Apple Stock Data Analysis & Prediction Models

Rena Haswah, Christopher Hong, Anna Kong

About the Data

	AAPL.Open	AAPL.High	AAPL.Low	AAPL.Close	AAPL.Volume	AAPL.Adjusted
1980-12-12	0.513393	0.515625	0.513393	0.513393	117258400	0.407747
1980-12-15	0.488839	0.488839	0.486607	0.486607	43971200	0.386473
1980-12-16	0.453125	0.453125	0.450893	0.450893	26432000	0.358108
1980-12-17	0.462054	0.464286	0.462054	0.462054	21610400	0.366972
1980-12-18	0.475446	0.477679	0.475446	0.475446	18362400	0.377609
1980-12-19	0.504464	0.506696	0.504464	0.504464	12157600	0.400656

	AAPL.Open	AAPL.High	AAPL.Low	AAPL.Close	AAPL.Volume	AAPL.Adjusted
2019-10-25	243.16	246.73	242.88	246.58	18369300	245.8419
2019-10-28	247.42	249.25	246.72	249.05	24143200	248.3045
2019-10-29	248.97	249.75	242.57	243.29	35709900	242.5618
2019-10-30	244.76	245.30	241.21	243.26	31130500	242.5318
2019-10-31	247.24	249.17	237.26	248.76	34790500	248.0154
2019-11-01	249.54	255.93	249.16	255.82	37781300	255.0543

Time Series & Problems

Interpretation of Interaction terms

Last time talked about transformations

Which variables are important

Dealing with model validation for time series.

Predictions using past 5 years. How much time is relevant?

A Historical Analysis

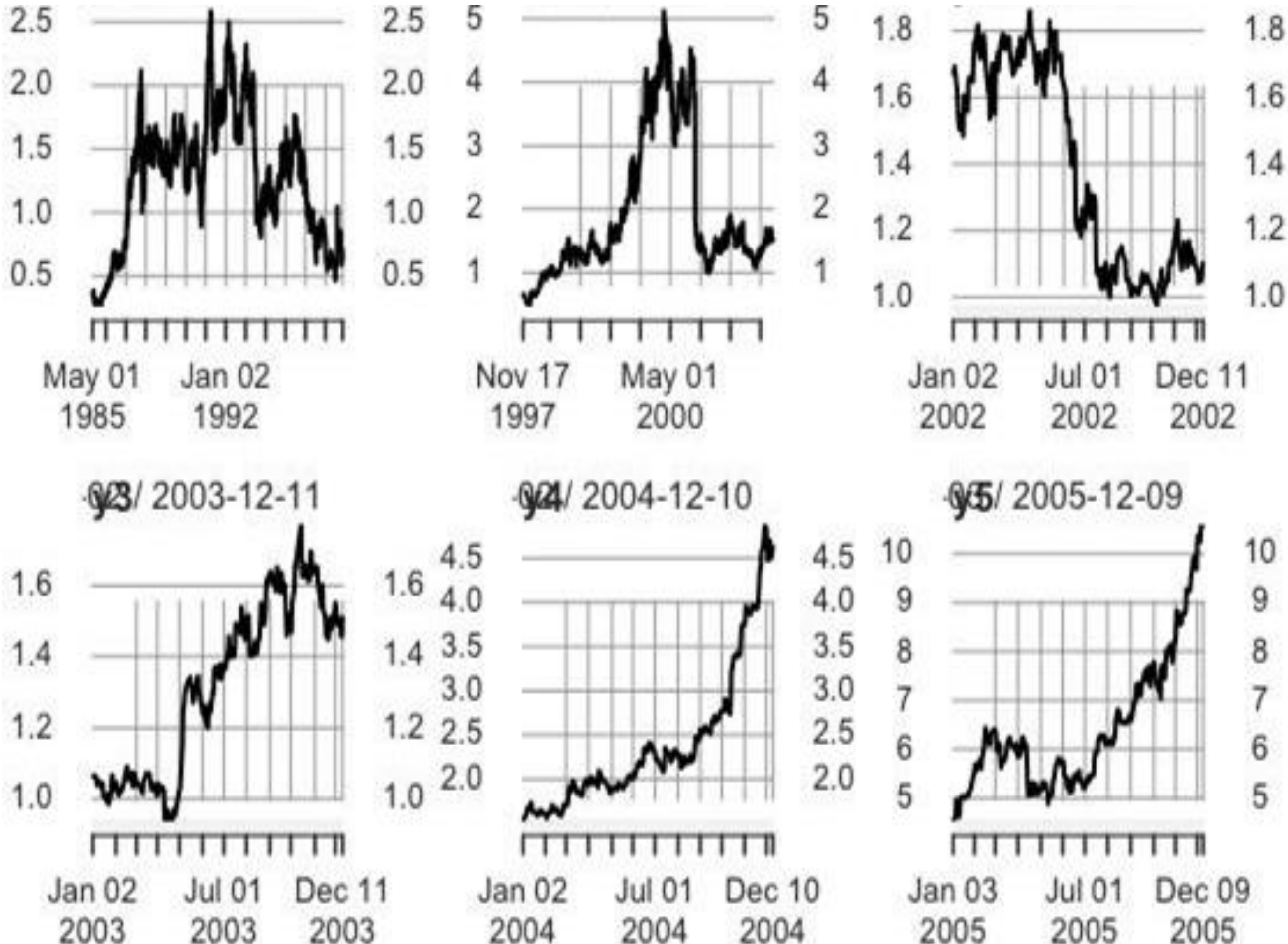
December 12, 1980: Apple goes public.

May 5 1985: Steve Jobs leaves the company

September 16, 1997: Jobs becomes CEO

April 28, 2003: iTunes launch

August 2004: Jobs discloses health problems



Response & Features

	AAPL.Close
2014-09-30	100.75
2014-10-01	99.18
2014-10-02	99.90
2014-10-03	99.62
2014-10-06	99.62
2014-10-07	98.75
2014-10-08	100.80
2014-10-09	101.02
2014-10-10	100.73
2014-10-13	99.81
2014-10-14	98.75
2014-10-15	97.54
2014-10-16	96.26
2014-10-17	97.67
2014-10-20	99.76
2014-10-21	102.47
2014-10-22	102.99
2014-10-23	104.83
2014-10-24	105.22
2014-10-27	105.11
2014-10-28	106.74
2014-10-29	107.34
2014-10-30	106.98

Coefficients:

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	0.04969	0.10423	0.477	0.6337	
AAPL.Open	-0.54977	0.02667	-20.616	<2e-16	***
AAPL.High	0.77782	0.02539	30.636	<2e-16	***
AAPL.Low	0.77185	0.02196	35.148	<2e-16	***

2019-10-08	224.40
2019-10-09	227.03
2019-10-10	230.09
2019-10-11	236.21
2019-10-14	235.87
2019-10-15	235.32
2019-10-16	234.37
2019-10-17	235.28
2019-10-18	236.41
2019-10-21	240.51
2019-10-22	239.96
2019-10-23	243.18
2019-10-24	243.58
2019-10-25	246.58
2019-10-28	249.05
2019-10-29	243.29
2019-10-30	243.26
2019-10-31	248.76
2019-11-01	255.82
2019-11-04	257.50
2019-11-05	257.13
2019-11-06	257.24
2019-11-07	259.43
2019-11-08	260.14

Predicting Apple Stock Close Price (Chris)

Feature Selection

Model Selection

Conclusion

Future work

Feature Engineering

Original Features of Apple Stock:

Open, High, Low, ~~Volume, Adjusted Close~~

Response: Close

Create 10 days lagging:

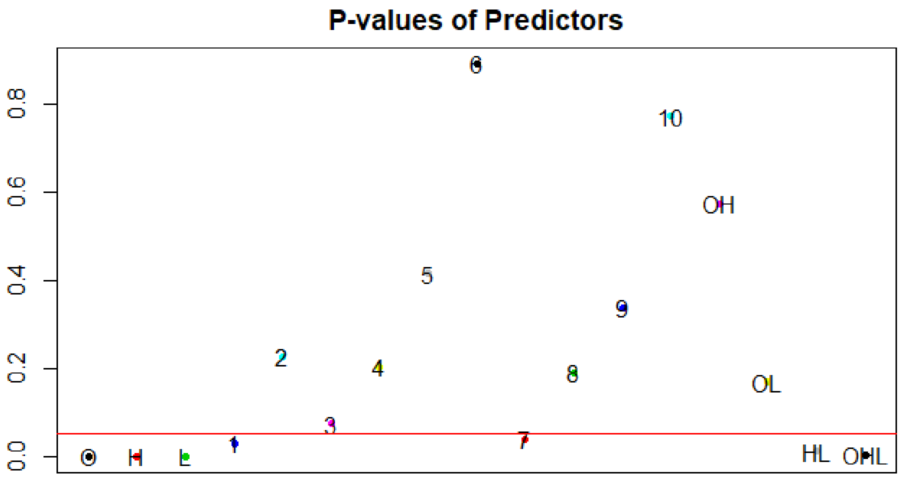
Lag.1, Lag.2, Lag.3, Lag.4, Lag.5, Lag.6, Lag.7, Lag.8, Lag.9, Lag.10

Create possible interaction terms:

Open*High, Open*Low, High*Low, Open*High*Low

Feature Selection

T-Test (full model):



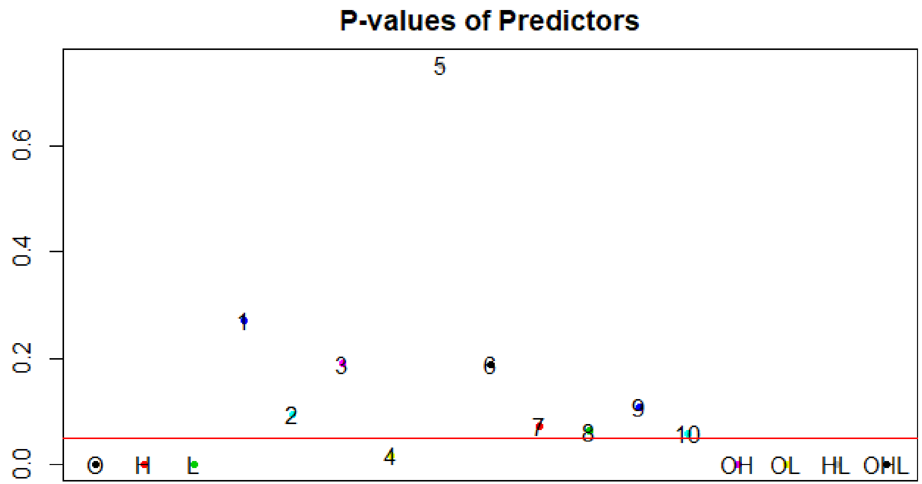
1.353e+05

< 2.2e-16

Conclusion: At least one predictor is

Feature Selection Cont’

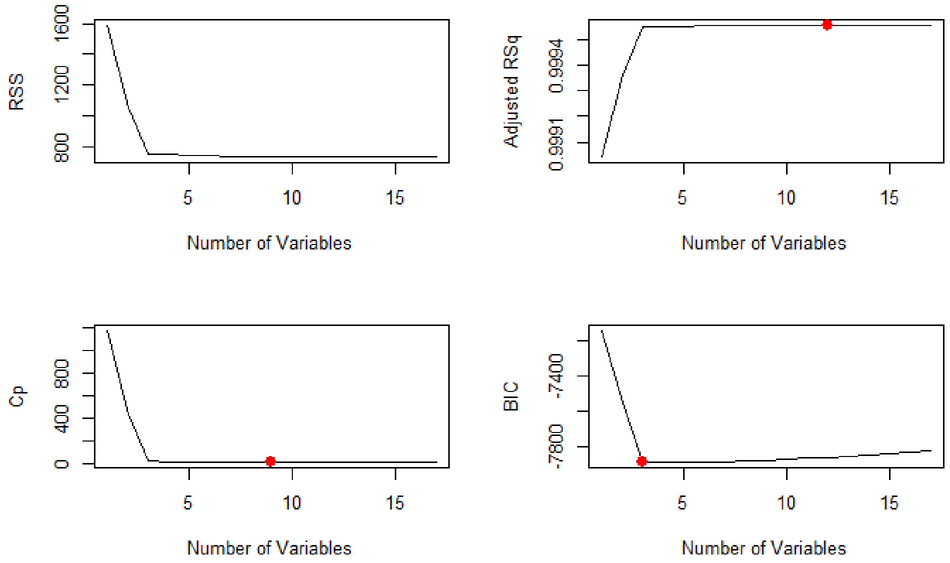
T-Test (simple model):



Significant predictors (P-value <= 0.05): AAPL.Open, AAPL.High, AAPL.Low, Lag.4, Open.High, Open.Low, High.Low, Open.High.Low

Feature Selection Cont’

Best subset selection:



Predictors selected: **AAPL.Low, AAPL.High, AAPL.Open**, Lag.7, Lag.1, High.Low, Open.High.Low

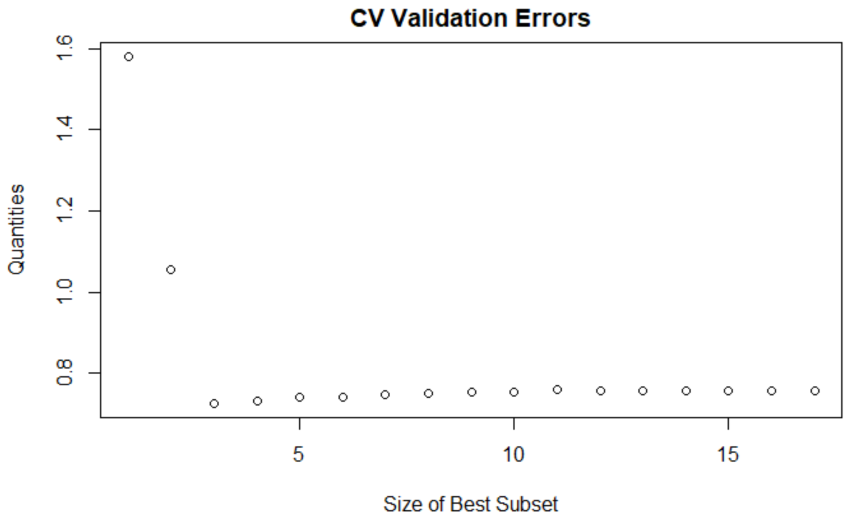
Forward selection: **AAPL.Low, AAPL.High, AAPL.Open**, Lag.7, Lag.1, Lag.3, Lag.4

Backward selection: **AAPL.Low, AAPL.High, AAPL.Open,** Open.High.Low, High.Low, Lag.7, Lag.1

Feature Selection Cont’

Validation set approach:

CV Validation set



Validation set approach: **AAPL.Open, AAPL.High, AAPL.Low**, Lag.7, High.Low, Open.High.Low

CV validation set approach: **AAPL.Open, AAPL.High, AAPL.Low**

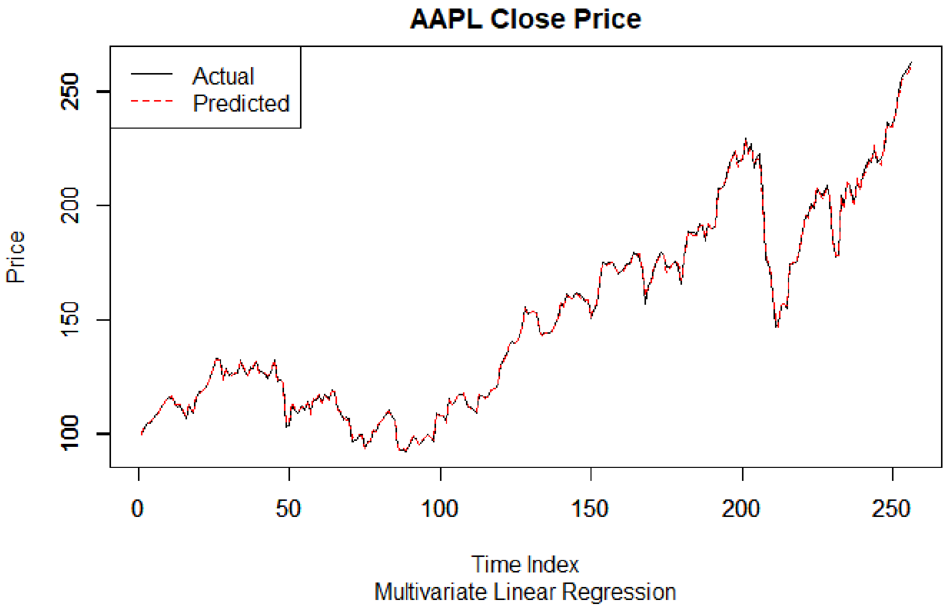
Multivariate Linear Regression

```
fmla_best <- as.formula("AAPL.Close ~ AAPL.Open + AAPL.High + AAPL.Low")
model_lm_best <- lm(fmla_best, data = aapl_df, subset = train)
```

	Open	High	
P-values:	<2e-16	<2e-16	<2e-16

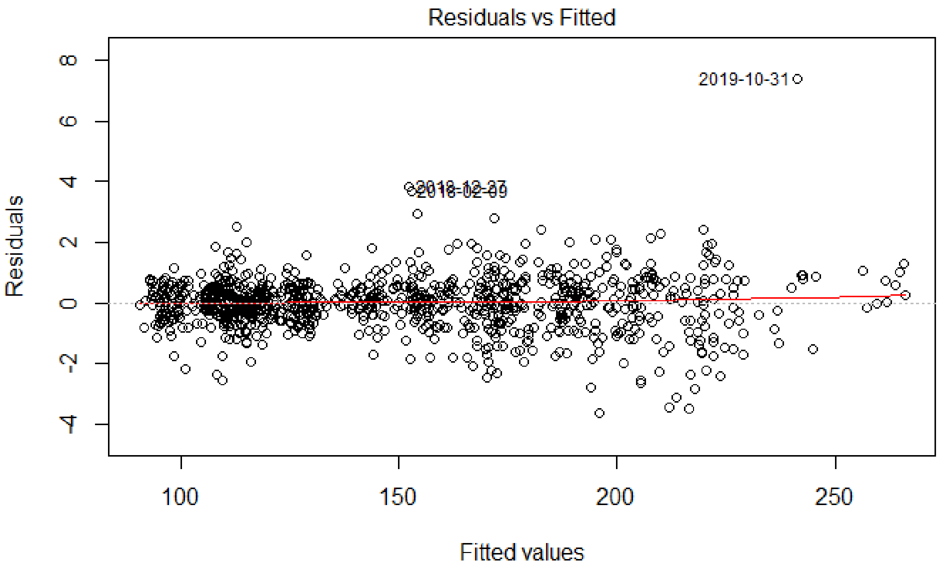
Adjusted R-squared: 0.9995

Test RMSE: 0.83377



Multivariate Linear Regression Cont’

Check for non-linearity relationship:



Check for multicollinearity: 1517.2430 1299.9873 902.1373 (highly correlated)

10 * 9-fold CV Multivariate Linear Regression

```
# Fit a 10 * 9-fold CV linear model
model_lm_best_cv <- train(
    fmla_best,
    aapl_df[train, ],
    method = "lm",
    trControl = trainControl(
        method = "repeatedcv",
        number = 9,
        repeats = 10,
        verboseIter = F
        )
    )
```

Test RMSE: 0.83377 (identical to MLR)

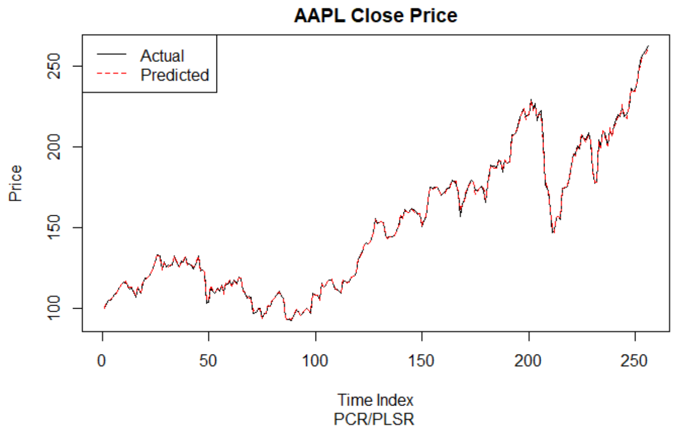
10 * 9-fold CV Multivariate Linear Regression w PCA

```r
# Fit a 10 * 9-fold CV linear model with PCA preprocess
model_lm_pca <- train(
    fmla_best,
    aapl_df[train, ],
    method = "lm",
    preProcess = c("center", "scale", "pca"),
    trControl = trainControl(
        method = "repeatedcv",
        number = 9,
        repeats = 10,
        verboseIter = F
        )
)
```

Test RMSE: 1.258917 (worse than MLR)

PCR & PLSR

```r
model_pcr <- pcr(fmla_best,
                 data = aapl_df,
                 subset = train,
                 scale = T,
                 validation = "CV")
```



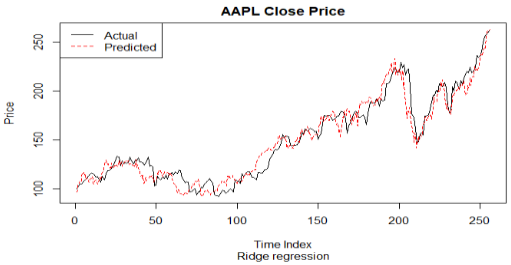

```r
model_plsr <- plsr(fmla_best,
                   data = aapl_df,
                   subset = train,
                   scale = T,
                   validation = "CV")
```

Test RMSE: 0.83377 (identical to MLR)

Ridge & Lasso Regression

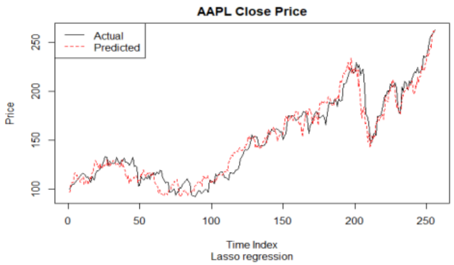
```
Ridge model test RMSE: 1.199273

19 x 1 sparse Matrix of class "dgCMatrix"
                            1
(Intercept)      4.240078e+01
(Intercept)      .
AAPL.Open       -1.167801e-01
AAPL.High        2.496978e-01
AAPL.Low         4.930493e-02
Lag.1            1.497672e-02
Lag.2           -2.221193e-03
Lag.3           -1.170855e-04
Lag.4            5.866054e-03
Lag.5            9.580030e-03
Lag.6            3.884985e-03
Lag.7            6.938748e-03
Lag.8            1.338014e-03
Lag.9           -3.504066e-04
Lag.10          -2.337754e-03
Open.High       -6.095482e-04
Open.Low         2.009370e-03
High.Low         3.602157e-03
Open.High.Low -9.685770e-06
```



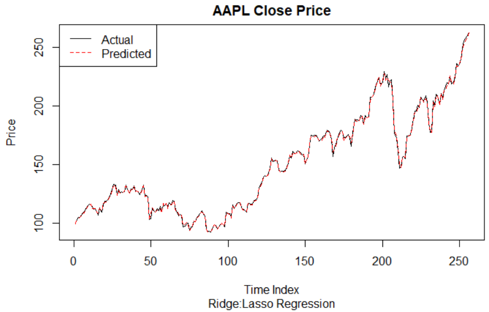

```
Lasso model test RMSE: 1.313807

19 x 1 sparse Matrix of class "dgCMatrix"
                            1
(Intercept)      4.773127e+01
(Intercept)      .
AAPL.Open        .
AAPL.High        .
AAPL.Low         8.183082e-02
Lag.1            .
Lag.2            .
Lag.3            .
Lag.4            .
Lag.5            .
Lag.6            .
Lag.7            .
Lag.8            .
Lag.9            .
Lag.10           .
Open.High        1.281371e-03
Open.Low        -8.597724e-05
High.Low         4.403576e-03
Open.High.Low -1.092844e-05
```




```
Ridge:Lasso model test RMSE: 1.337053
```

```r
# Choose the best alpha between ridge and lasso
model_glmnet <- train(
    AAPL.Close ~ .,
    aapl_df[train,],
    method = "glmnet",
    tuneGrid = expand.grid(
        alpha = 0:1,
        lambda = 10^seq(10, -2, length = 100)
    ),
    trControl = trainControl(
        method = "cv",
        number = 9,
        verboseIter = F
    )
)
```



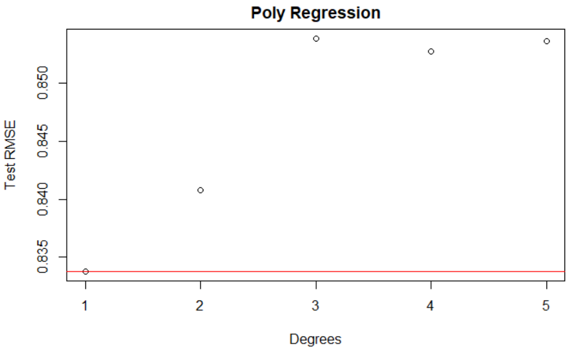
Ridge model is the best of the three, but still performs much worse than MLR.

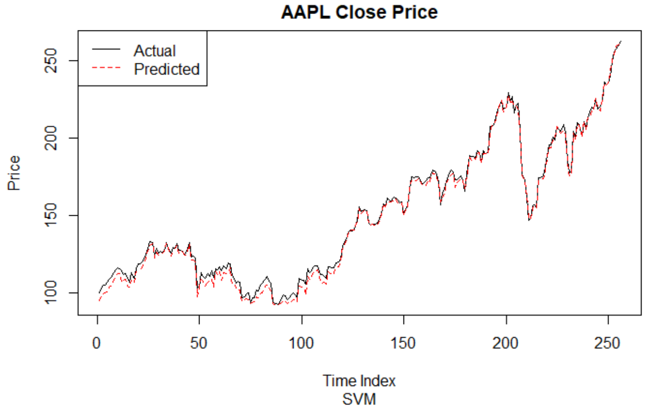
Poly Regression / SVM

```r
model_poly <- lm(AAPL.Close ~ poly(AAPL.Low, 1)
                 + poly(AAPL.High, 1)
                 + poly(AAPL.Open, 1),
                 data = aapl_df[train, ])
```

```r
model_svm <- svm(fmla_best, data = aapl_df[train, ])
```

```
SVM test RMSE: 1.556552
```





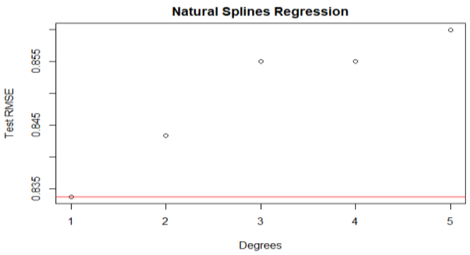
Performance of poly regression model of degree of 1 is identical to MLR.

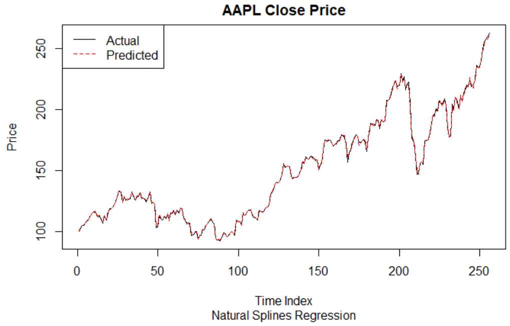
SVM is defeated by MLR.

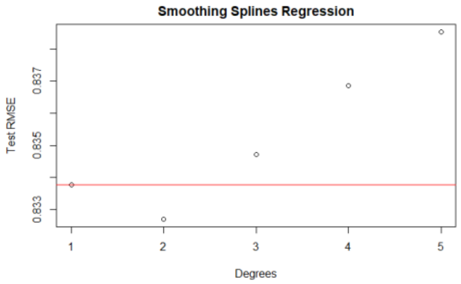
GAM Natural Splines & Smoothing

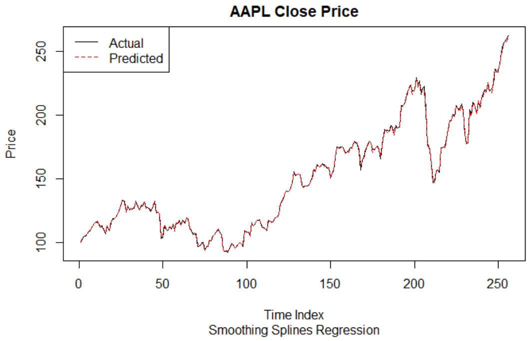
```r
model_ns <- lm(AAPL.Close ~ ns(AAPL.Open, df = 1)
          + ns(AAPL.High, df = 1)
          + ns(AAPL.Low, df = 1),
          data = aapl_df[train, ])
```

```r
model_s <- gam(AAPL.Close ~ s(AAPL.Open, df = 2)
          + s(AAPL.High, df = 2)
          + s(AAPL.Low, df = 2),
          data = aapl_df[train, ])
```





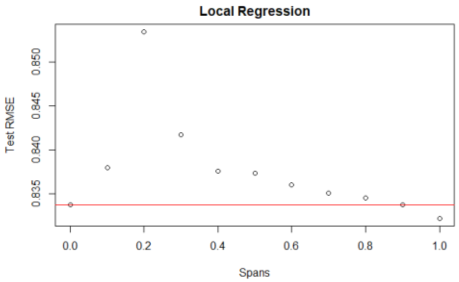


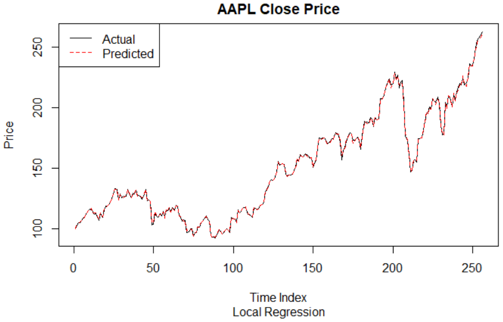


When degrees = 2, smoothing splines regression beats natural splines regression and MLR.

GAM Local Regression & KNN

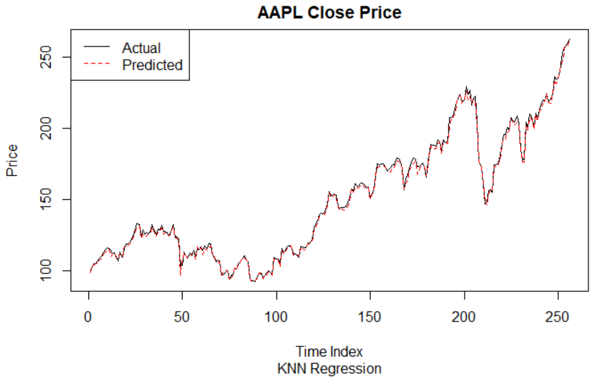
```r
model_lo <- gam(AAPL.Close ~ lo(AAPL.Open, span = 1)
          + lo(AAPL.High, span = 1)
          + lo(AAPL.Low, span = 1),
          data = aapl_df[train, ])
```





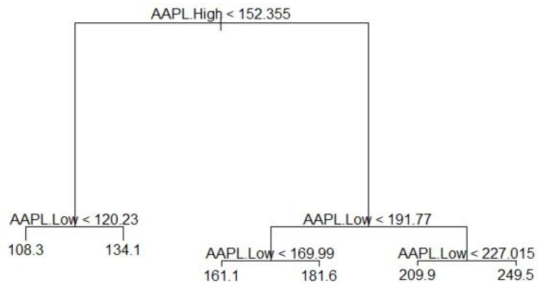

```r
model_knn <- train(
    fmla_best,
    aapl_df[train, ],
    method = "knn",
    preProcess = "pca",
    tuneLenghth = 5,
    trControl = trainControl(
        method = "repeatedcv",
        number = 10,
        repeats = 10,
        verboseIter = F
    )
)
```

KNN model test RMSE: 1.32794



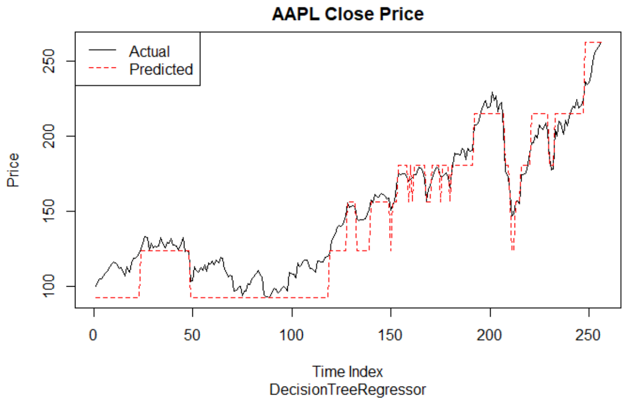
When span = 1, local regression beats all models so far.

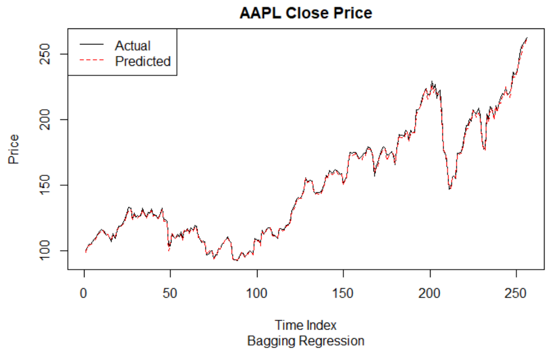
Decision Tree & Bagging




```
model_bagging <- train(
  fmla_best,
  tuneLength = 3,
  data = aapl_df[train, ],
  method = "ranger",
  trControl = trainControl(
    method = "cv",
    number = 10,
    verboseIter = F
  )
)
```

Regression tree model test RMSE: 8.071919





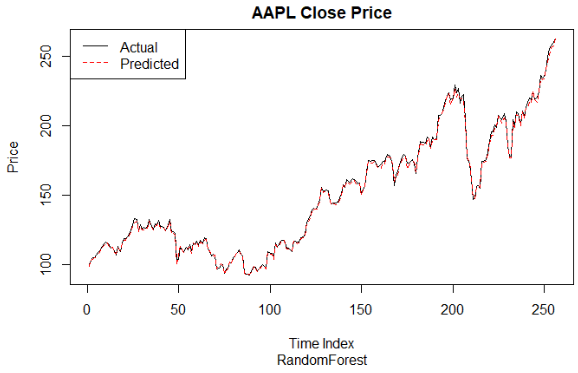
Bagging regression tree model test RMSE: 1.110949

Bagging substantially improves the performance of decision tree, but is still defeated by MLR.

Random Forest & Boosting

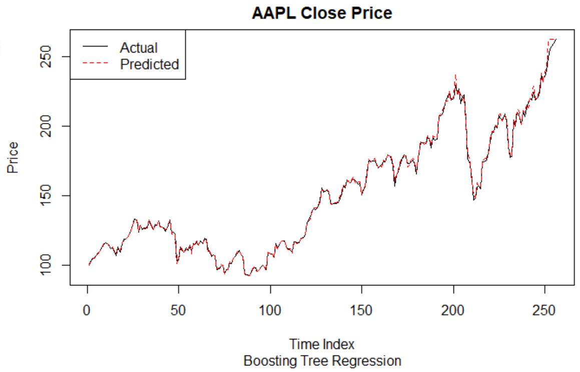
```
model_rf <- train(
  fmla_best,
  tuneLength = i,
  data = aapl_df[train, ],
  method = "ranger",
  trControl = trainControl(
    method = "cv",
    number = 10,
    verboseIter = F
  )
)
```

```
Number of variables used: 2
RandomForest model test RMSE: 1.117912
```



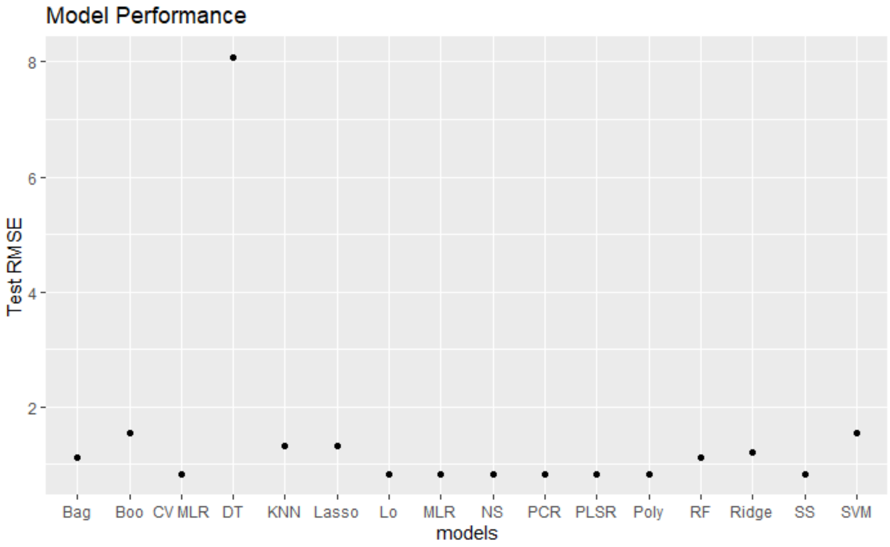

```
model_tree_boost <- gbm(AAPL.Close ~ .,
                        data = aapl_df[train, ],
                        distribution = "gaussian",
                        n.trees = 5000,
                        interaction.depth = 4,
                        shrinkage = 0.2,
                        verbose = F)
```

```
Boosting Tree model test RMSE: 1.541871
```



Both RandomForest and Boosting are defeated by Bagging.

Conclusion



models <chr>	model_rmse <dbl>
Lo	0.8322157
SS	0.8327063
MLR	0.8337700
CV MLR	0.8337700
PCR	0.8337700
PLSR	0.8337700
Poly	0.8337700
NS	0.8337700
Bag	1.1109490
RF	1.1179120
Ridge	1.1992730
Lasso	1.3138070
KNN	1.3279400
Boo	1.5418710
SVM	1.5565520
DT	8.0719190

Since there are negligible improves of Local Regression and Smoothing Splines over Multivariate Linear Regression, it is hard to choose one over another, results will be kept to assess the performance of these models over time.

Future Work

SARIMA

RNN

```
Today's close price:  263.19
Local Regression predicted close price: 262.2029
Error: 0.9871324
Smoothing Splines predicted close price: 262.3672
Error: 0.8228347
Natural Smoothing predicted close price: 262.0859
Error: 1.104081
MLR predicted close price: 262.0859
Error: 1.104081
```

Classifying Stock Direction

Feature selection

Model testing

Results

Future Work

Linear Discriminant Analysis

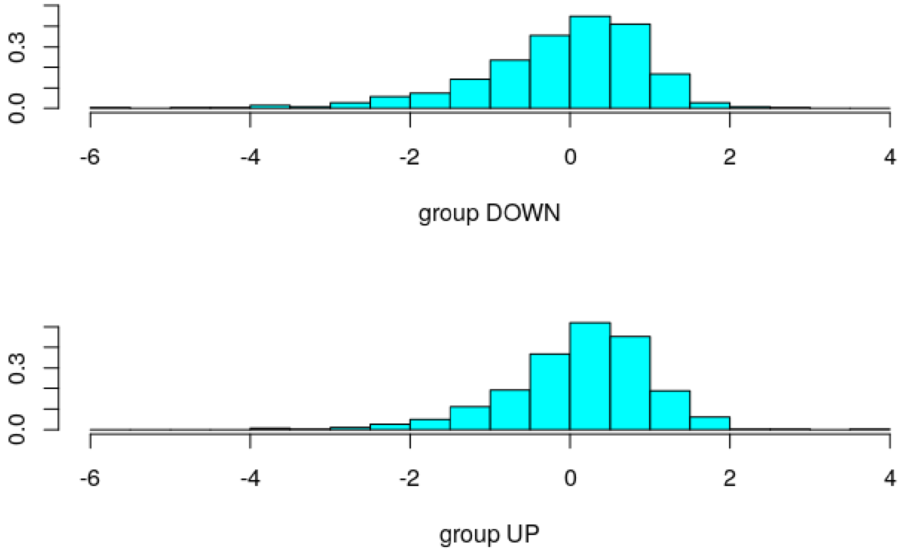
```
Call:
lda(Direction ~ ., data = aapl_df_train)

Prior probabilities of groups:
     DOWN        UP 
0.4775225 0.5224775 

Group means:
     AAPL.Volume        Lag.1        Lag.2        Lag.3        Lag.4        Lag.5        Lag.6         Lag.7        Lag.8         Lag.9       Lag.10
DOWN    38638477 0.0008911106 0.0004894734 0.0003193953 0.0004270201 0.0006860126 0.0011397269 -0.0000880746 0.0016675926 -0.0002318008 0.0007716514
UP      35519389 0.0002522747 0.0007666444 0.0008764654 0.0005628416 0.0011406642 0.0007282096  0.0011584954 0.0005060658  0.0016629528 0.0009788301

Coefficients of linear discriminants:
                      LD1
AAPL.Volume -3.784950e-08
Lag.1       -1.309289e+01
Lag.2       -1.500402e+00
Lag.3        8.299885e+00
Lag.4       -1.665164e+00
Lag.5        6.443851e+00
Lag.6       -1.000261e+01
Lag.7        1.934848e+01
Lag.8       -2.303711e+01
Lag.9        2.712996e+01
Lag.10       1.098986e+00
```

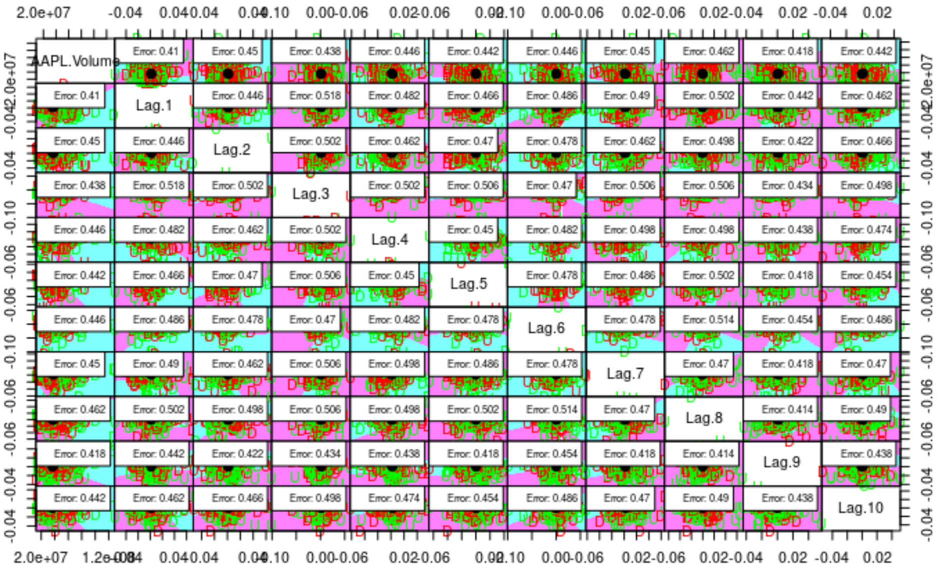
Linear Discriminant Analysis




```
         Actual
Predicted DOWN UP
     DOWN   38 37
     UP     81 93
```

```
[1] 0.5261044
```

Quadratic Discriminant Analysis

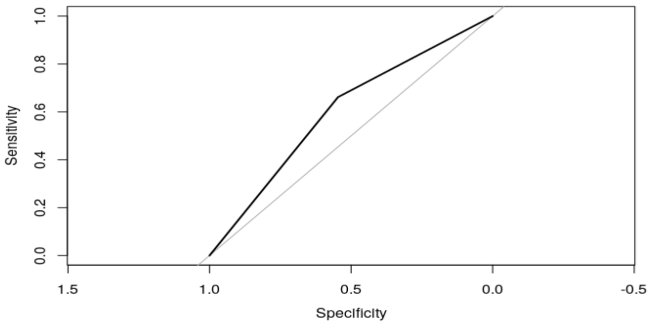



```
         Actual
Predicted DOWN UP
     DOWN   48 43
     UP     71 87
[1] 0.5421687
```

K nearest neighbors

```
getknnerr <-function(n, traindata, testdata, trainresp, testresp) {

  ncount=0
  err=0
  errcount=0
  tablen=0
  knnsamp <- knn(traindata, testdata, trainresp, k=1)
  tablesamp <- table(knnsamp, testresp)
  print(tablesamp)
  err=(tablesamp[1,1]+tablesamp[2,2])/(nrow(testdata))
  cat("Base error at k=1 :",err)
  for(i in 2:n){
    knnsamp2 <- knn(traindata, testdata, trainresp, k=i)
    tablesamp2 <- table(knnsamp2, testresp)
    errcheck=(tablesamp2[1,1]+tablesamp2[2,2])/(nrow(testdata))
    if(errcheck>err){
      ncount=i
      err=errcheck
      tablen=tablesamp2
      }
  }
  cat("\nfinal accuracy score:",err)
  cat("\nachieved at k=",ncount)
  tablen
  return(ncount)
}
```




```
Area under the curve: 0.6039
```

```
knnreturn DOWN UP
     DOWN   65 44
     UP     54 86
[1] 0.6064257
```

Best k=6

Support Vector Machine

Tuning - linear and radial (not shown):

```
tune.out=tune(svm, Direction~., data=aapl_df_train, kernel="linear",ranges=list(cost=c(0.01, 0.1,.5, 1, 10)))
```

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost
 0.01

- best performance: 0.4804455

- Detailed performance results:
   cost     error dispersion
1  0.01 0.4804455 0.04948808
2  0.10 0.4804653 0.04147936
3  0.50 0.4844653 0.04747819
4  1.00 0.4834653 0.04768754
5 10.00 0.4854653 0.04976422
```

Best cost = .01

```
postResample(predict(svmfit1, newdata = X_test), Y_test)
```

```
  Accuracy      Kappa
0.54618474 0.05532986
```

Literature review suggests 55-60% for predicting stock direction with svm

Results

Model	**Accuracy**
LDA	.526
QDA	.542
KNN	.606
SVM	.546

KNN > SVM > QDA > LDA

Future Work

- Take into account outside features - company specific, macroeconomic (GDP, interest rate, etc), and news analysis would possibly be able to make better predictions.

- Take specific events/times into account for data analysis - can we have different predictions in between major events or keynotes, and does that affect our accuracy.