Texas Instruments Stock Performance

Andrew Esch, Evan Lee, Collin Stratton

1/16/2022

# Introduction

The purpose of this assignment is to take the data from the Texas Instruments stock and, given its performance characteristics and investment objectives, predict its future performance using linear modeling. This project also serves as an introduction to R and linear modeling.

## Data Setup and exploration

Import the data and set the dataframe as txn.

# If you need to knit this, use your full directory path (with forward-slashes) like this: C:/Users/andre/Documents/GitHub/CST-425/PerformancePredictions/TXN.csv  
txn <- read.csv(file='C:/Users/Evan/OneDrive/Documents/GitHub/CST-425/PerformancePredictions/TXN.csv', header=TRUE, sep=',')  
txn.df <- data.frame(txn)  
head(txn.df)

## Date Open High Low Close Adj.Close Volume  
## 1 2017-01-11 74.64 75.20 74.46 75.20 66.00017 4361400  
## 2 2017-01-12 75.00 75.14 74.09 74.85 65.69299 4408200  
## 3 2017-01-13 74.83 75.30 74.73 75.00 65.82464 3372500  
## 4 2017-01-17 74.67 74.95 74.29 74.49 65.37703 4413100  
## 5 2017-01-18 74.73 74.84 74.25 74.38 65.28049 5531400  
## 6 2017-01-19 74.22 74.81 73.87 73.88 64.84164 4468500

# Check for missing values  
sum(is.na(txn.df))

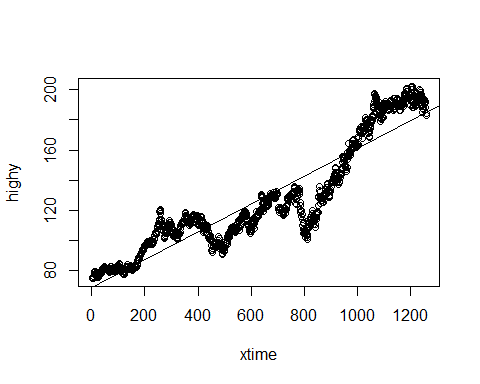
## [1] 0

# Find data correlation  
cor(txn.df[2:7])

## Open High Low Close Adj.Close Volume  
## Open 1.0000000 0.9994412 0.9994682 0.9987938 0.9981596 -0.2227816  
## High 0.9994412 1.0000000 0.9992375 0.9993760 0.9989188 -0.2106327  
## Low 0.9994682 0.9992375 1.0000000 0.9994275 0.9985675 -0.2332480  
## Close 0.9987938 0.9993760 0.9994275 1.0000000 0.9992420 -0.2235351  
## Adj.Close 0.9981596 0.9989188 0.9985675 0.9992420 1.0000000 -0.2182470  
## Volume -0.2227816 -0.2106327 -0.2332480 -0.2235351 -0.2182470 1.0000000

Use the pracma library to plot the data from Texas Instruments. The graph below depicts the price of the stock at its highest over time. Furthermore, abline is used to draw a line to create a linear model predicting the price of the stock.

library(pracma)  
# create an array x that contatins time equal to the recorded high prices  
xtime <- linspace(1,1259, n=1259)  
highy <- txn.df[,"High"]  
  
# plot time vs high price and add a line to create a linear model  
plot(xtime,highy)  
abline(lm(highy~xtime))



From the graph it is shown that Texas Instruments has performed well over the time frame selected. The linear plot of the data predicts the price of the stock will continue to increase over time. From here, more tests will be performed to determine if the linear plot is accurate.

## Training and Testing Sets Preparation

Install caTools and use the function sample to split the data into training and testing sets. The head of the training set is shown below.

library(caTools)

##   
## Attaching package: 'caTools'

## The following objects are masked from 'package:pracma':  
##   
## combs, trapz

set.seed(123) # to reproduce the sample  
  
# Split training and testing based on "High" variable  
sample <- sample.split(txn.df$High, SplitRatio = 0.7) # Split using 70/30 ratio (backed up with research)  
  
# Create training/testing csv files  
txn.df.train <- subset(txn.df, sample==TRUE)  
txn.df.test <- subset(txn.df, sample==FALSE)  
  
# Display training dataframe  
head(txn.df.train)

## Date Open High Low Close Adj.Close Volume  
## 1 2017-01-11 74.64 75.20 74.46 75.20 66.00017 4361400  
## 3 2017-01-13 74.83 75.30 74.73 75.00 65.82464 3372500  
## 6 2017-01-19 74.22 74.81 73.87 73.88 64.84164 4468500  
## 7 2017-01-20 74.16 74.89 74.16 74.75 65.60523 6661400  
## 9 2017-01-24 76.19 77.32 75.91 77.08 67.65017 8561600  
## 10 2017-01-25 77.72 78.72 77.19 78.58 68.96667 8112800

## Test Set Predictions (Build the model)

Build a model using the training set for the highest price of the stock over the given time frame. A summary model is shown below.

# Build model using training set  
highModel <- lm(High ~., data = txn.df.train[2:7])  
summary(highModel)

##   
## Call:  
## lm(formula = High ~ ., data = txn.df.train[2:7])  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -2.3537 -0.4165 -0.0617 0.3129 6.9418   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.003e+00 2.924e-01 3.431 0.000629 \*\*\*  
## Open 6.116e-01 2.244e-02 27.250 < 2e-16 \*\*\*  
## Low -1.973e-01 3.406e-02 -5.793 9.66e-09 \*\*\*  
## Close 4.717e-01 2.817e-02 16.741 < 2e-16 \*\*\*  
## Adj.Close 1.128e-01 1.659e-02 6.803 1.90e-11 \*\*\*  
## Volume 1.689e-07 1.216e-08 13.896 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.7208 on 875 degrees of freedom  
## Multiple R-squared: 0.9996, Adjusted R-squared: 0.9996   
## F-statistic: 4.499e+05 on 5 and 875 DF, p-value: < 2.2e-16

## Improve the Model using only Significant Variables

Using the P Value of the model gained from the previous section, the next step is to determine which variables are significant and use the training set to build a model with only significant variables. The code section below gives a summary of the significant variables found.

# Choose only variables with p-value < 2.2e-16  
highModel.significantVars <- lm(High ~ Open + Close + Volume, data = txn.df.train)  
highModel.significantVars

##   
## Call:  
## lm(formula = High ~ Open + Close + Volume, data = txn.df.train)  
##   
## Coefficients:  
## (Intercept) Open Close Volume   
## -1.077e+00 5.189e-01 4.910e-01 2.158e-07

summary(highModel.significantVars)

##   
## Call:  
## lm(formula = High ~ Open + Close + Volume, data = txn.df.train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -2.9152 -0.4304 -0.0641 0.2887 7.2438   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -1.077e+00 1.192e-01 -9.038 <2e-16 \*\*\*  
## Open 5.189e-01 1.453e-02 35.704 <2e-16 \*\*\*  
## Close 4.910e-01 1.454e-02 33.773 <2e-16 \*\*\*  
## Volume 2.158e-07 1.114e-08 19.381 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.7565 on 877 degrees of freedom  
## Multiple R-squared: 0.9996, Adjusted R-squared: 0.9996   
## F-statistic: 6.807e+05 on 3 and 877 DF, p-value: < 2.2e-16

## Regression Output Interpretation

After finding each significant variable, the next step is to determine the regression output interpretation by determining the number of fitted values that best fit the TXN model.

# The variables used in the model  
names(highModel.significantVars)

## [1] "coefficients" "residuals" "effects" "rank"   
## [5] "fitted.values" "assign" "qr" "df.residual"   
## [9] "xlevels" "call" "terms" "model"

# The number of fitted values in the model  
length(highModel.significantVars$fitted.values)

## [1] 881

### Calculate residuals

The residuals are calculated using the fitted values gathered from the significant variables. The residuals are shown below.

#Calculate the predicted values using the highmodel fitted values  
predicted.train <- highModel.significantVars$fitted.values  
  
#print out first 6 figures of predicted training model  
head(predicted.train)

## 1 3 6 7 9 10   
## 75.51709 75.30405 74.67416 75.54348 78.15097 79.58450

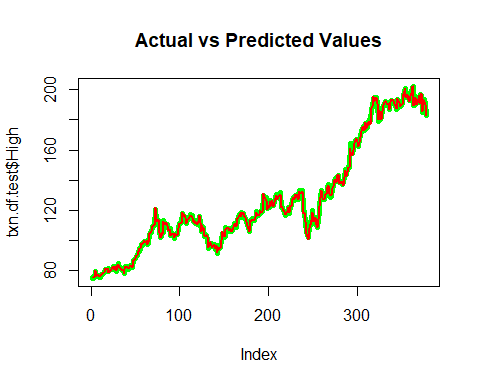
#make predicted training model a data frame  
predicted.train.df <- data.frame(predicted.train)  
  
# Calculate residual values  
predicted.train.df.residuals <- highModel.significantVars$residuals  
head(predicted.train.df.residuals)

## 1 3 6 7 9 10   
## -0.317088965 -0.004042367 0.135841485 -0.653485413 -0.830971852 -0.864501509

### Make predictions using the test set

It appears that the model for the predicted values are greater than the test model, which means the model overvalued each entry.

predicted.test <- predict(highModel.significantVars, newdata = txn.df.test)  
predicted.test.df <- data.frame(predicted.test)  
  
# Plot actual values vs predicted values  
plot(txn.df.test$High, col="green", type="l", lty=1, lwd=5, main = "Actual vs Predicted Values")  
lines(predicted.test.df, col="red", type="l", lty=1, lwd=2.5)

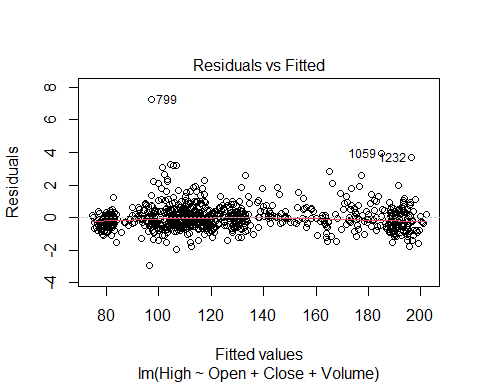


## Verify Model

There’s a lot of steps to verify the model (definitely more than 4) 1. Confirm linearity 2. Confirm normality 3. Check for Homoscedasticity 4. Confirm if there are significant outliers that affect the model 5. Verify that the residuals are independent 6. Verify Homoscedasticity A. NCV Test (not required unless there is a pattern in verifying Homoscedasticity) 7. Verify colinearity

### Step 1: Confirm linearity

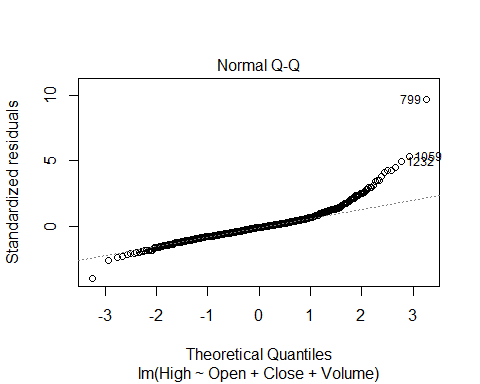
plot(highModel.significantVars, which=1)



Explanation: To confirm linearity, check the graph for non-linear patterns. Although the data shows that there are several residual outliers, the data altogether forms a clear linear pattern. In addition, the residuals appear to be spread-out. Thus, the residuals vs. fitted graph shows that linearity can be assumed.

### Step 2: Confirm normality

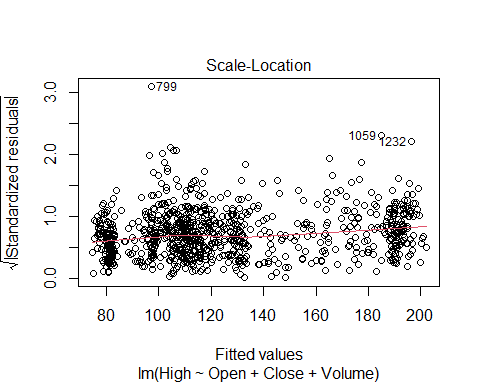
# Verify that the residuals are normally distributed  
plot(highModel.significantVars, which=2)



Explanation: To confirm normality, it must be verified that the standardized residuals vs. the theoretical quartiles form a normal distribution. Although the data appears to curve away from the line at the borders of the graph, the plot shows a clear straight line. Consequently, normality is normality distributed and the normality assumption is satisfied.

### Step 3: Check for Homoscedasticity

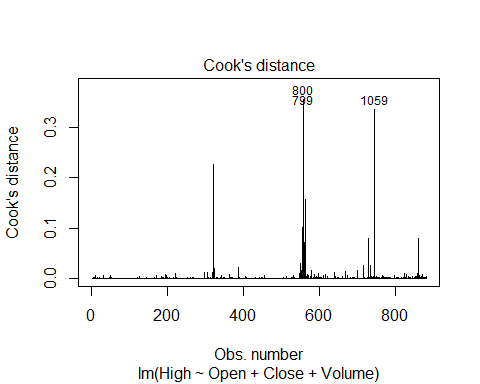
# Verify that the residuals are randomly spread  
plot(highModel.significantVars, which=3)



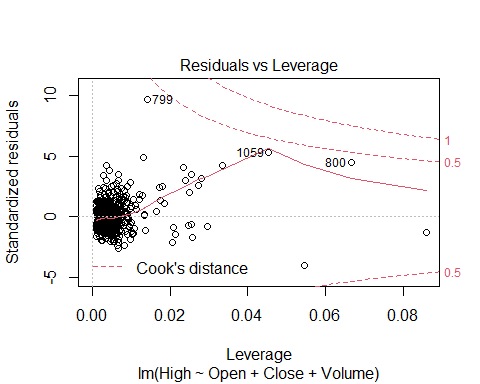
Explanation: There are three outliers: 736, 1056, and 1232. However, the data appears to form a linear pattern and are randomly-spread. Thus, we can confirm that homoscedasticity assumption is verified.

### Step 4: Verify if there are any outliers that could skew results

# Assess the presence of significant outliers that could skew the results  
plot(highModel.significantVars, which=4)



plot(highModel.significantVars, which=5)



Explanation: Using the first graph to verify the cook’s distance vs. observation number, there are 3 outliers that appear: 799, 800, 1059. However, the general average cook’s distance vs. observation number appears to stay around 0.0. To confirm that these are outliers, graph #2 demonstrates that the same observation numbers also appear as outliers in the standardized residuals vs. leverage plot. Although there appears to be several data points that stray away from the main cluster of data, all data fits within the cook’s distance lines. Thus, the graphs affirm that there are no significant outliers that significantly affect the model.

### Step 5: Verify that there is no autocorrelation

# Verify that the residuals are independent (i.e. not auto-correlated)  
library(car) # companion to applied regression

## Loading required package: carData

##   
## Attaching package: 'car'

## The following object is masked from 'package:pracma':  
##   
## logit

# Test for autocorrelation using the Durbin-Watson Test  
durbinWatsonTest(highModel.significantVars)

## lag Autocorrelation D-W Statistic p-value  
## 1 0.2728448 1.452077 0  
## Alternative hypothesis: rho != 0

Explanation: To confirm that the residuals are independent, the Durbin-Watson test is one method to determine if an autocorrelation exists.

Autocorrelation is the correlation measure of a lagged time series in comparison to a current time series. In other words, to avoid a bias in the linear regression prediction model, there cannot be statistical evidence of an autocorrelation. The null hypothesis is that there is no autocorration. The alternative hypothesis would state that there is an autocorrelation. For this model, use the standard that the p-value < 0.001. Using the Durbin-Watson test, the test-statistic and p-value and insigificant (D-W statistic = 1.45, p-value = 0). Thus, we fail to reject the null hypothesis. This means that there is no statistical evidence that the residuals are not independent and the assumption is satisfied.

### Step 6: Verify that there is no colinearity

# Test Collinearity using Variance Inflation Factor (VIF)  
vif(highModel.significantVars)

## Open Close Volume   
## 425.664485 425.916494 1.051044

# Test that SQRT(VIF) > 5  
sqrt(vif(highModel.significantVars)) > 5

## Open Close Volume   
## TRUE TRUE FALSE

Explanation: Reviewing the vif outputs for the three significant variables, the “Open” and “Close” variables show a significantly high VIF value (vif = 425). Consequently, these variables has a sqrt(vif) > 5 and failed the collinarity test. This means that the variables “Open” and “Close” are correlated to each other.

### Final financial “High” prediction results

predicted.test <- predict(highModel.significantVars, newdata = txn.df.test)  
predicted.test.df <- data.frame(predicted.test)  
head(predicted.test.df[order(predicted.test.df$predicted.test),], 20)

## [1] 75.19521 75.41370 75.54214 76.08057 76.37498 76.40719 76.42209 76.74691  
## [9] 76.98636 77.16974 77.52983 77.59773 78.32859 78.46715 78.56303 78.86475  
## [17] 79.72664 79.83957 79.89450 79.96517

## Christian Worldview

When analyzing a publicly traded stock, one must not only consider the financial values of the company, but also the moral and ethical values of the company. This analysis done for TXN does not always mean that this is a good investment. In Matthew 25:14-30, the parable describes a master investing into three servants, each with varying levels of return. Two servant doubles the amount given, while the third does not make any. This parable is shown that knowing a company is reliable and trustworthy is important to investing as well.

## References

<https://finance.yahoo.com/quote/TXN/history?period1=1484438400&period2=1642204800&interval=1d&filter=history&frequency=1d&includeAdjustedClose=true> <https://padlet.com/isac_artzi/9014mn5elmn7r8sg> <https://www.researchgate.net/publication/328589123_On_Splitting_Training_and_Validation_Set_A_Comparative_Study_of_Cross-Validation_Bootstrap_and_Systematic_Sampling_for_Estimating_the_Generalization_Performance_of_Supervised_Learning>