Apple Stock Data Analysis & Prediction Models

CSP571

Rena Haswah
Christopher Hong
Anna Kong

November 27th, 2019

Table of contents

1. Abstract

Apple stock historical data dates back to December 12, 1980 when the company went public. After almost 20 years of little to no activity, Steve Jobs returns to be CEO and the stock started moving. Early 2000's, Apple introduces iTunes and other iProducts, but in the last 5 years the stock has really gone from average to high end. Several statistical methods were used to predict the closing price of the stock. The average closing price of 2013 is $67.52, and so-far in 2019 it has averaged to $201.20. This is definitely a stock that has transformed into a very profitable one, but now we seek to learn from these past five years and to see if these growth trends can lead to any information of price prediction that could determine buying decisions.

2. Overview

Can we use machine learning methods to make predictions on stock price? Given data on how the stock has been behaving what will be the closing price tomorrow? The goal was to determine the closing price of the Apple stock as well as to generally explore the historical data that goes with it. This was accomplished by multiple linear regression. Some classification methods were used for prediction of direction. Regression used important features such as the opening, high, and low, and even engineering some new features, such as lags to help the predictive power. The volume turns out to be very important in explaining the closing price, but ends up being left out of regression models because we are not able to obtain the volume before the stock is closed.

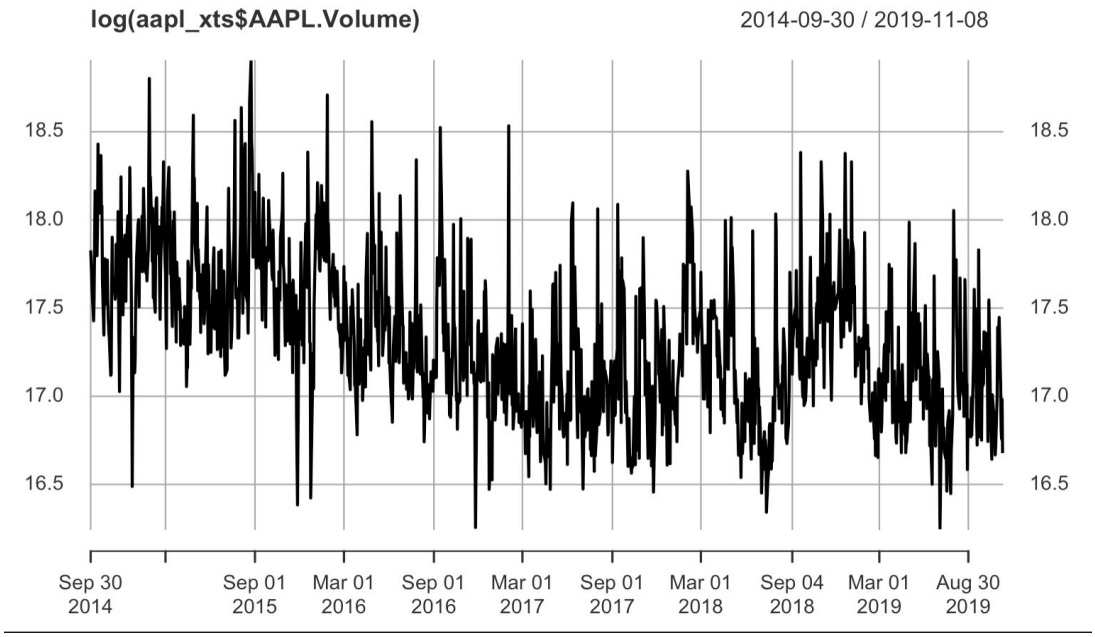
3. Research Summary, Findings and Future Work

After statistical analysis and feature exploration, Open, High, and Low are used as predictors in the base model, multivariate linear regression. Principal component analysis (PCA) preprocessing, principal component regression (PCR) and partial least squares regression (PLSR) were used to combat multicollinearity, but those methods has the same predictive performance as the base model. Ridge regression resulted in Lasso was used for variable selection. Local regression with span of 1, smoothing splines with 2 degrees of freedom and the base model performed best in terms of RMSE. Polynomial regression, natural splines regression, SVM and K-Nearest Neighbors regression performed slightly worse than the best three ones. Decision tree regression has the worst performance, while bagging, random forest and boosting did substantially improve the performance of the tree model. Since there are negligible prediction improvement of local regression and smoothing splines over multivariate linear regression, it is hard to choose the best one among these three models. Future prediction results will be kept to assess the performance of these models over time. We are planning to perform SARIMA and RNN models to predict the close price of Apple stock.

Lagged returns were used to determine the direction of the stock. Classification methods used: LDA, QDA, SVM, KNN. The classifier with the best performance was KNN. There is a lot of room in our models to use more advanced financial features such as price volatility, price momentum, sector volatility and sector momentum or outside features such as news sentiment or macroeconomic trends to increase the accuracy of our predictions.

4. Data Processing

Historical data is available directly in R by sourcing from Yahoo Finance with library Quantmod. There were no real preprocessing challenges because of this. The regression models use data from 2014-09-30 to 2019-11-11. Lags of 10 days were used in feature engineering. Log transformation leads to a better form of the data that we can assume follows a random normal distribution. The log of the volume feature shows a constant mean and variance.



5. Data Analysis

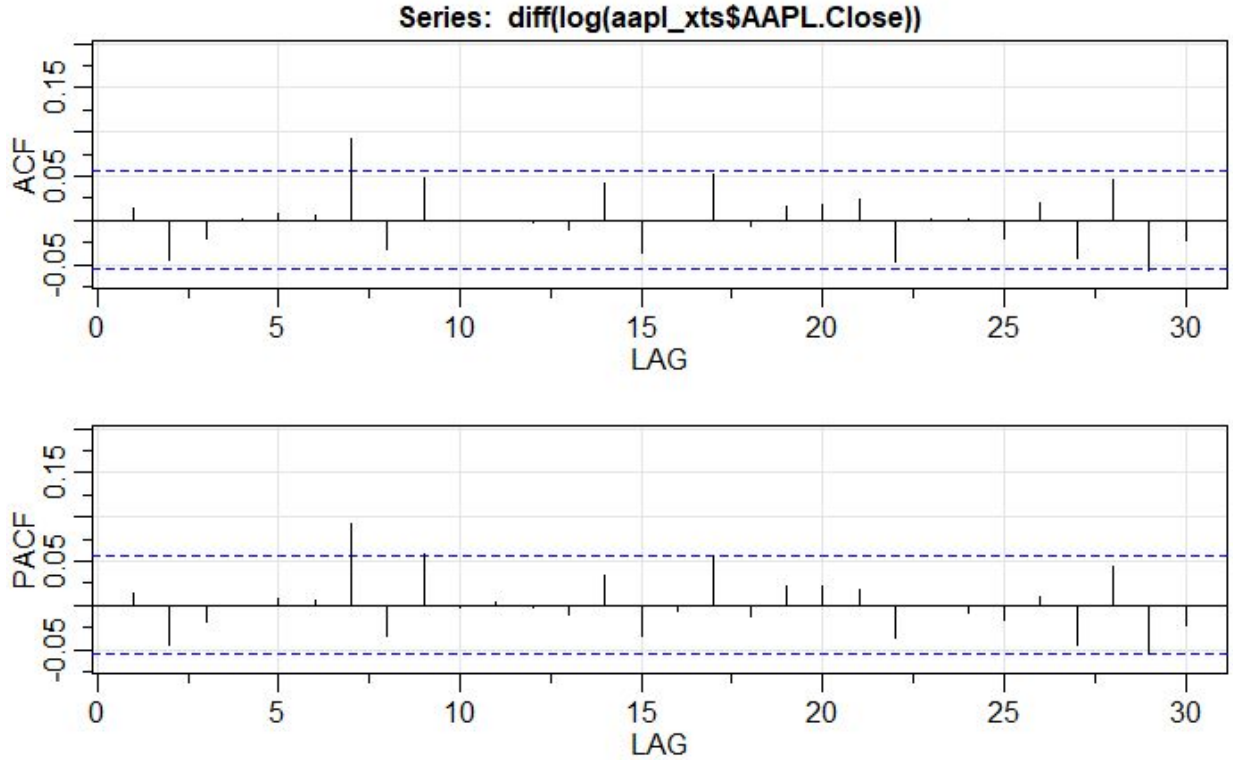
Testing was used to determine the statistical significance of the features. In Rstudio, p-values with three stars have significance level α < 0.001.

```
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 ***
```

6. Regression: Prediction of Close Price

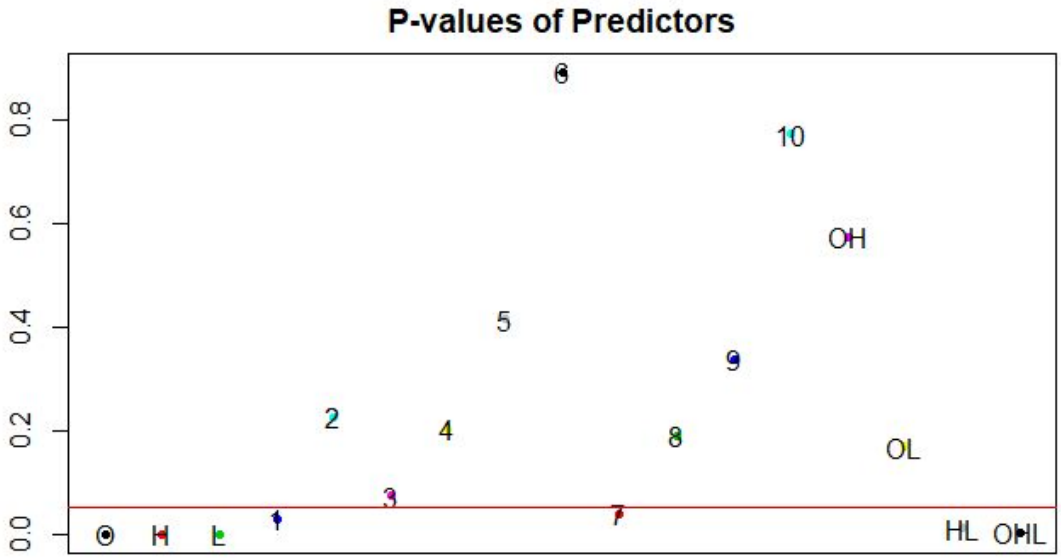
6.1 Feature Engineering

Since we are building a model to predict close price of apple stock and are not able to obtain the volume and the adjusted close price before the stock is closed. Therefore, we excluded the AAPL.Volume and AAPL.Adj.Close from the data set. The autocorrelation graph showed that that data was a random walk model. We suspected that today's close price might partially depends on ones of previous days, so we built features of the previous 10 days, Lag1 through Lag10. We also suspected that there might be synergy between open, high and low price, so we also built interactions of combinations of these features, Open.High, Open.Low, High.Low and Open.High.Low.

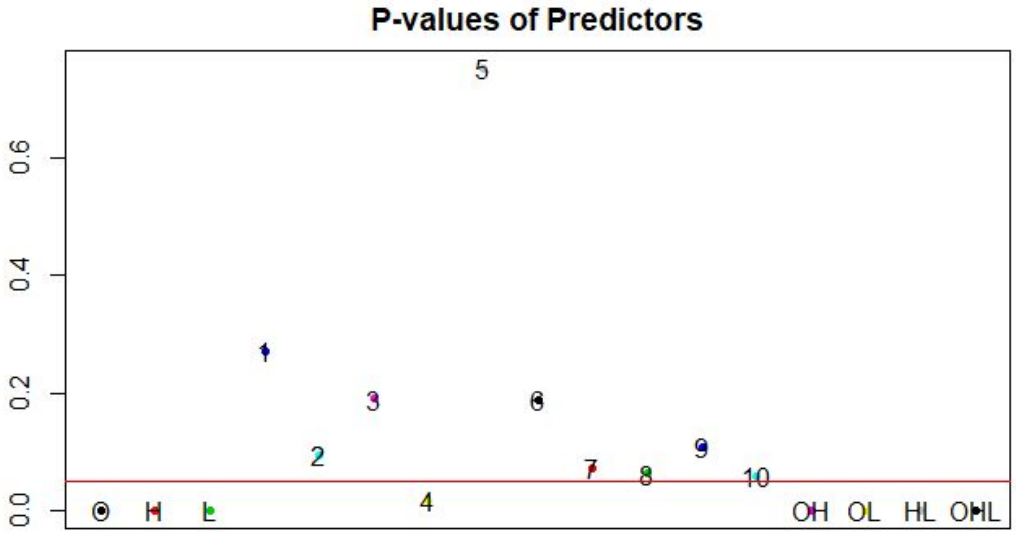


6.2 Feature Selection

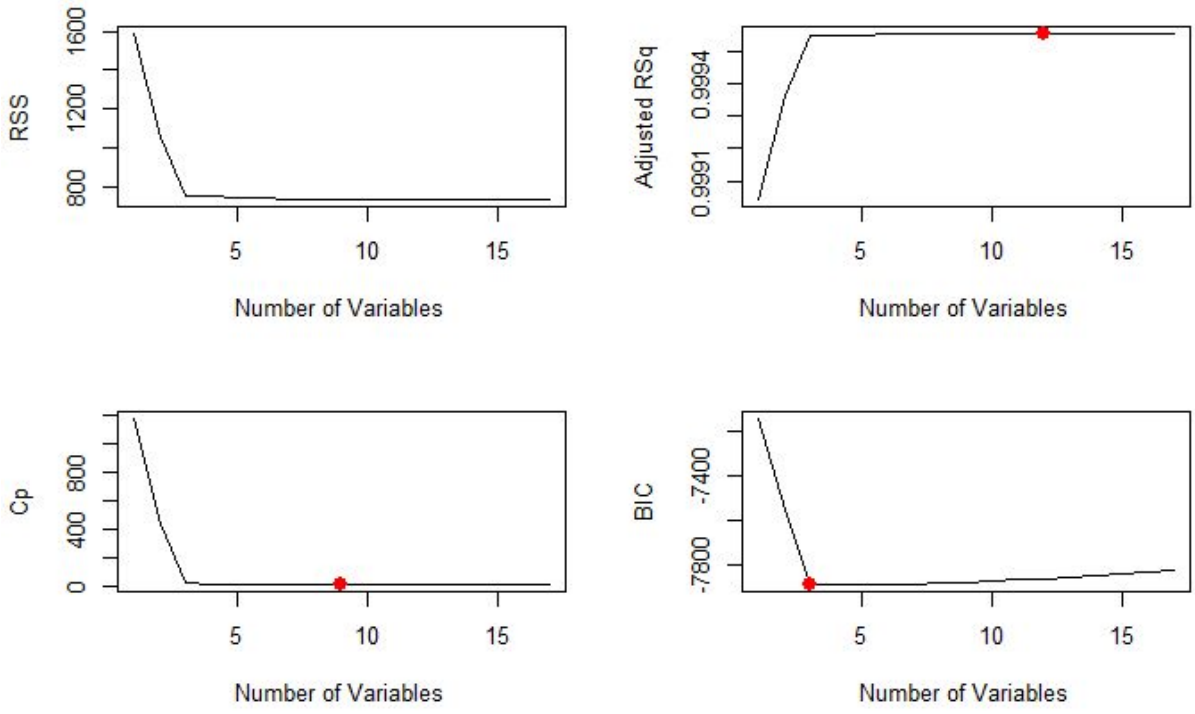
Since we didn't know if there was a relationship between these predictors and the response AAPl.Close, we regressed the response on all the features. The F statistic (1.353e+05) confirmed that there was indeed a relationship between them. The test result showed that Open, High, Low, Lag1, Lag7, High.Low, Open.High.Low were significant based on threshold of 0.05.



In order to find out which feature(s) contributes to AAPL.Close, we did regression test on every single feature. Lag1 and Lag7 were not significant to the response anymore. Instead, Lag4, Open.High and Open.Low became significant while the rest of the significant ones were the same as the ones in the full regression test.

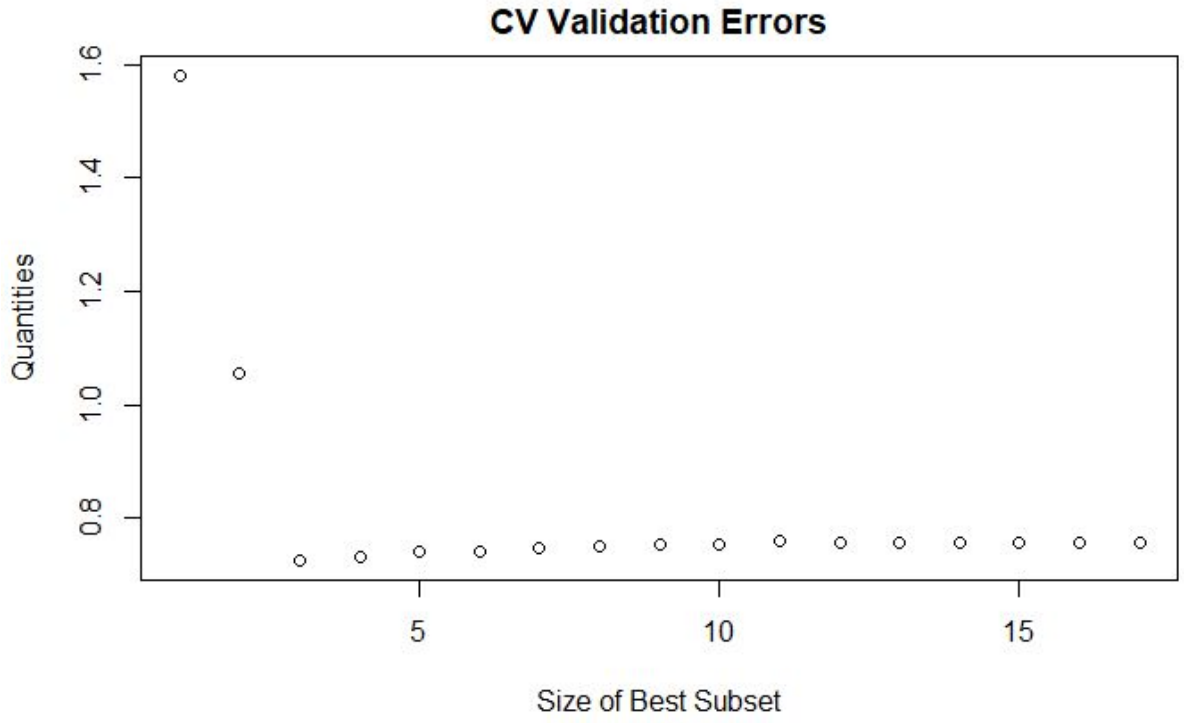


Since the results between the full regression and simple regression tests were inconsistent, we performed the best subset selection. Results of RSS, Adjusted RSq, Cp and BIC showed that features Open, High and Low explain 99.95% of variance, while adding Lag1 and Lag7 helped improve the predictive accuracy a little. We also performed forward and backward selection and their results were consistent with the best subset approach with regard to the first three features. However, there were discrepancies of feature importance besides other than those three features.



To confirm if extra features contributed to the prediction errors, we ran validation set and cross-validation approaches. From below two graphs, we saw that the subset of size three leaded to lowest test errors. Finally, we came to the conclusion that Open, High and Low were the most relevant features.

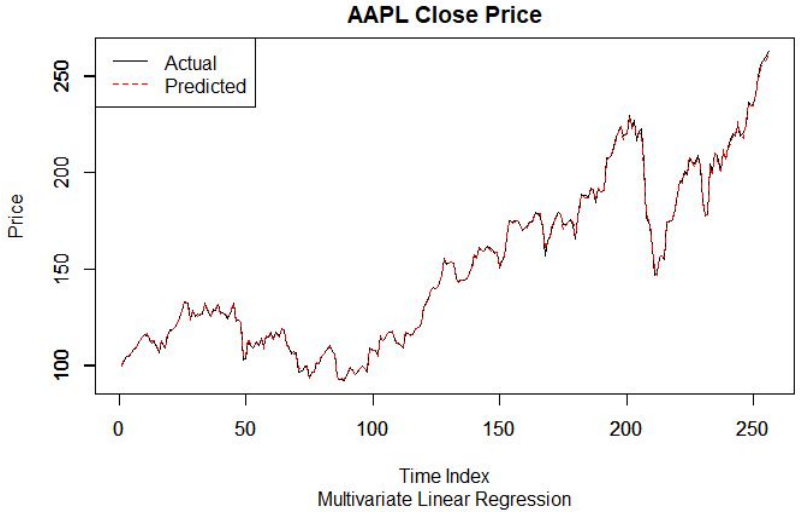

Validation Errors
Quantities
0.8
1.0
1.2
1.4
1.6
5
10
15
Size of Best Subset


CV Validation Errors
Quantities
0.8
1.0
1.2
1.4
1.6
5
10
15
Size of Best Subset

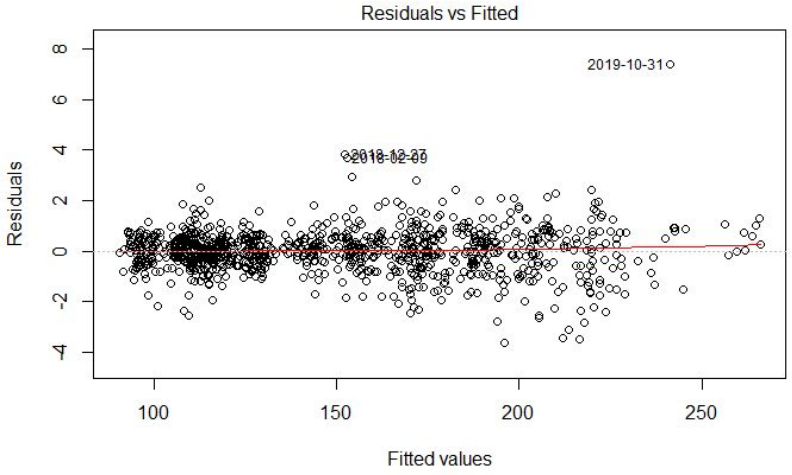
6.3 Model Selection

6.3.1 Multivariate Linear Regression

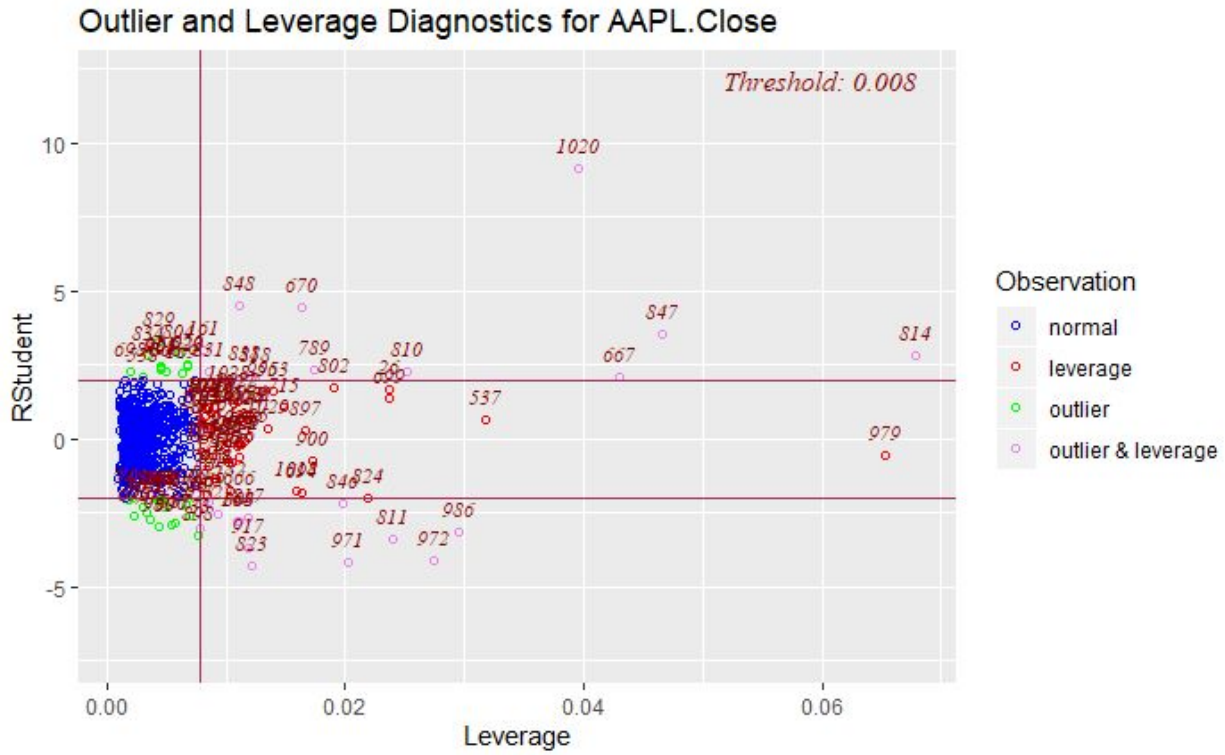
We built a multivariate linear regression model as a baseline model to accommodate all these three predictors. This model explained 99.95% of the variance and all predictors were significant. The predicted values were close to the actual ones with test RMSE 0.83377.



We inspected residual plot and confirmed that it was an approximate linear trend. The evidence of non-constant variances of error terms were not really obvious because a funnel shape in the residual plot was not present. However, we need to further address if there is any heteroscedasticity in this data set.



We did outlier and leverage diagnosis. As expected, there were lots of outliers and some of them were leverage points. It implied that Apple stock had not always been stable, but it is fair since the financial market can be affected by many factors. Therefore, we did not have a good reason to delete those outliers.



We assessed the possibility of multicollinearity by computing the variance inflation factor (VIF). Values of VIF of all predictors, 1517.2387, 1299.9905 and 902.1389 are substantially large compared to 5 or 10. However, no matter which predictor was dropped, the accuracy rate dropped and adjusted R-square also dropped. Therefore, we kept all the predictors.

6.3.2 Cross Validation Multivariate Linear Regression w PCA Preprocessing

We built and 10*9-fold cross validation multivariate linear regression model using the model from the caret package. It turned out that the test RMSE is identical to the baseline model. We also try a 10*9-fold cross validation multivariate linear regression model with center, scale and PCA preprocessing in the hope of solving the multicollinearity problems. However, the test RMSE, 1.25891, is 0.42514 higher than the baseline model.

6.3.3 Principal Component Regression (PCR) & Partial Least Squares Regression (PLSR)

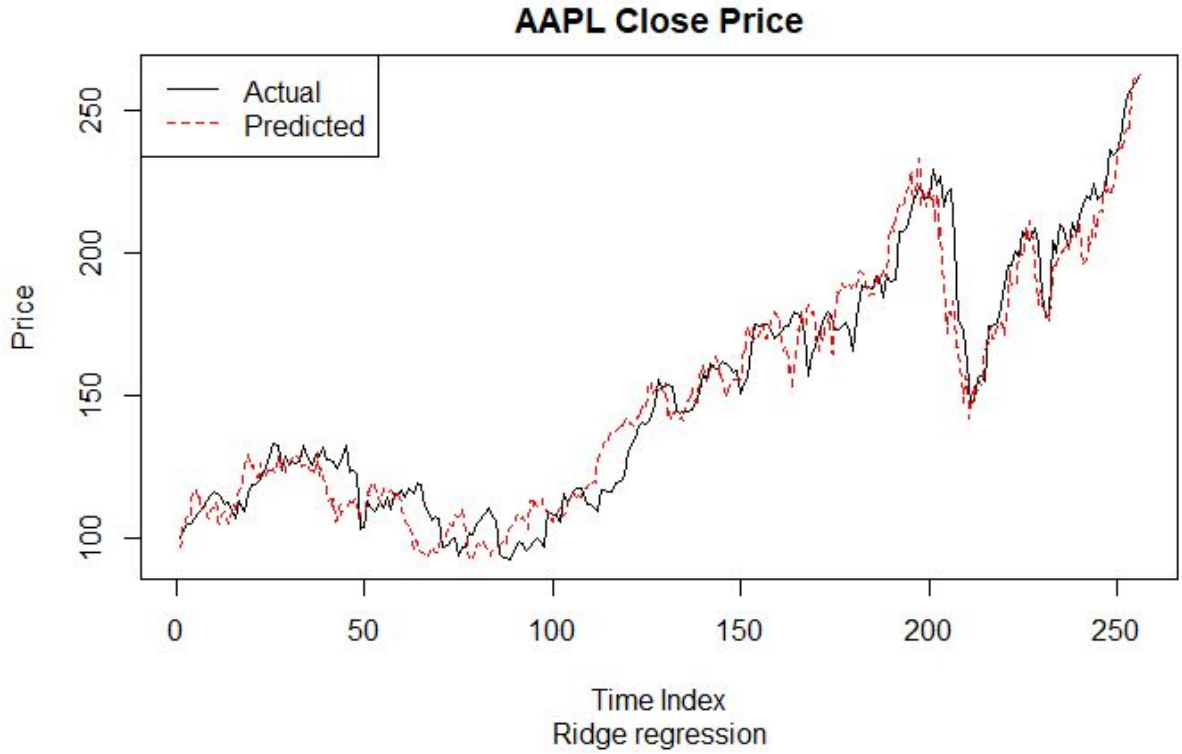
Since PCA preprocessing did not solve the multicollinearity problems, we tried the PCR and PLSR models. Their test RMSEs are identical to the baseline model. Therefore, multicollinearity would not be a concern in this data set.

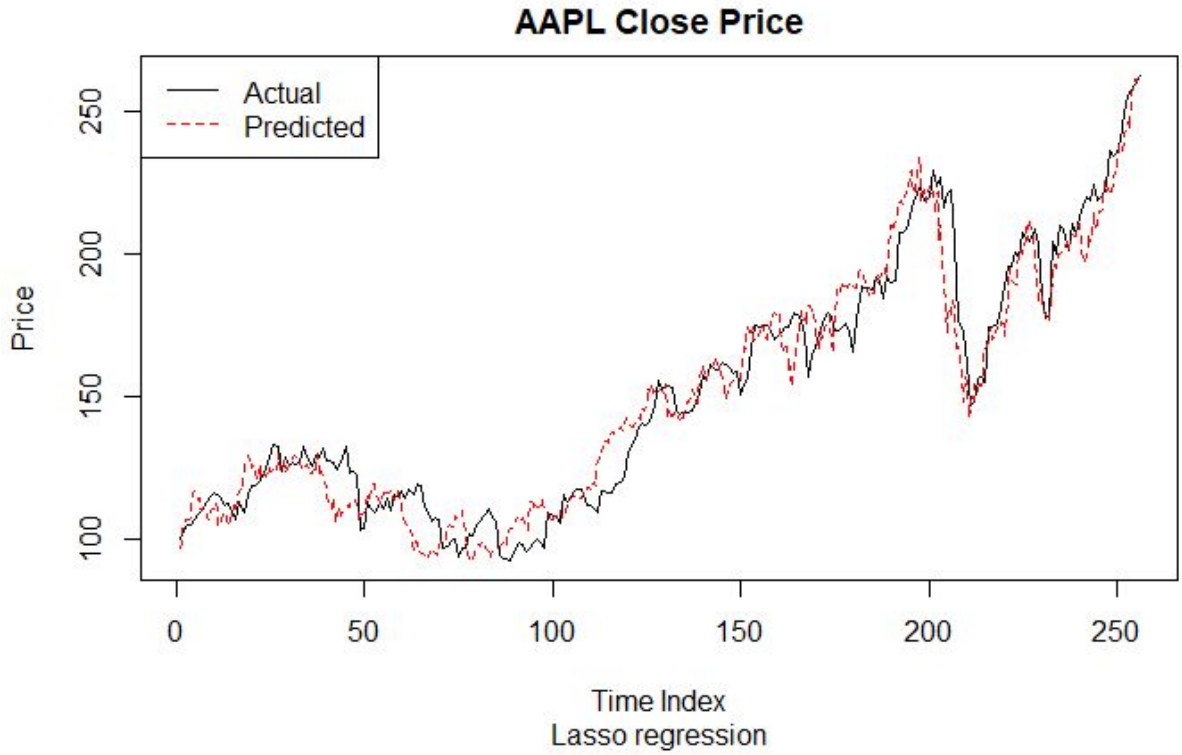
6.3.4 Ridge Regression & Lasso Regression

Since these two methods has the effect of reducing variance through regularization, each model was built. The ridge coefficient estimates were shrunk substantially, while lasso model performed variable selection. However, these models are still defeated by the baseline model.

```
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
```

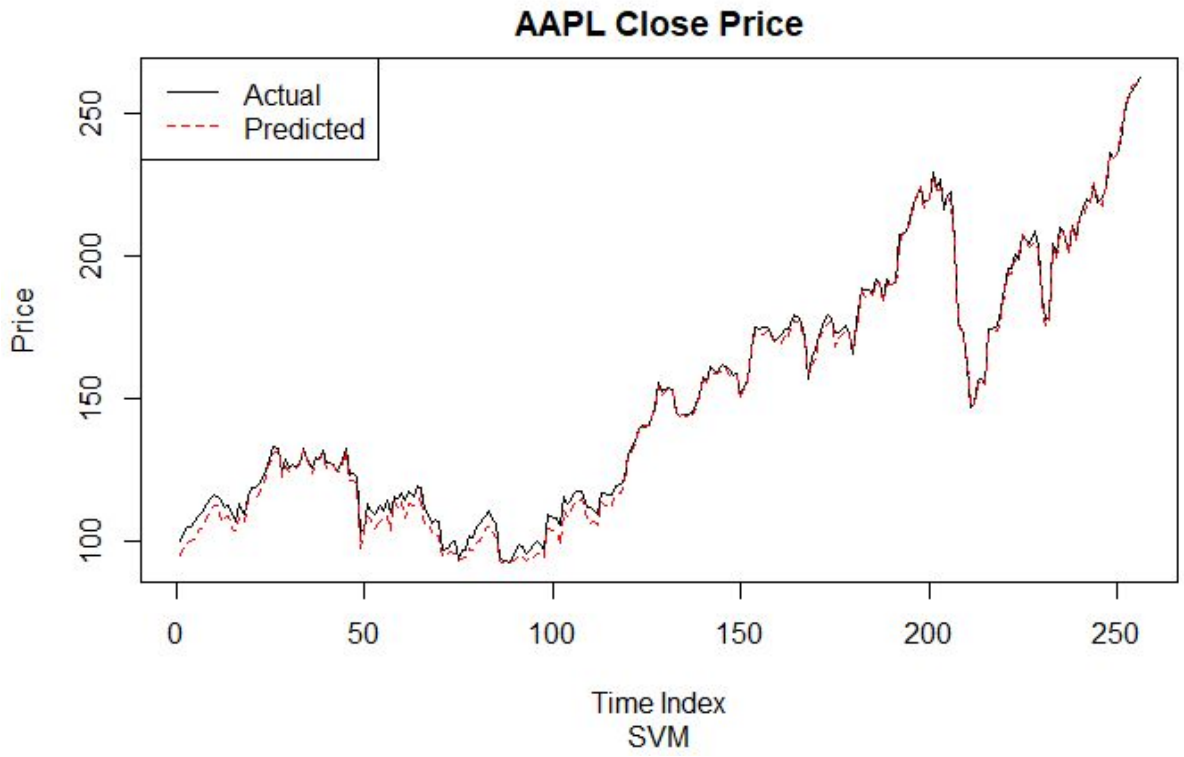
```
Lasso model test RMSE: 1.313807

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
```


AAPL Close Price
Actual
Predicted
Price
100
150
200
250
0
50
100
150
200
250
Time Index
Ridge regression


AAPL Close Price
Actual
Predicted
Price
100
150
200
250
0
50
100
150
200
250
Time Index
Lasso regression

6.3.5 Polynomial Regression & Support Vector Machine

We tried a polynomial regression with degrees from 1 to 5 and found that the model with degree 1 had identical test RMSE as the baseline model. We also built a SVM regression model, but it had the worst test RMSE, 1.55655, so far.



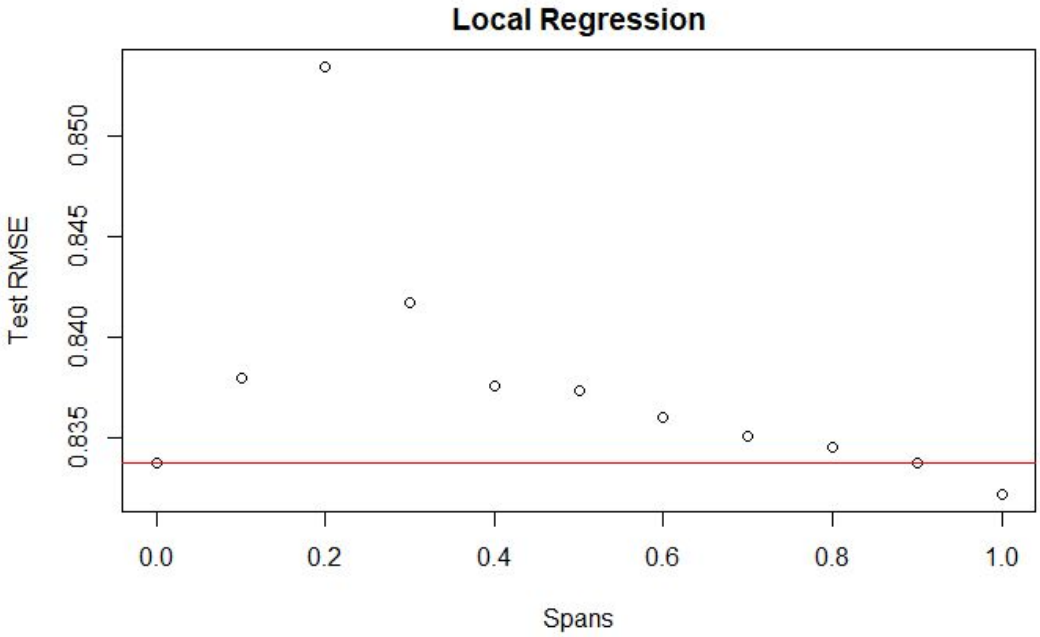
6.3.6 Natural Splines Regression & Smoothing Splines Regression

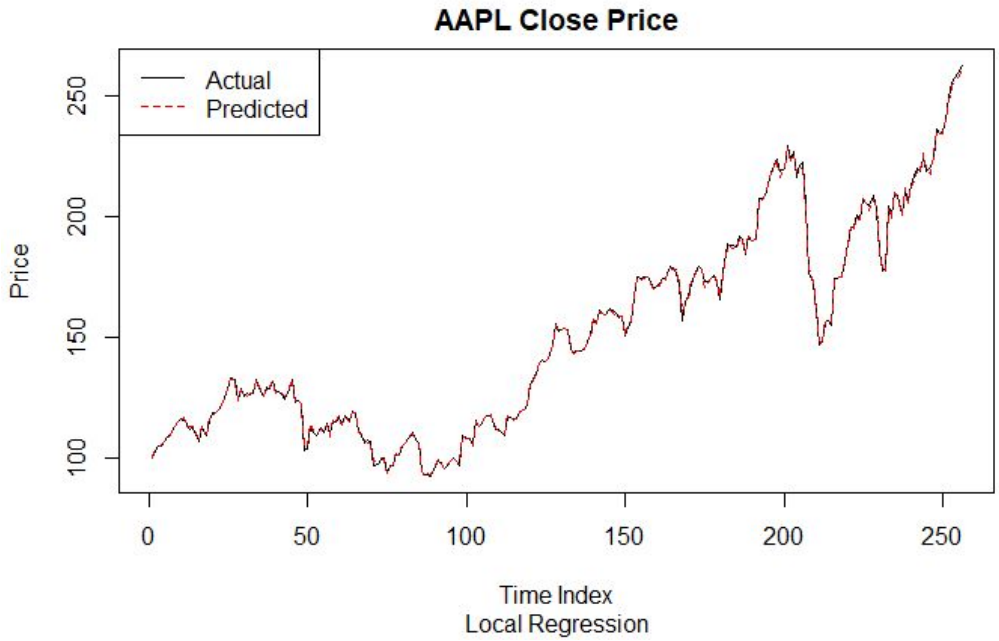
Since the autocorrelation plot showed that the data set was a random walk model with seasonal linear trend, we expected the regression spline model would be a better fit for the data. The natural splines regression model has identical test RMSE as the baseline one. The smoothing splines regression model with 1 degree of freedom slightly outperformed the baseline one.

6.3.7 Local Regression & K-Nearest Neighbors

Local regression computes the fit at a target point using only nearby training observations, the smaller the value of span s, the more local and wiggly will be the fit; the larger the s, the more

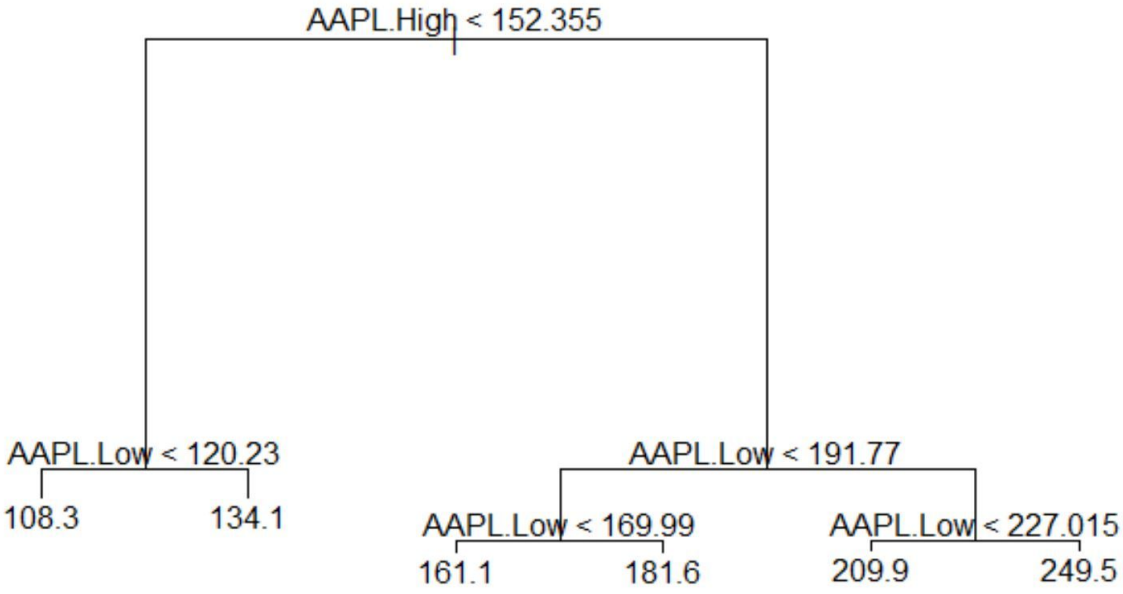
global global and smooth will be the fit (James, etc., 2017). Local regression can perform poorly if features are larger than about 3 or 4. We built a GAM local regression model with span from 0.1 to 1. When the span equaled 1, as we were expecting, the model had the lowest test RMSE so far. Nearest neighbor regression uses similar ideas to fit a data point and also suffers from high dimensions. However, the nearest neighbor regression model did a much worse job than the baseline one.



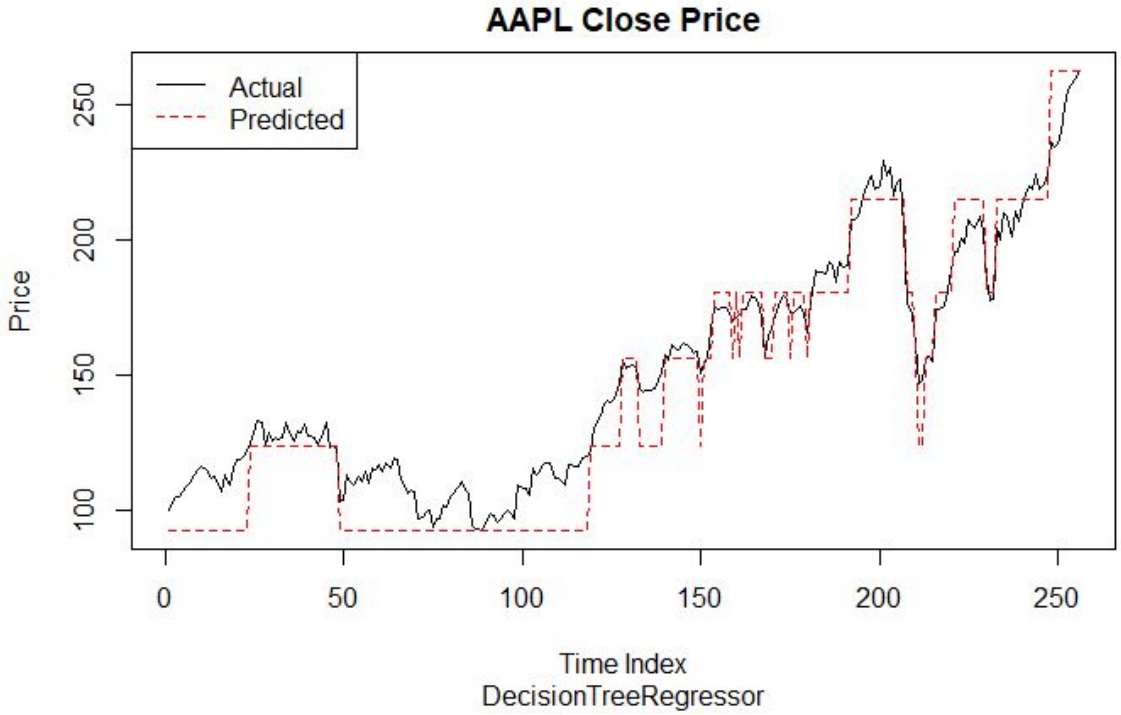


6.3.8 Decision Tree Regression & Ensemble Methods

Since we had been convinced by the previous models that there was an approximate linear relationship in the data set, we expected the decision tree regression model perform poorly. It turned out that the test RMSE (8.07192) was almost 10 times larger than the baseline model.



Bagging, Random Forest and Boosting

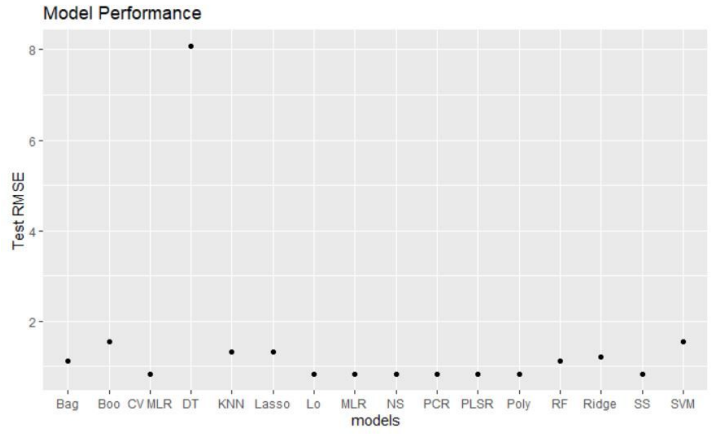


These ensemble methods can often dramatically improve the prediction accuracy at the expense of some loss of interpretability and visualization. We built and tested all these three models respectively. Bagging and random forest models did improve the prediction accuracy, 1.11095 and 1.11791 respectively, compared to the decision tree regression one. However, boosting only outperformed SVM and decision tree models.

6.4 Regression Conclusion & Future Work

Since there are negligible prediction improvement of local regression and smoothing splines over multivariate linear regression, it is hard to choose the best one among these three models. Future prediction results will be kept to assess the performance of these models over time.

We are planning to perform SARIMA and RNN models to predict the close price of Apple stock. SARIMA model supports time series data forecasting while explicitly taking into account the seasonal component (Brownlee, 2019). RNN makes use of a sequence of data and outputs a sequence of data which every output element depends on the computation of previous elements (Britz, 2015). This makes RNN become one of the ideal tools to forecast time series data.



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

7. Classification: Prediction of Close Price Direction

We used the dailyReturn function to build a 2-level factor variable to the direction of the stock (up or down). Using the lag() function contained in quantmod, we created 10 lagged return vectors, where each vector is the stock's returns shifted 1:10 days down. The volume function Vo() extracted the volume from the original xts object. Using the na.omit() function we ensure that we have no observations with missing data in our model.

After cleaning our data, we use the createDataPartition() function from the caret package to separate our data ranging from 2014-10-14 to 2019-11-19 into an 80:20 test/train split

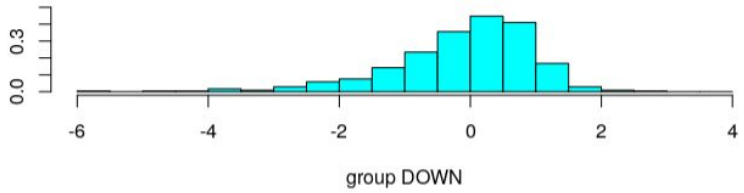
7. 1 Linear Discriminant Analysis

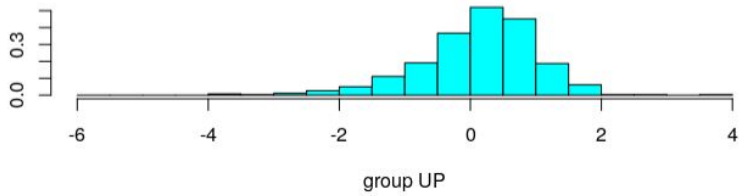
Linear Discriminant Analysis (LDA) is a supervised learning algorithm that uses Bayes' theorem to estimate the posterior probability density function. LDA is a dimensionality reduction, linear classifier: it seeks to find a linear combination of input feature variables that act as a decision rule for classifying unseen observations. It splits the input data with a decision boundary separate the data points by class.

To perform LDA in R, we used the lda() function in the MASS package (found here: https://cran.r-project.org/web/packages/MASS/MASS.pdf) to fit our model. The function takes on a formula y ~ x1 + x2 + ..., and a data frame.

```
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
```

The lda.fit output indicates that 48% of our training observations corresponded to down, while 52% corresponded to upward trending stocks. Plotting the histogram of our lda model, the similar spread and mean of the two plots suggests that the lda model does not do a great job at linearly separating the two classes.





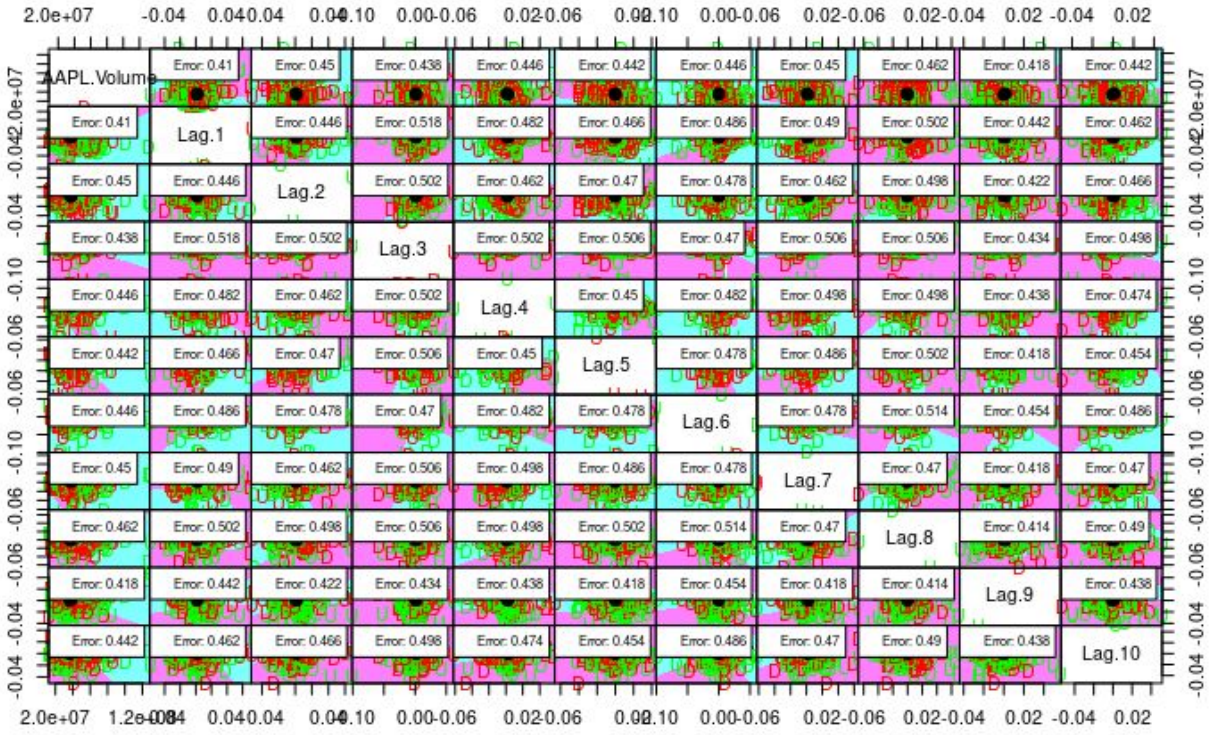
The confusion matrix below visually represents the classification of our data. The model predicted classifications of our data are row-wise while the actual classes are column wise. Summing up the correctly classified elements and dividing by the total number of observations gives us our accuracy rate, 52.6%

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

[1] 0.5261044
```

7.2 Quadratic Discriminant Analysis

QDA holds the same assumptions as LDA, except that it does not assume the covariance matrix is the same for the two classes, leading to a quadratic decision boundary.

We use the qda() function in MASS to fit our model. The function takes on a formula y ~ x1 + x2 + ..., and a data frame.



The partition plot from the partimat() function in the klaR library shows the classification of our qda() model based on every combination of two variables.

As with lda(), the confusion matrix visually represents the classification of our data and we get an accuracy score of 54%.

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

7.3 SVM

Support vector machines classify data by creating a decision boundary (linear hyperplane) such that most points in one category fall on one side of the boundary while most points in the other category fall on the other side of the boundary. The optimal hyperplane maximizes the distance from the plane to any point. Though SVCs only apply linear boundaries, SVMs apply non-linear kernel functions to map inputs into higher dimensional space and linearly classify in that space.

In order to test the best hyperparameters for our model, we use the tune() function from the package e1071. This function tunes specified hyperparameters using a grid search. We tested both linear and radial kernels for our model.

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

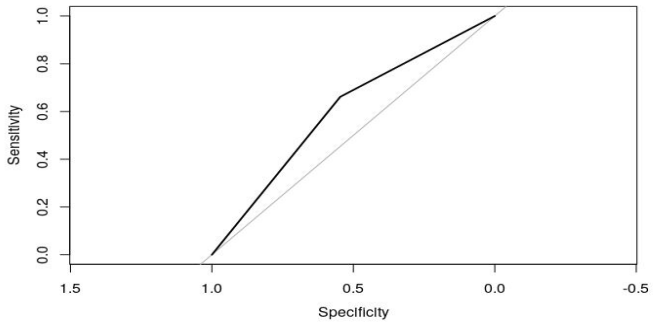
```
Parameter tuning of 'svm':

- 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


  Accuracy      Kappa
0.54618474 0.05532986
```

After tuning the model, we found that the best cost for the training data was .01 with a linear kernel. The best training error rate was .48. Then predicting on test data, we were able to get 54 % accuracy, higher than either the LDA or QDA. Previous work on using SVMs to classify stock prediction suggest that a 55-60% accuracy rate is possible [3] However, their model used more financial features such as price volatility, price momentum,sector volatility, and sector momentum. Though we did not have the time or financial knowledge to effectively utilize these features, future work should leverage these features to possibly gain higher accuracies.

7.4 K-Nearest Neighbors

KNN is a supervised machine learning algorithm that classifies data points based on the classes of the K nearest points. For our analysis, in order to choose the k we iterated through 1:10 and chose the one with the best accuracy on our training data. From this, we selected k=6 as the parameter.

Plotting the ROC curve, which is a visual representation of the performance of our classifier, we get an AUC of .6039, meaning our model is moderately good at classifying our data (0 is the worst, 1 is perfect).




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

Looking at our confusion matrix:

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

The accuracy of our KNN model on the test data was .606 - the highest of all 4 classifiers we tested.

7.5 Classification Conclusion

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

KNN > SVM > QDA > LDA

The classifier with the best performance was KNN, suggesting that the non-parametric quality of the model did better with our stock data for predicting direction. While we were unable to fit a classifier to get above 60% accuracy, given more time, future work with classification would involve using ensemble techniques to transform our weak learners into stronger learners. For instance, we could use bagging to decrease the variance in our predictions or boosting to adjust the observation weights. In addition, as mentioned there is a lot of room in our models to use more advanced financial features such as price volatility, price momentum, sector volatility and sector momentum or outside features such as news sentiment or macroeconomic trends to increase the accuracy of our predictions.

8. Data Sources

Our stock data was sourced from Yahoo!Finance.

9. Source Code

Packages used:

- quantmod
- zoo
- xts
- caret
- MASS
- class
- pROC
- e1071
- kLAR
- astsa
- leaps
- glmnet
- olsrr
- car
- gam
- tree
- randomForest
- gbm

References and Related Works

[1] Britz, Denny. 2015. Recurrent neural networks tutorial, Part 1 – Introduction to RNNs. Retrieved from http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

[2] Brownlee, Jason. 2019. A gentle introduction to SARIMA for time series forecasting in Python. Retrieved from https://machinelearningmastery.com/sarima-for-time-series-forecasting-in-python/

[3] Madge, Saahil. 2015. Predicting Stock Price Direction using Support Vector Machines. Retrieved from https://www.cs.princeton.edu/sites/default/files/uploads/saahil_madge.pdf

[4] James, G, Witten D., Hastie, T, etc. 2017. An introduction to statistical learning with application in R.