1122.645   0.13 0.13 0.18 1.15  2.57 -0.023190529  0.003360717 4.71
5.77
## 207 1124.482   0.14 0.14 0.18 1.27  2.67  0.007317073 -0.005981308 4.78
5.80
## 208 1125.059   0.14 0.14 0.18 1.34  2.75 -0.008474576 -0.003760812 4.80
5.84
## 209 1125.643   0.14 0.13 0.18 1.23  2.69  0.003663004  0.006795017 4.81
5.85
##              GBP          JPY          CAD           CNY          AAPL
## 202 -0.012013369  0.004236582  0.015602725  6.029363e-05 -0.0267610008
## 203  0.009047941 -0.005972382 -0.010358180  8.892774e-04  0.0033603574
## 204 -0.008405753  0.003022676  0.003521471 -2.560048e-04 -0.0032524427
## 207  0.008939069  0.006334686  0.004120071  4.963750e-04 -0.0025579783
## 208 -0.004794916  0.004204765  0.004337632  2.691122e-03 -0.0007140887
## 209  0.000000000  0.000000000  0.000000000  1.064564e-03 -0.0084138027
##             AMZN           GE           JNJ           JPM         MSFT
## 202 -0.029897286 -0.011076923 -0.0089257750 -0.0133508373 -0.027885360
## 203  0.000000000 -0.001244617  0.0048980407  0.0108781908  0.008366494
## 204  0.039705068  0.003115327  0.0061321386 -0.0104986095  0.004346148
## 207  0.005621284  0.006226713 -0.0021881838  0.0035069059  0.028185747
## 208 -0.014357176 -0.003093997 -0.0042293233  0.0091397847  0.005791467
## 209 -0.003999755 -0.001862259 -0.0001572912 -0.0007992275  0.008829252
##              WFC          XOM         FCHI         FTSE        GDAXI
## 202 -0.012866988 -0.017501449 -0.007127416 -0.006721828 -0.003980576
## 203  0.042769900  0.013667060  0.005510650  0.004382966  0.005216682
## 204  0.016796914  0.004696228  0.013092517  0.005062054  0.013251483
## 207  0.007387286  0.004984925 -0.004480643 -0.007771244 -0.003827286
## 208 -0.006175222 -0.012926514 -0.009575175 -0.010740597 -0.006924885
## 209  0.007378680  0.008375255  0.004997698  0.005650000  0.004153439
##               DJI          HSI          IXIC          SSEC          RUT
## 202 -0.0148128947  0.012541857 -0.0176202956  0.0157774598 -0.022502895
## 203  0.0117819552 -0.008720452  0.0083875100  0.0006988687  0.011467205
## 204  0.0034750352  0.003947126  0.0009278255 -0.0067973634 -0.005682856
## 207  0.0004846051 -0.001128747  0.0025854390 -0.0032362004 -0.001356174
```

```
## 208 -0.0038658707 -0.018501577  0.0023905798 -0.0146290463 -0.003819350
## 209 -0.0011081941  0.001997847  0.0016419018 -0.0014911319 -0.004543901
##             NYSE  TE1  TE2  TE3   TE5  TE6  DE1  DE2  DE4  DE5  DE6
CTB3M
## 202 -0.019475399 2.36 2.36 2.32  0.00 0.04 1.05 3.25 5.57 5.61 5.61
0.00000000
## 203  0.013492037 2.37 2.37 2.34  0.00 0.03 1.05 3.23 5.57 5.60 5.60
-0.05555556
## 204 -0.001081917 2.44 2.44 2.39  0.00 0.05 1.06 3.20 5.59 5.64 5.64
0.05882353
## 207 -0.002064576 2.53 2.53 2.49  0.00 0.04 1.02 3.13 5.62 5.66 5.66
0.00000000
## 208 -0.006630064 2.61 2.61 2.57  0.00 0.04 1.04 3.09 5.66 5.70 5.70
0.00000000
## 209  0.003205507 2.55 2.56 2.51 -0.01 0.04 1.04 3.16 5.67 5.72 5.71
0.00000000
##           CTB6M       CTB1Y   AUD Brent CAC.F copper.F WIT.oil DAX.F DJI.F
## 202 -0.02631579 -0.02631579 -2.09 -3.88 -0.73    -2.47   -4.32 -0.44 -0.92
## 203 -0.05405405  0.00000000  1.83  3.08  0.58     0.88    2.87  0.61  1.10
## 204  0.05714286  0.03603604 -0.89 -2.12  1.31    -0.32   -1.48  1.16  0.18
## 207  0.08108108  0.05833333 -0.56  0.14 -0.57     0.18    0.04 -0.25  0.08
## 208  0.00000000  0.05511811 -1.37 -0.51 -0.94    -2.36   -0.74 -0.80 -0.47
## 209 -0.07500000 -0.08208955  0.70  0.43  0.62     0.32    0.29  0.48 -0.21
##       EUR FTSE.F gold.F HSI.F KOSPI.F NASDAQ.F GAS.F Nikkei.F   NZD
silver.F
## 202 -1.53  -0.72  -2.63  1.55   -1.21    -1.29  2.39     0.42 -1.84
-2.59
## 203  1.69   0.39   0.61 -1.05    0.79     0.79  0.74    -1.57  1.40
1.03
## 204 -0.34   0.60  -1.38  0.50    0.41    -0.18 -4.83    -0.11 -0.98
-3.69
## 207 -0.75  -0.79  -0.02 -0.56   -0.08     0.36  1.12    -0.74 -0.52
2.42
## 208 -0.66  -1.04  -1.18 -2.12   -0.74     0.37 -1.85     0.53  0.03
-2.97
## 209  1.18   0.55   1.51  0.23    0.06     0.09 18.17    -0.43  0.86
2.02
##     RUSSELL.F S.P.F   CHF Dollar.index.F Dollar.index wheat.F   XAG   XAU
## 202     -1.74 -1.23  1.27           1.66         1.62   -2.58 -4.14 -2.51
## 203      0.91  0.94 -1.01          -1.29        -1.29    2.52  2.44  0.82
## 204     -0.53  0.09  0.64           0.28         0.32   -3.09 -3.26 -1.55
## 207     -0.23  0.00  1.17           0.79         0.79    3.65  0.85 -0.04
## 208     -0.26 -0.34  0.47           0.60         0.57    0.61 -1.22 -1.09
## 209     -0.84  0.04 -0.53          -1.12        -1.07    2.06  1.95  1.44
##     Close30
## 202 1206.07
## 203 1221.53
## 204 1224.71
## 207 1228.28
```

```
## 208 1233.00
## 209 1240.40
```

data cleansing: view consecutive rows with missing data

```
# has_NAs: vector of whether the row in SP500Num has missing data
# skipped_dates: vector of numbers of consecutive days that has any missing
data
has_NAs <- !complete.cases(SP500Num)
count <- 0
skipped_dates <- c()
for (has_NA in has_NAs) {
  if(!has_NA){
    if(count != 0){
    skipped_dates <- c(skipped_dates, count)
    count = 0
    }
  } else {
    count = count + 1
  }
}

plot(skipped_dates[-1], ylab = "Number of consecutive trading days missed")
```

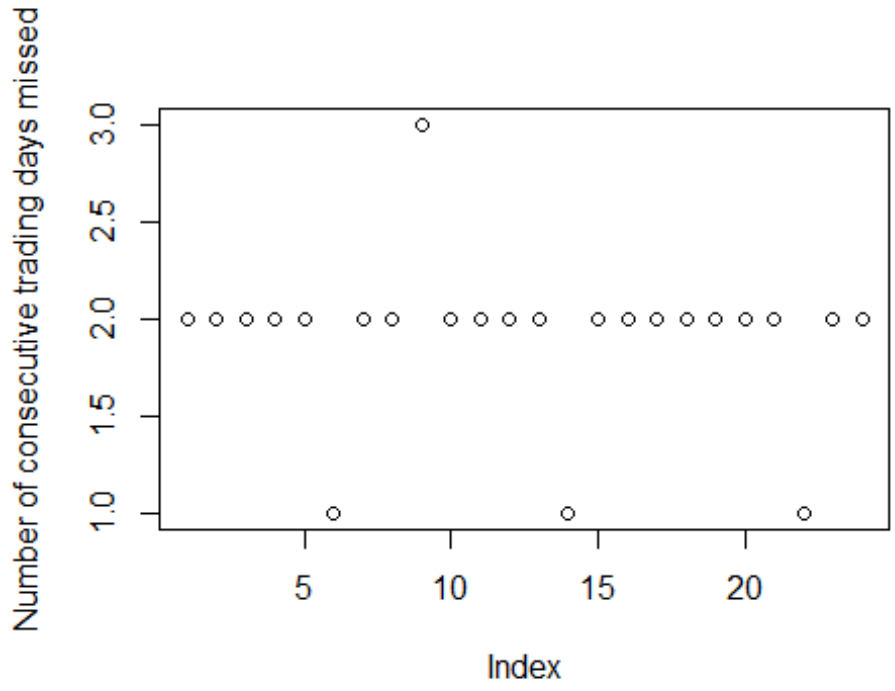



```
table(skipped_dates[-1])
```

```
##
##   1   2   3   4   5   6   7   8   9  10
##   6 148  39  15   1   6   2   7   1   1
```

data cleansing: remove columns that have many missing data (>50)

```
# SP500NumFew: SP500Num with columns that have few missing data
SP500NumFew <- SP500Num[,colSums(is.na(SP500Num))<50]
SP500NONAFew <- SP500NumFew[complete.cases(SP500NumFew), ]
has_NAs <- !complete.cases(SP500NumFew)
count <- 0
skipped_dates <- c()
for (has_NA in has_NAs) {
  if(!has_NA){
    if(count != 0){
    skipped_dates <- c(skipped_dates, count)
    count = 0
    }
  } else {
    count = count + 1
  }
}

plot(skipped_dates[-1], ylab = "Number of consecutive trading days missed")
```




```
table(skipped_dates[-1])
```

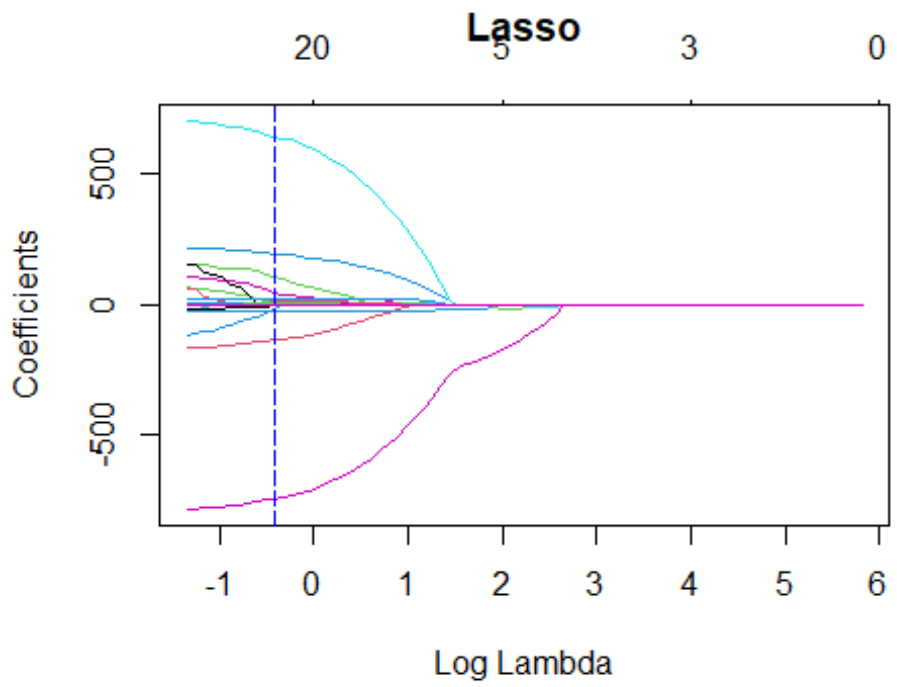
```
##
##  1  2  3
##  3 20  1
```

train/test split: 80/20

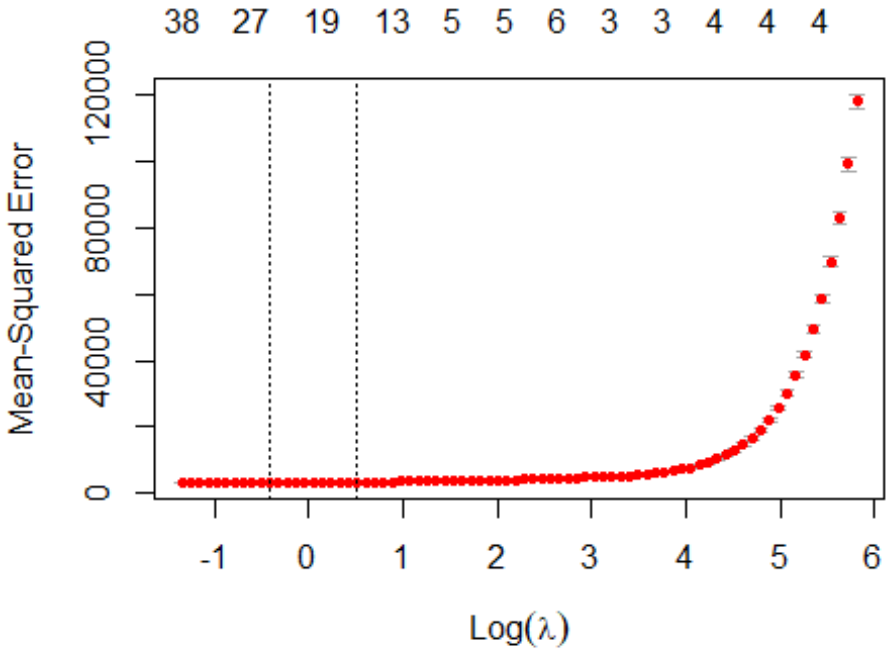
```
train_set <- SP500NONAFew[1:round(nrow(SP500NONAFew)*0.8),]
test_set <-
SP500NONAFew[(round(nrow(SP500NONAFew)*0.8)+1):nrow(SP500NONAFew),]
x_train <- data.matrix(train_set[, !names(train_set) %in% c("Close30")])
y_train <- train_set[["Close30"]]
x_test <- data.matrix(test_set[, !names(test_set) %in% c("Close30")])
y_test <- test_set[["Close30"]]
```

apply Lasso to find the best lambda, test MSE and number of variables

```
set.seed(1)
model_l1 <- glmnet(x_train, y_train,family = "gaussian", alpha = 1)
plot(model_l1, xvar='lambda', main="Lasso")
model_l1_cv <- cv.glmnet(x_train,y_train,family = "gaussian", alpha = 1)
best_l1_lambda <- model_l1_cv$lambda.min
plot(model_l1, xvar='lambda', main="Lasso")
abline(v=log(best_l1_lambda), col="blue", lty=5.5 )
```



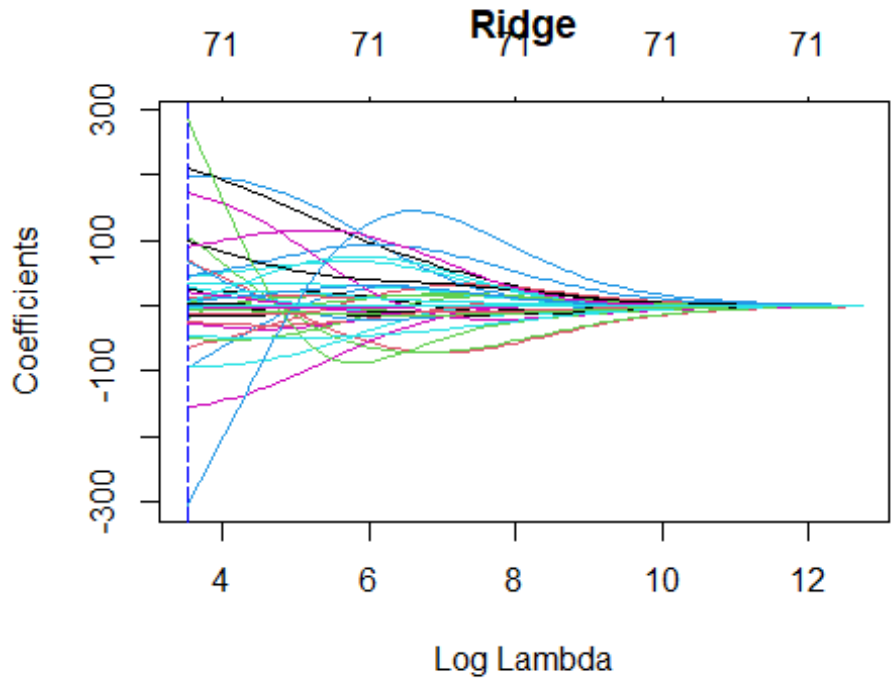

```
plot(model_l1_cv)
```



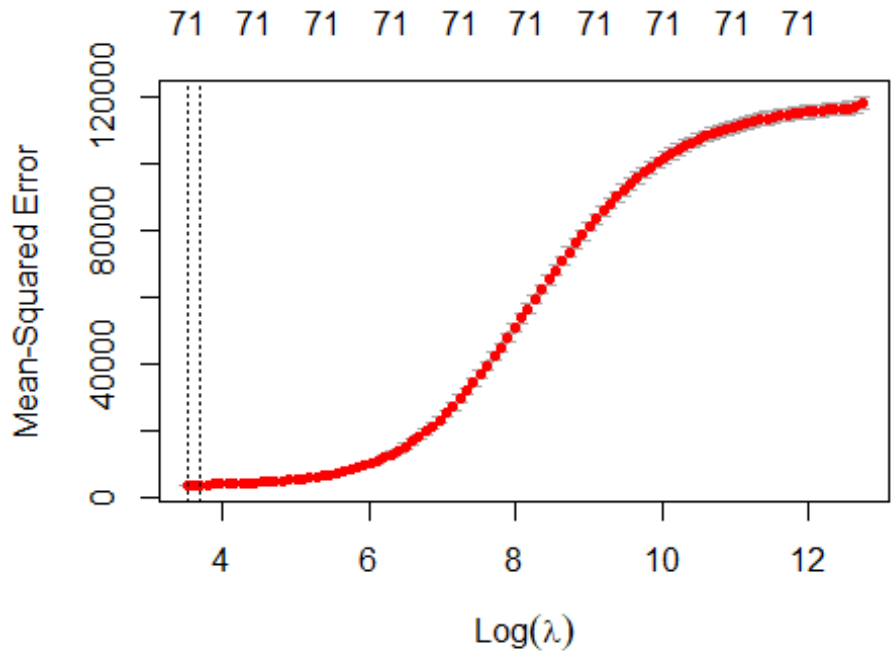

```
l1_predict <- predict(model_l1, s = best_l1_lambda, newx = x_test)
selected_index = which(coef(model_l1_cv, s = "lambda.min") != 0)
selected_variables = colnames(x_train)[selected_index]
message("best lambda = ", best_l1_lambda)

## best lambda = 0.661972359185739

message("test_set MSE = ", mean((l1_predict-y_test)^2))

## test_set MSE = 8154.62740354259

message("number of variables selected = ", length(selected_variables))

## number of variables selected = 25
```

apply Ridge to find the best lambda and test MSE

```
set.seed(1)
model_l2 <- glmnet(x_train,y_train,family = "gaussian", alpha = 0)
plot(model_l2, xvar='lambda', main="Ridge")
model_l2_cv <- cv.glmnet(x_train,y_train,family = "gaussian", alpha = 0)
best_l2_lambda <- model_l2_cv$lambda.min
plot(model_l2, xvar='lambda', main="Ridge")
abline(v=log(best_l2_lambda), col="blue", lty=5.5 )
```




```
plot(model_l2_cv)
```



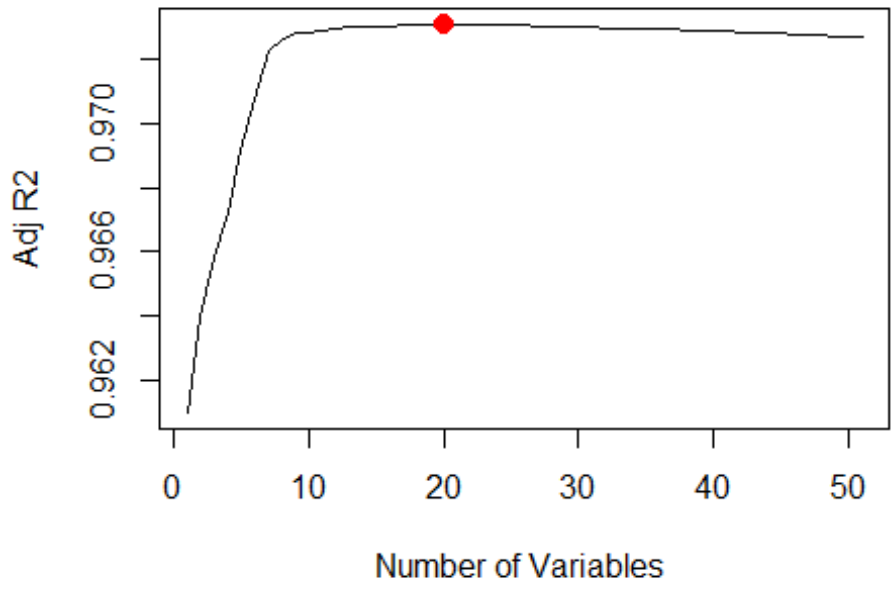
```
l2_predict <- predict(model_l2, s = best_l2_lambda, newx = x_test)
message("best lambda = ", best_l2_lambda)

## best lambda = 33.7217855968576

message("test_set MSE = ", mean((l2_predict-y_test)^2))

## test_set MSE = 3000.20264242421
```

cannot apply best subset selection directly due to large number of predictors

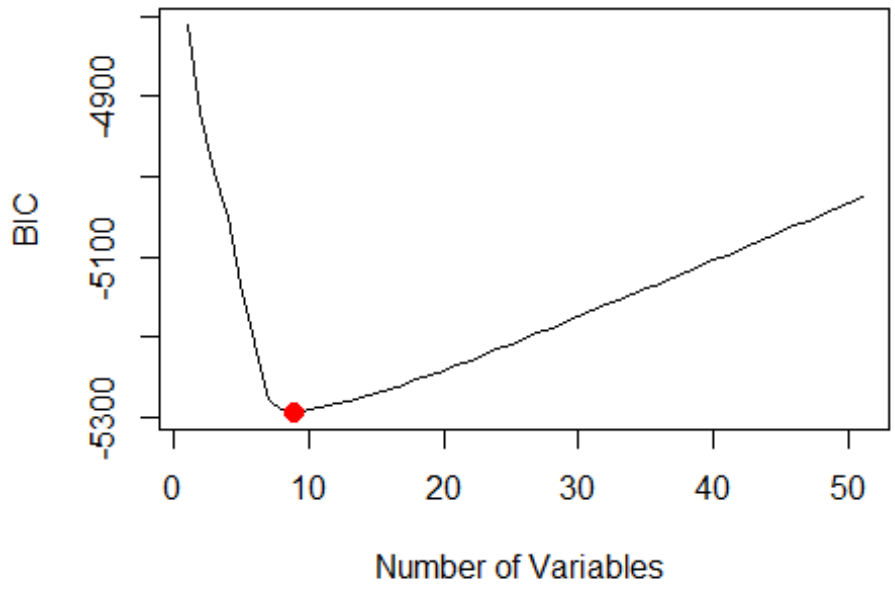
```
#selection_full <- regsubsets (Close30 ~ .,data = train_set, nvmax=10)
# returns error
```

preform forward stepwise selection and use adjusted R-squared, BIC and Cp to find the optimal model

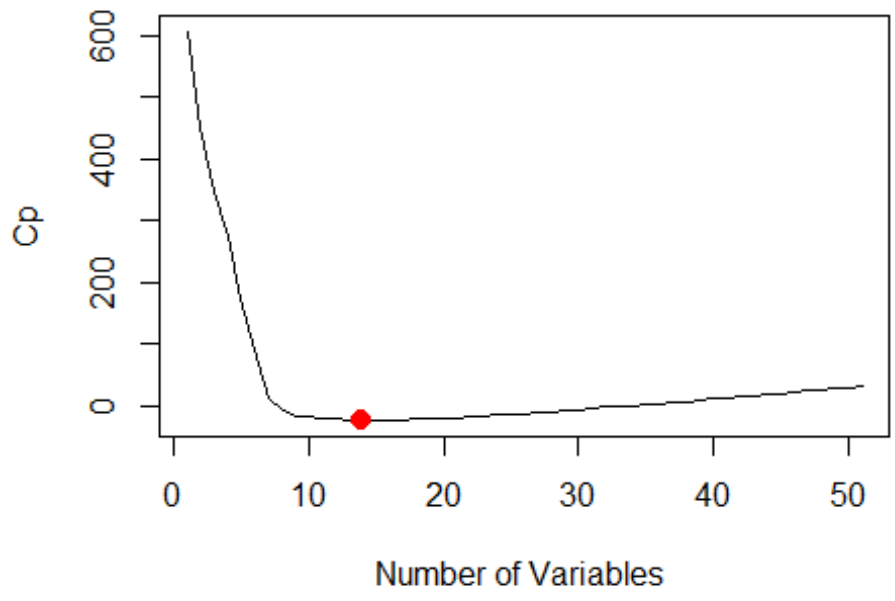
```
selection_forward <- regsubsets (Close30 ~ .,data=train_set ,nvmax=50, method
="forward")

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 10 linear dependencies found

## Reordering variables and trying again:

summary_ <- summary(selection_forward)
plot(summary_$adjr2, xlab="Number of Variables ", ylab="Adj R2", type="l")
varF.adjr2 <- which.max(summary_$adjr2)
points(varF.adjr2,summary_$adjr2[varF.adjr2], col="red",cex=2,pch=20)
```




```
plot(summary_$bic, xlab="Number of Variables ", ylab="BIC", type="l")
varF.bic <- which.min(summary_$bic)
points(varF.bic,summary_$bic[varF.bic], col="red",cex=2,pch=20)
```



```
plot(summary_$cp, xlab="Number of Variables ", ylab="Cp", type="l")
varF.cp <- which.min(summary_$cp)
points(varF.cp,summary_$cp[varF.cp], col="red",cex=2,pch=20)
```

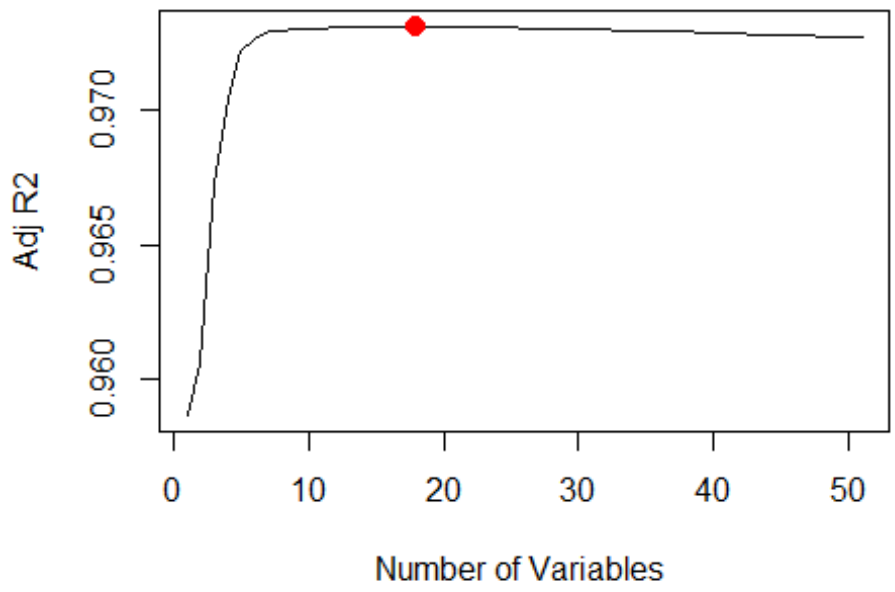



```
message("number of variables for best adjusted R-squared = ", varF.adjr2)

## number of variables for best adjusted R-squared = 20

message("number of variables for best BIC = ", varF.bic)

## number of variables for best BIC = 9

message("number of variables for best Cp = ", varF.cp)

## number of variables for best Cp = 14
```

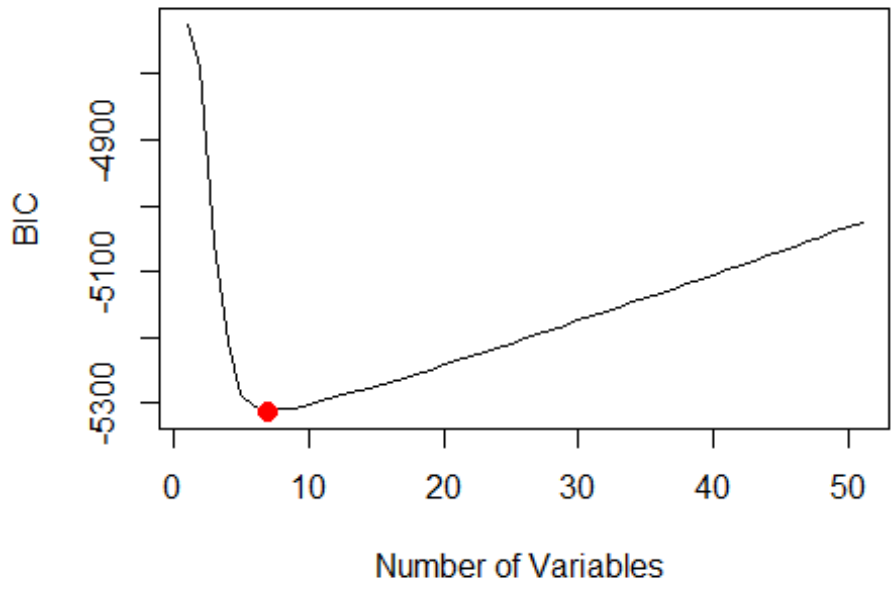
preform backward stepwise selection and use adjusted R-squared, BIC and Cp to find the optimal model

```
selection_backward <- regsubsets (Close30 ~ .,data=train_set ,nvmax=50,
method ="backward")

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 10 linear dependencies found

## Reordering variables and trying again:
```

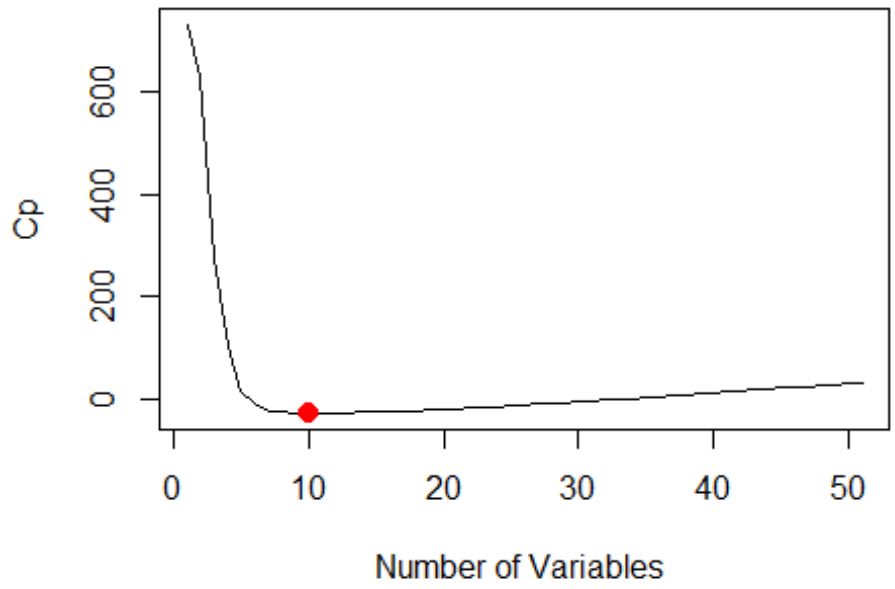
```
summary_ <- summary(selection_backward)
plot(summary_$adjr2, xlab="Number of Variables ", ylab="Adj R2", type="l")
varB.adjr2 <- which.max(summary_$adjr2)
points(varB.adjr2,summary_$adjr2[varB.adjr2], col="red",cex=2,pch=20)
```




```
plot(summary_$bic, xlab="Number of Variables ", ylab="BIC", type="l")
varB.bic <- which.min(summary_$bic)
points(varB.bic,summary_$bic[varB.bic], col="red",cex=2,pch=20)
```



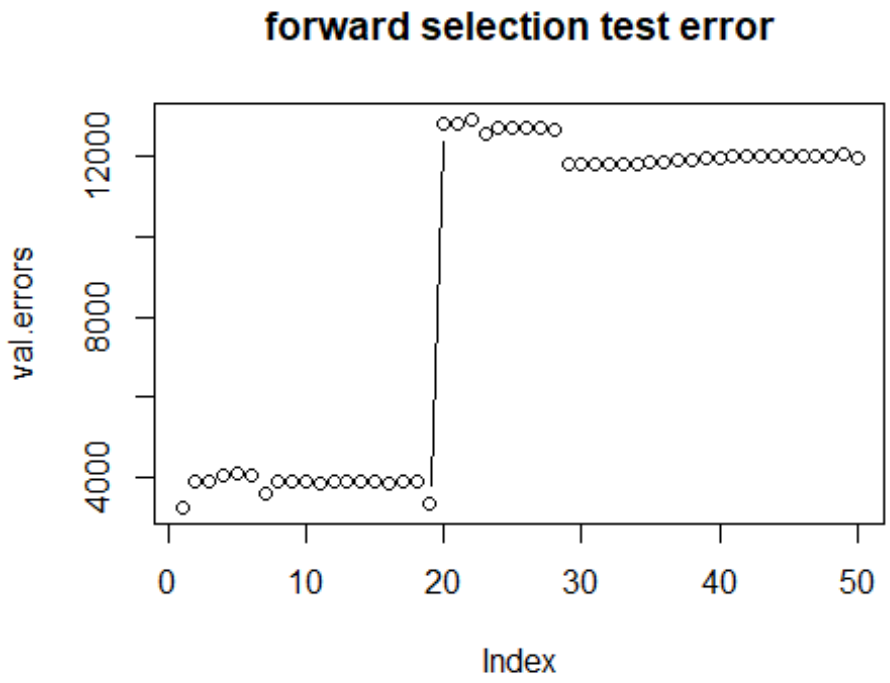

```
plot(summary_$cp, xlab="Number of Variables ", ylab="Cp", type="l")
varB.cp <- which.min(summary_$cp)
points(varB.cp,summary_$cp[varB.cp], col="red",cex=2,pch=20)
```



```
message("number of variables for best adjusted R-squared = ", varB.adjr2)

## number of variables for best adjusted R-squared = 18

message("number of variables for best BIC = ", varB.bic)

## number of variables for best BIC = 7

message("number of variables for best Cp = ", varB.cp)

## number of variables for best Cp = 10
```

calculate test MSE with the best forward model

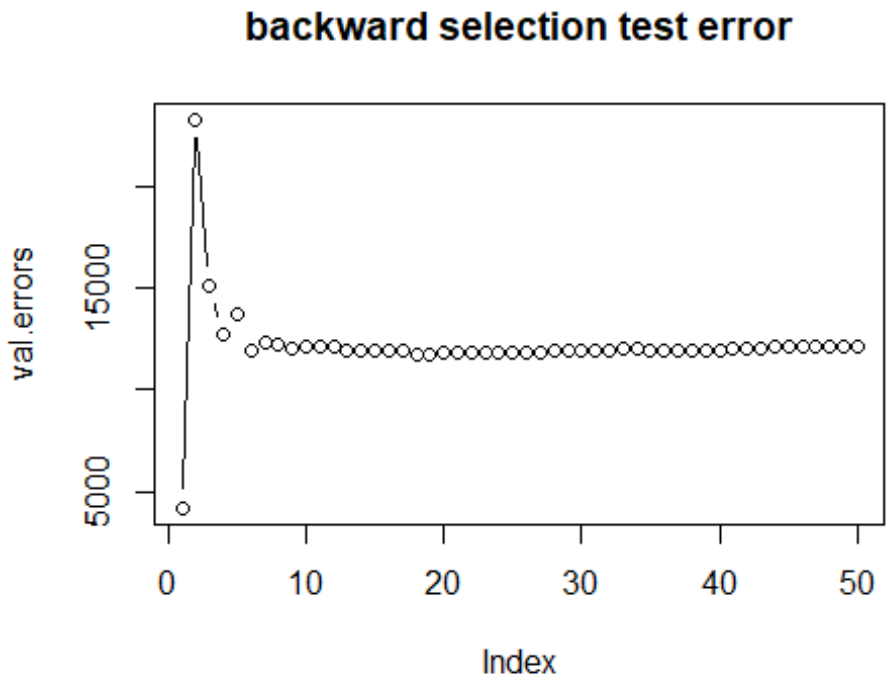
```
test_mat <- model.matrix(Close30 ~ ., data = test_set)
coef = coef(selection_forward, varF.adjr2)
pred = test_mat[,names(coef)]%*%coef
forward_errors.adjr2 = mean((y_test-pred)^2)
coef = coef(selection_forward, varF.bic)
pred = test_mat[,names(coef)]%*%coef
forward_errors.bic = mean((y_test-pred)^2)
coef = coef(selection_forward, varF.cp)
pred = test_mat[,names(coef)]%*%coef
forward_errors.cp = mean((y_test-pred)^2)
message("test_set MSE with best adjusted R-squared = ", forward_errors.adjr2)

## test_set MSE with best adjusted R-squared = 12843.515687094

message("test_set MSE with best BIC = ", forward_errors.bic)

## test_set MSE with best BIC = 3914.04291247753

message("test_set MSE with best Cp = ", forward_errors.cp)

## test_set MSE with best Cp = 3914.09462818934

val.errors=rep(NA,50)
for (i in 1:50){
 coefi = coef(selection_forward, id=i)
 pred = test_mat[,names(coefi)]%*%coefi
 # MSE
 val.errors[i] = mean((y_test-pred)^2)
}
plot(val.errors ,type='b', main="forward selection test error")
```



calculate test MSE with the best backward model

```
test_mat <- model.matrix(Close30 ~ ., data = test_set)
coef = coef(selection_backward, varB.adjr2)
pred = test_mat[,names(coef)]%*%coef
backward_errors.adjr2 = mean((y_test-pred)^2)
coef = coef(selection_backward, varB.bic)
pred = test_mat[,names(coef)]%*%coef
backward_errors.bic = mean((y_test-pred)^2)
coef = coef(selection_backward, varB.cp)
pred = test_mat[,names(coef)]%*%coef
backward_errors.cp = mean((y_test-pred)^2)
message("test_set MSE with best adjusted R-squared = ",
backward_errors.adjr2)

## test_set MSE with best adjusted R-squared = 11746.5286164661

message("test_set MSE with best BIC = ", backward_errors.bic)

## test_set MSE with best BIC = 12288.8326369765

message("test_set MSE with best Cp = ", backward_errors.cp)

## test_set MSE with best Cp = 12083.813122192

val.errors=rep(NA,50)
for (i in 1:50){
 coefi = coef(selection_backward, id=i)
```

```
 pred = test_mat[,names(coefi)]%*%coefi
 # MSE
 val.errors[i] = mean((y_test-pred)^2)
}
plot(val.errors ,type='b', main="backward selection test error")
```



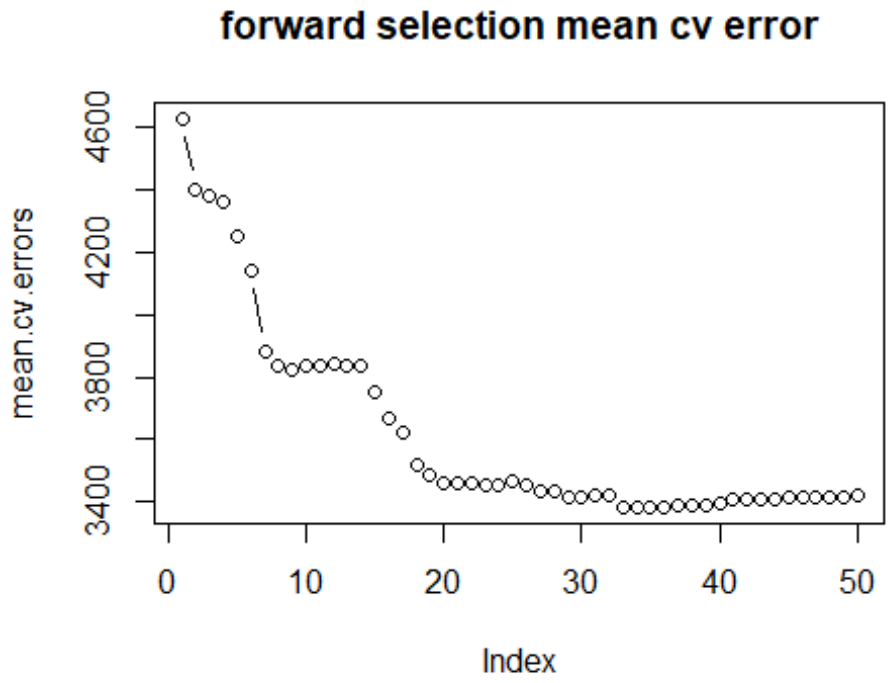
calculate mean cv errors(10-fold)

```
k=10
set.seed(1)
folds = sample(1:k, nrow(train_set), replace=TRUE)
cv.errors=matrix(NA, k, 50, dimnames=list(NULL, paste(1:50)))

predict.regsubsets = function (object, newdata, id ,...){
 form=as.formula(object$call [[2]])
 mat=model.matrix(form,newdata)
 coefi=coef(object ,id=id)
 xvars=names(coefi)
 mat[,xvars]%*%coefi
 }

for(j in 1:k){
 best.fit = regsubsets(Close30~., data=train_set[folds!=j,], nvmax=50, method
="forward")
 for(i in 1:50){
 pred = predict(best.fit, train_set[folds==j,], id=i)
```

```
 cv.errors[j, i] = mean((train_set$Close30[folds==j]-pred)^2)
 }
}
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 10 linear dependencies found

## Reordering variables and trying again:

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 10 linear dependencies found

## Reordering variables and trying again:

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 10 linear dependencies found

## Reordering variables and trying again:

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 10 linear dependencies found

## Reordering variables and trying again:

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 10 linear dependencies found

## Reordering variables and trying again:

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 10 linear dependencies found

## Reordering variables and trying again:

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 10 linear dependencies found

## Reordering variables and trying again:

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 10 linear dependencies found

## Reordering variables and trying again:
```

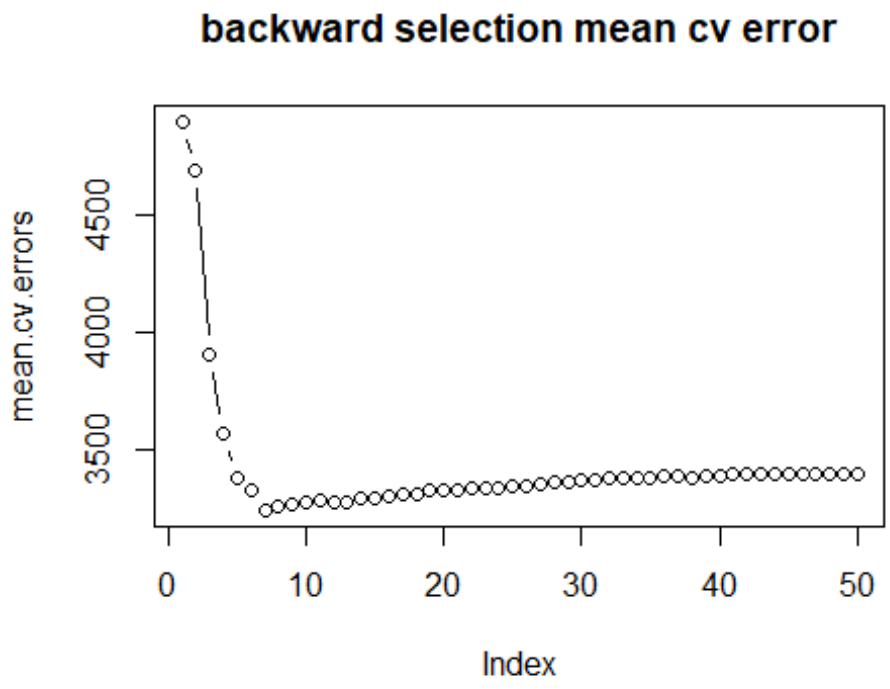
```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 10 linear dependencies found

## Reordering variables and trying again:

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 10 linear dependencies found

## Reordering variables and trying again:

mean.cv.errors = apply(cv.errors, 2, mean)
plot(mean.cv.errors ,type='b', main="forward selection mean cv error")
```




```
for(j in 1:k){
 best.fit = regsubsets(Close30~., data=train_set[folds!=j,], nvmax=50, method
="backward")
 for(i in 1:50){
 pred = predict(best.fit, train_set[folds==j,], id=i)
 cv.errors[j, i] = mean((train_set$Close30[folds==j]-pred)^2)
 }
}

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 10 linear dependencies found
```

```
## Reordering variables and trying again:

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 10 linear dependencies found

## Reordering variables and trying again:

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 10 linear dependencies found

## Reordering variables and trying again:

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 10 linear dependencies found

## Reordering variables and trying again:

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 10 linear dependencies found

## Reordering variables and trying again:

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 10 linear dependencies found

## Reordering variables and trying again:

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 10 linear dependencies found

## Reordering variables and trying again:

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 10 linear dependencies found

## Reordering variables and trying again:

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 10 linear dependencies found

## Reordering variables and trying again:

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 10 linear dependencies found
```

```
## Reordering variables and trying again:

mean.cv.errors = apply(cv.errors, 2, mean)
plot(mean.cv.errors ,type='b', main="backward selection mean cv error")
```




```
message("backward selection mean cv error of 7 variables = ",
mean.cv.errors[7])

## backward selection mean cv error of 7 variables = 3246.21484113369
```

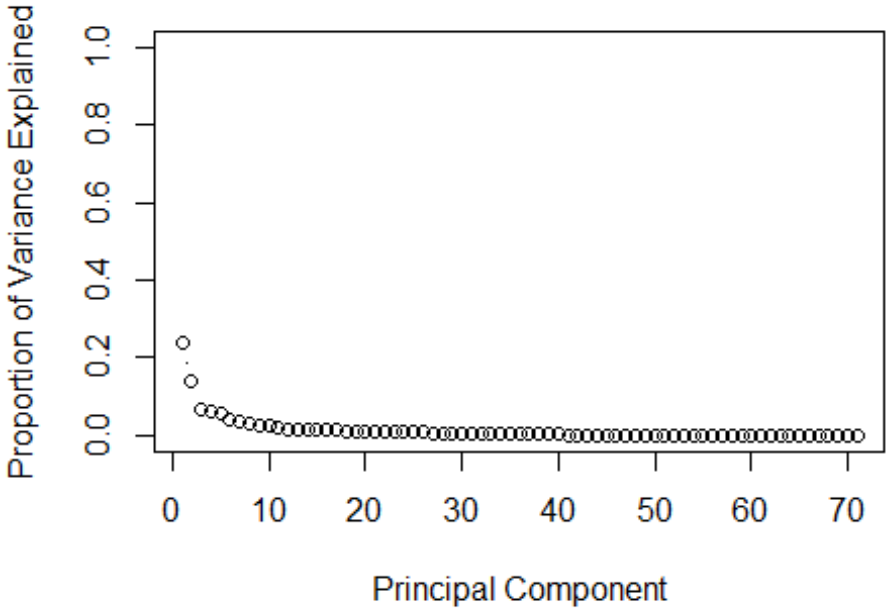
final selected model

```
coefs <- coef(selection_backward, 7)
names_ <- names(coefs)
names_ <- names_[!names_ %in% "(Intercept)"]
response <- as.character(as.formula(selection_forward$call[[2]])[[2]])
form <- as.formula(paste(response, paste(names_, collapse = " + "), sep = " ~
"))
model_best_forward <- glm(form, data = train_set)
summary(model_best_forward)

##
## Call:
## glm(formula = form, data = train_set)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -229.386   -29.415     7.857    37.719   171.703
```

```
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.686e+00  1.795e+01  -0.205    0.837
## ROC_15       3.078e+00  7.646e-01   4.025 5.99e-05 ***
## ROC_20       3.174e+00  6.853e-01   4.631 3.95e-06 ***
## EMA_50       1.035e+00  5.368e-03 192.790  < 2e-16 ***
## DTB3         1.083e+03  6.083e+01  17.797  < 2e-16 ***
## DTB6        -8.316e+02  4.176e+01 -19.914  < 2e-16 ***
## DGS10       -5.447e+01  6.486e+00  -8.399  < 2e-16 ***
## DAAA         3.177e+01  6.361e+00   4.994 6.61e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 3207.334)
##
##     Null deviance: 175952061  on 1486  degrees of freedom
## Residual deviance:   4743647  on 1479  degrees of freedom
## AIC: 16235
##
## Number of Fisher Scoring iterations: 2
```

use principal component to view variance explained

```
pc <- prcomp(x_train, scale = T, center = T)
var <- pc$sdev^2
pve <- var / sum(var)
plot(pve, xlab = "Principal Component", ylab = "Proportion of Variance
Explained", ylim = c(0, 1), type = "b")
```




```
plot(cumsum(pve), xlab = "Principal Component", ylab = "Cumulative Proportion
of Variance Explained", ylim = c(0, 1), type = "b")
abline(h=0.8, col="blue", lty=5.5 )
```