

# Use Machine Learning To Possibly Become A Millionaire: Predicting The Stock Market?

Nisha Selvarajan

11/19/2020

- Objectives:
- Data Import
- Data Science Analysis
- Numerical Summaries
- Correlation and Regression
- Inferential Statistics
- Hypothesis Test, Mean Change Price
- Paired Tests
- Independent Tests
- Regression Diagnostic Plots
- MultiCollinearity
- Regression- Find if Market.Cap is affected by trading technical indicators
  - Linear Regression
  - Polynominal Regression
  - Spline Regression
  - Generalized Linear Model
  - Penalized Cubic Regression Spline
  - Extreme Gradient Boosting
  - Compare Regression Models
  - Conclusion
- Generalized Linear Model
  - GAM Regression Prediction & Accuracy
  - GAM Scatter Plot 3D Visualization
- Extreme Gradient Boosting
  - Extreme Gradient Boosting Prediction & Accuracy
- Decision Tree to Trade Bank of America Stock (Real Time Value Analysis)
  - Building a Strategy
  - Trained Decision Tree classifier results:
  - Plot Decision Tree
- KNN Classifier to predict if daily stock price will increase
  - Building KNN model
  - KNN Accuracy
  - Overall insights obtained from the implemented project
- K-means Clustering NYSE trades.
  - KNN Cluster Plotting
  - Conclusion
- ARIMA Forecasting Expedia Stock Price incorporating COVID-19 (Real Time)
  - Graphical Representation of Data
  - ARIMA Model
  - Diagnostic measures
  - Augmented Dickey-Fuller & Kwiatkowski-Phillips-Schmidt-Shin
  - Forecasting with HoltWinters
  - Conclusion
- Using a Neural Network to Model the Amazon Stock Price Change (Real Time Analysis)
  - Normalize Data
  - Machine Learning: Classification using Neural Networks
  - Accuracy of Neural Network model:
  - Conclusion
- Using SVM to predict price change for WALMART (Real Time Analysis)
  - Machine Learning: Classification using SVM
  - SVM Classifier using Linear Kernel
  - SVM Classifier using Non-Linear Kernel

- Comparision between SVM models
  - Conclusion
- Naive Bayes Sentimental Analysis of Twitter Data for Apple Stock
  - Creating Corpus
  - Positive Bag Of Words
  - Negative Bag Of Words & wordcloud for negative
  - Interpretation of the results and prediction accuracy achieved
  - Overall insights obtained from the implemented project
- Amazon Price Trend Predict - Random Forest (Real Time Analysis)
  - Methodology
  - Prediction & Accuracy.
  - Ensemble Method for Random Forest
  - Overall insights obtained from the implemented project
- Discussion and Conclusions

## Objectives:

**Use Machine Learning To Possibly Become A Millionaire: Predicting The Stock Market?** The stock market is one of the most well-known infrastructures through which anyone can potentially make a fortune. If anyone could crack the code to predicting what future stock prices are, they'll practically rule the world. here's just one problem. It's pretty much impossible to accurately predict the future of the stock market.

In this project, we will work with historical data about the stock prices of a publicly listed company. We will implement a mix of machine learning algorithms to predict the future stock price of the company, starting with simple algorithm like linear regression, and then move on to advanced techniques like Auto ARIMA and Neural Networks.

## Data Import

- The data set we will be working with is from the New York Stock Exchange (NYSE) and represent the historical prices and other fundamental data points of the S&P 500 from 2010 to the end of 2016. Dataset consists of following files:
- prices.csv: raw, as-is daily prices. Most of data spans from 2010 to the end 2016, for companies new on stock market date range is shorter. There have been approx. 140 stock splits in that time, this set doesn't account for that.
- prices-split-adjusted.csv: same as prices, but there have been added adjustments for splits.
- securities.csv: general description of each company with division on sectors
- fundamentals.csv: metrics extracted from annual SEC 10K filings (2012-2016), should be enough to derive most of popular fundamental indicators.
- The majority of our focus will be on the prices-split-adjusted.csv file, as this contains the adjusted prices for the stocks we will be trying to predict.

**Description of fundamentals.csv**

Names	Description
Stocks Variable Columns	Categorical - Ticker.Symbol,Period.Ending
Stocks Technical Indicator Columns	Numerical - Accounts.Payable, Accounts.Receivable, Add.I.income.expense.items, After.Tax.ROE, Capital.Expenditures, Capital.Surplus, Cash.Ratio, Cash.and.Cash.Equivalents, Changes.in.Inventories,Common.Stocks, Cost.of.Revenue,Current.Ratio, Deferred.Asset.Charges,Total.Assets, Deferred.Liability.Charges, Depreciation,Earnings.Before.Interest.and.Tax, Earnings.Before.Tax,Net.Income.Adjustments, Effect.of.Exchange.Rate, Equity.Earnings.Loss.Unconsolidated.Subsidiary, Fixed.Assets,Goodwill,Gross.Margin, Gross.Profit,Income.Tax, Intangible.Assets,Interest.Expense, Inventory,Investments,Liabilities, Long.Term.Debt,Long.Term.Investments, Minority.Interest,Misc..Stocks, Net.Borrowings,Net.Cash.Flow , Net.Cash.Flow.Operating,Net.Cash.Flows.Financing, Net.Cash.Flows.Investing,Net.Income, Net.Income.Applicable.to.Common.Shareholders, Net.Income.Cont.,Operations, Net.Receivables,Non.Recurring.Items, Operating.Income,Operating.Margin, Other.Assets,Other.Current.Assets, Other.Current.Liabilities, Other.Equity,Other.Financing.Activities, Other.Investing.Activities,Other.Liabilities, Other.Operating.Activities,Pre.Tax.Margin Other.Operating.Items,Pre.Tax.ROE, Profit.Margin,Quick.Ratio, Research.and.Development, Retained.Earnings,Short.Term.Debt Sale.and.Purchase.of.Stock, Sales..General.and.Admin., Current.Portion.of.Long.Term.Debt, Short.Term.Investments,Equity, Total.Current.Assets,For.Year Total.Current.Liabilities, Total.Equity,Total.Liabilities,

Names	Description
	Total.Revenue,Treasury.Stock, Earnings.Per.Share, Estimated.Shares.Outstanding

**Description of securities.csv**

Names	Description
Ticker.symbol	Categorical - Ticker.Symbol
GICS.Sector	Categorical -Industry of the ticker
GICS.Sub.Industry	Categorical -Sub.Industry of the ticker

**Description of sprice.csv**

Names	Description
Date	Date - Date for which prices are recorded
Symbol	Categorical - Ticker.Symbol
Open	Numerical - Opening price of the stock on a particular day
Close	Numerical - Closing price of the stock on a particular day
Low	Numerical -Low price of the stock on a particular day
High	Numerical -High price of the stock on a particular day
Volume	Numerical -Volume of the stock on a particular day

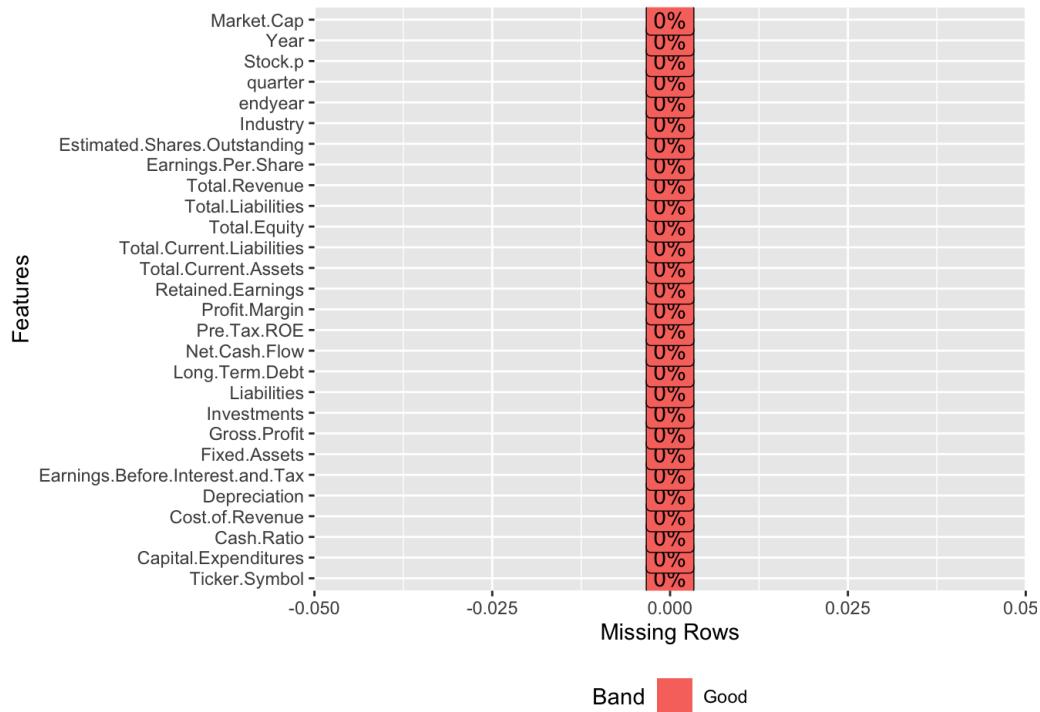
### **Data Structure and manipulation.**

- Merge the three dataframe to combine information about a stock grouped by year.
- Filter out unnecessary columns and display the final dataset which we will work on.
- Final data after merging info from three dataframes

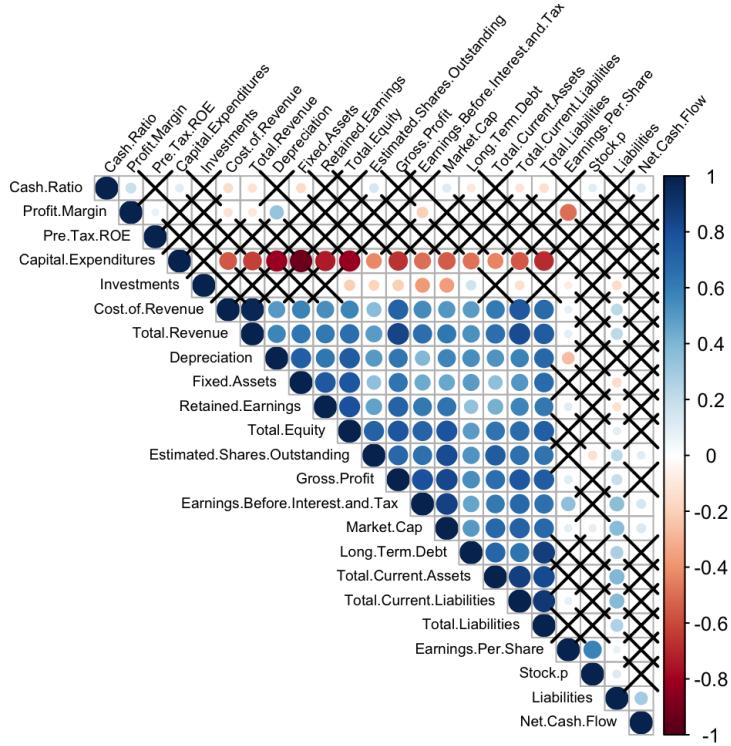
Names	Description
Ticker.Symbol	Categorical - Stock Symbol
Industry	Categorical - Industry of the Stock
endyear	Date - Year of the revenue
quarter	Date - Quarter of the revenue
Stock.p	Numerical - Mean of Close price
Capital.Expenditures	Numerical - Technical Indicator
Cash.Ratio	Numerical - Technical Indicator
Cost.of.Revenue	Numerical - Technical Indicator
Depreciation	Numerical - Technical Indicator
Earnings.Before.Interest.and.Tax	Numerical - Technical Indicator
Effect.of.Exchange.Rate	Numerical - Technical Indicator
Fixed.Assets	Numerical - Technical Indicator
Gross.Profit	Numerical - Technical Indicator
Investments	Numerical - Technical Indicator
Liabilities	Numerical - Technical Indicator
Long.Term.Debt	Numerical - Technical Indicator
Long.Term.Investments	Numerical - Technical Indicator
Net.Cash.Flow	Numerical - Technical Indicator
Net.Income	Numerical - Technical Indicator
Pre.Tax.ROE	Numerical - Technical Indicator
Profit.Margin	Numerical - Technical Indicator
Retained.Earnings	Numerical - Technical Indicator
Total.Current.Assets	Numerical - Technical Indicator
Total.Current.Liabilities	Numerical - Technical Indicator
Total.Equity	Numerical - Technical Indicator
Total.Liabilities	Numerical - Technical Indicator
Total.Revenue	Numerical - Technical Indicator
Earnings.Per.Share	Numerical - Technical Indicator
Estimated.Shares.Outstanding	Numerical - Technical Indicator

## Data Science Analysis

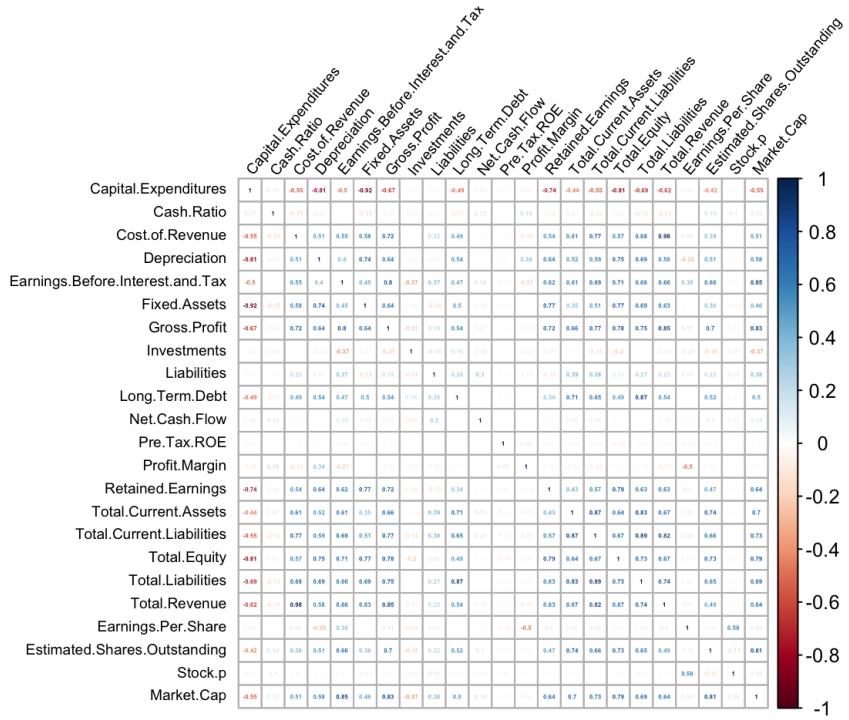
- Plot the percentage of missing data for each attribute.



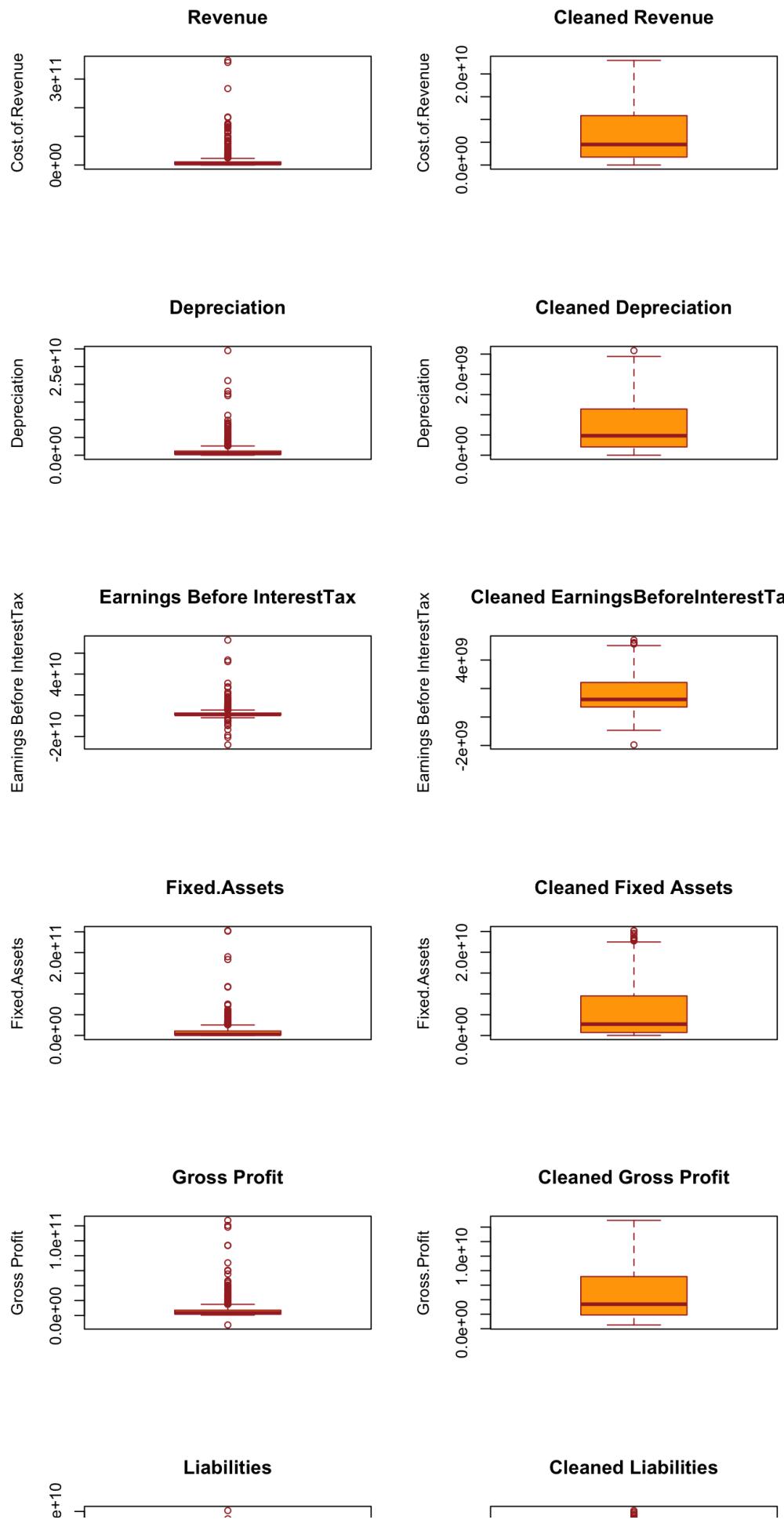
- Correlation plot with correlation coefficient before data clean up

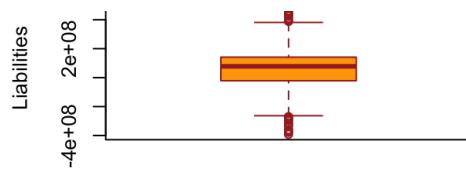
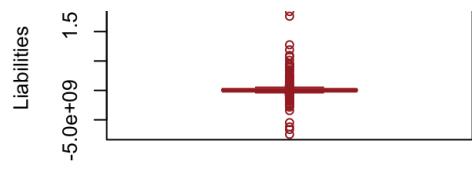


Corrplot With Correlation Coefficient

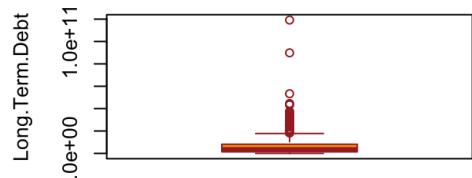


- Remove outliers of the features and replace them with the mean of the value.
- Replacing the missing values with the mean / median / mode is a best way of treating missing values. + Depending on the context, like if the variation is low or if the variable has low leverage over the response, such a rough approximation is acceptable and could possibly give satisfactory results.

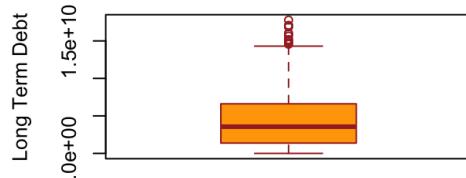




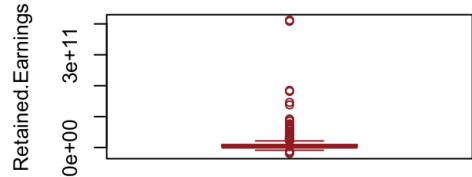
**Long.Term.Debt**



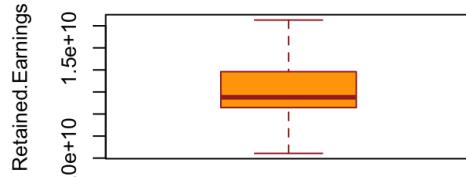
**Cleaned Long Term Debt**

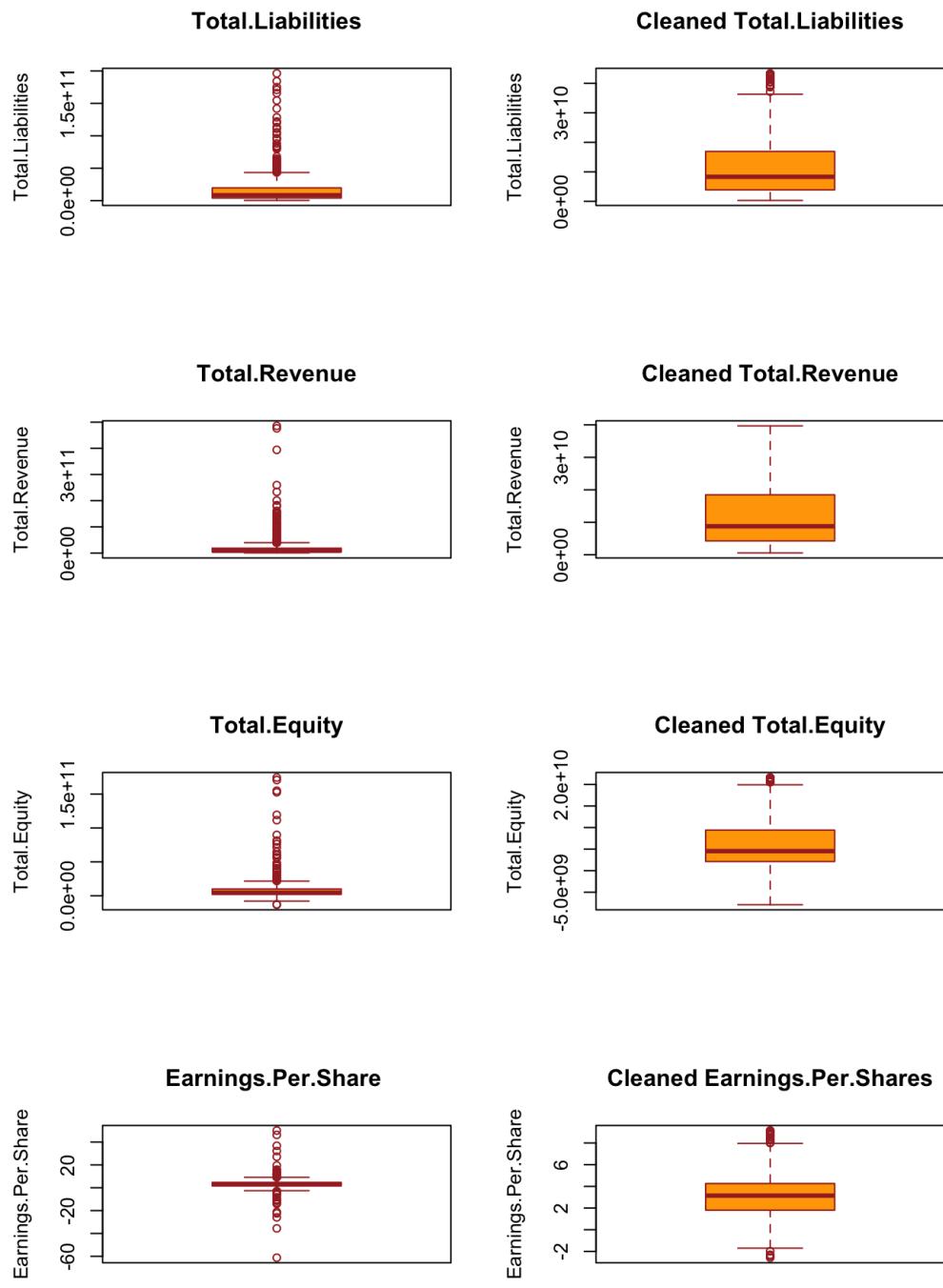


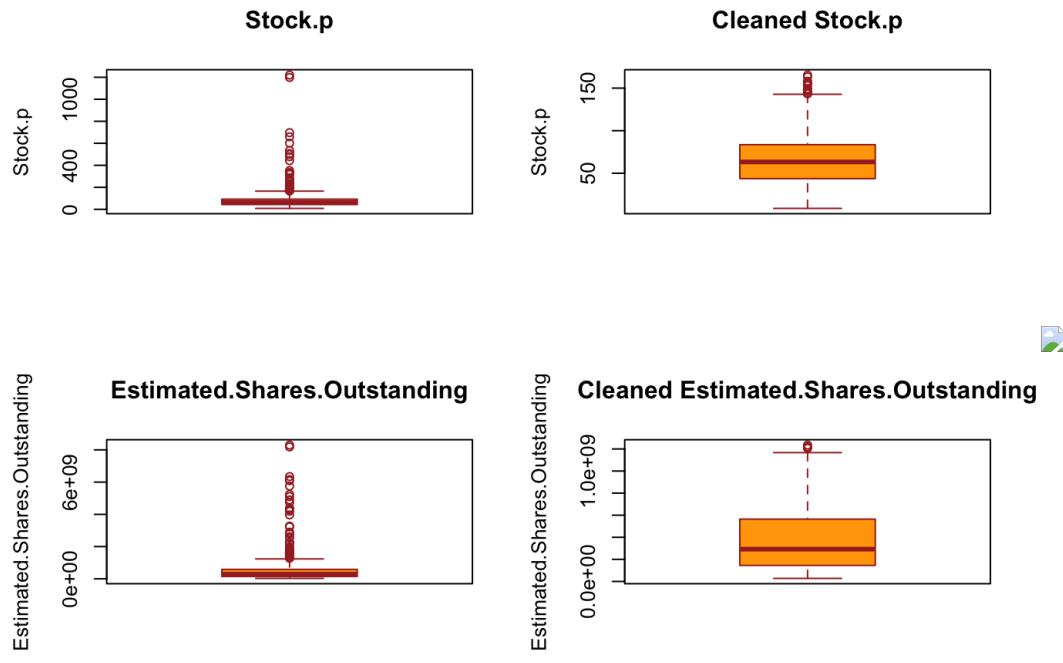
**Retained.Earnings**



**Cleaned Retained.Earnings**

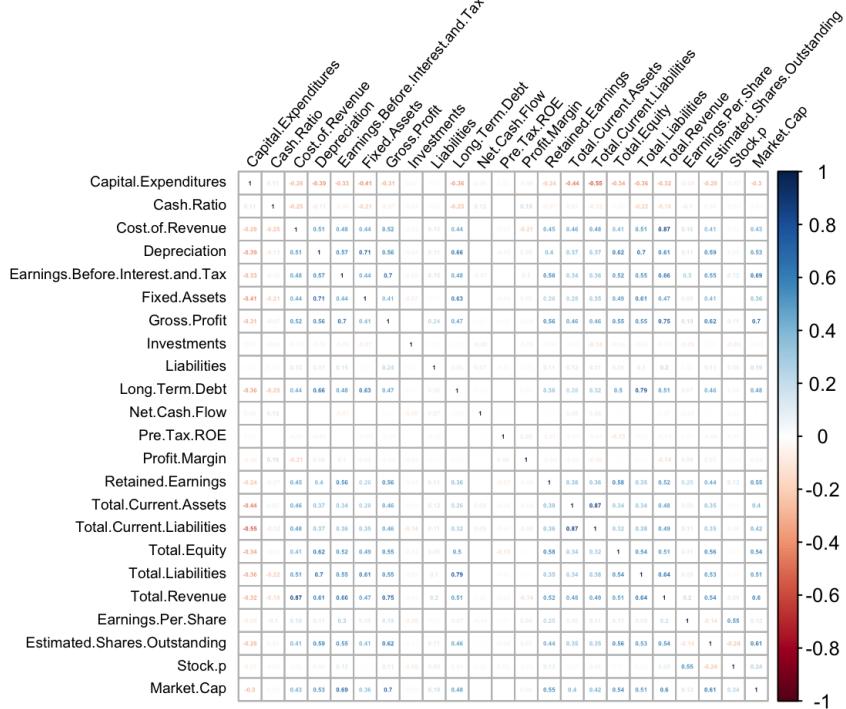






- Correlation plot with correlation coefficient after data clean up

### Correlation of Variables



### Numerical Summaries

- Frequency distribution, calculate relevant proportions, and contingency tables for categorical variable marketCapIndicator (Low/Medium/High) & industry.

Frequency Table for Industry:

Consumer Discretionary	120	Consumer Staples	50	Energy	58
Financials	10	Health Care	74	Industrials	120
Information Technology	108	Materials	44	Real Estate	14
Utilities	42				

Frequency Table for MarketCapIndiactor:

high	low	medium
292	301	47

Contingency Table for Industry/ MarketCapIndiactor:

	high	low	medium
Consumer Discretionary	46	67	7
Consumer Staples	28	18	4
Energy	32	23	3
Financials	3	4	3
Health Care	45	23	6
Industrials	47	65	8
Information Technology	49	52	7
Materials	17	24	3
Real Estate	6	6	2
Utilities	19	19	4

Proportions Table for Industry/ MarketCapIndiactor:

	high	low	medium
Consumer Discretionary	0.0718750	0.1046875	0.0109375
Consumer Staples	0.0437500	0.0281250	0.0062500
Energy	0.0500000	0.0359375	0.0046875
Financials	0.0046875	0.0062500	0.0046875
Health Care	0.0703125	0.0359375	0.0093750
Industrials	0.0734375	0.1015625	0.0125000
Information Technology	0.0765625	0.0812500	0.0109375
Materials	0.0265625	0.0375000	0.0046875
Real Estate	0.0093750	0.0093750	0.0031250
Utilities	0.0296875	0.0296875	0.0062500

Summary MarketCapIndiactor

Length	Class	Mode
640	character	character

Summary Industry

Consumer Discretionary	120	Consumer Staples	50	Energy	58
Financials	10	Health Care	74	Industrials	120
Information Technology	108	Materials	44	Real Estate	14
Utilities	42				

#### Numerical Summary of Market Cap

```
Min. 1st Qu. Median Mean 3rd Qu. Max.
2.609e+09 1.030e+10 1.678e+10 3.534e+10 3.393e+10 6.907e+11
```

#### Numerical Summary of Market Cap grouped by Industry

# A tibble: 10 x 6	Industry	sum	mean	min	max
	<fct>	<dbl>	<dbl>	<dbl>	<dbl>
1	Consumer Discretionary	3.33e12	27755715066.	4.62e9	2.23e11
2	Consumer Staples	2.74e12	54777333131.	4.69e9	2.53e11
3	Energy	2.44e12	42025301685.	4.57e9	4.16e11
4	Financials	1.55e11	15499103508.	7.49e9	2.69e10
5	Health Care	3.31e12	44710159546.	3.22e9	2.08e11
6	Industrials	3.04e12	25355115660.	4.07e9	1.01e11
7	Information Technology	5.52e12	51151898108.	2.61e9	6.91e11
8	Materials	9.05e11	20569895953.	4.04e9	5.99e10
9	Real Estate	3.33e11	23773975192.	5.95e9	4.93e10
10	Utilities	8.41e11	20019656246.	3.28e9	5.23e10

- Does skewness deviate a lot from 1 ?
  - Skewness for Industry is left skewed. It has longer tail on the left side of distribution.
  - Skewness for MarketCapIndiactor is right skewed. It has longer tail on the right side of distribution.
  - Skewness for Market.Cap is right skewed. It has longer tail on the right side of distribution.
  - Skewness for Total.Revenue is right skewed. It has longer tail on the right side of distribution.
  - Skewness for Gross.Profit is right skewed. It has longer tail on the right side of distribution.

```
"skewness of Industry -0.051643"
```

```
"skewness of MarketCapIndiactor 0.476626"
```

```
"skewness of Market.Cap 5.468953"
```

```
"skewness of Total.Revenue 6.125020"
```

```
"skewness of Gross.Profit 5.369591"
```

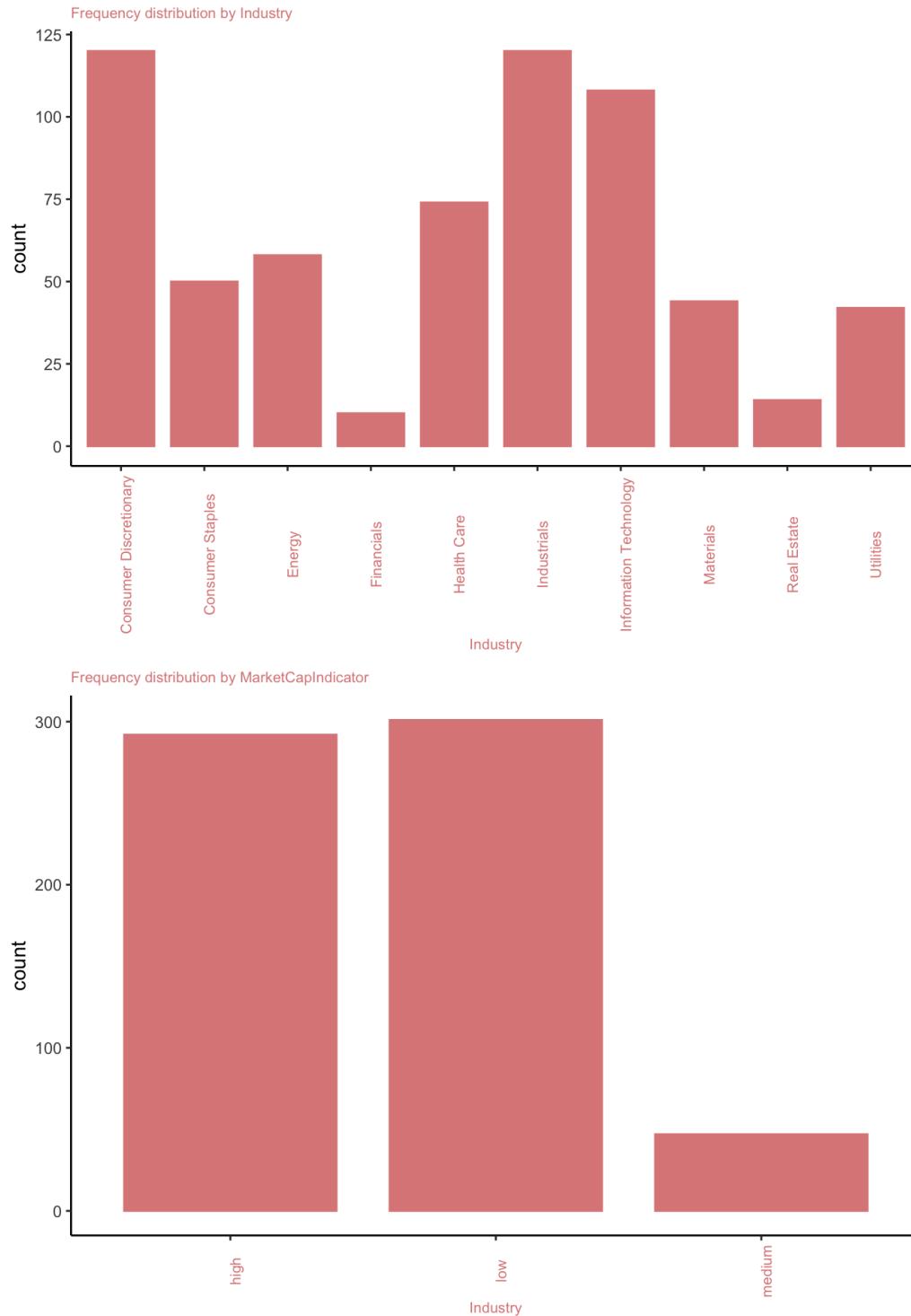
- Shapiro Wilkes Test for Normality.
  - Industry -> p-value < 8.458e-09 < 0.05 = not normally distributed

```
Shapiro-Wilk normality test
```

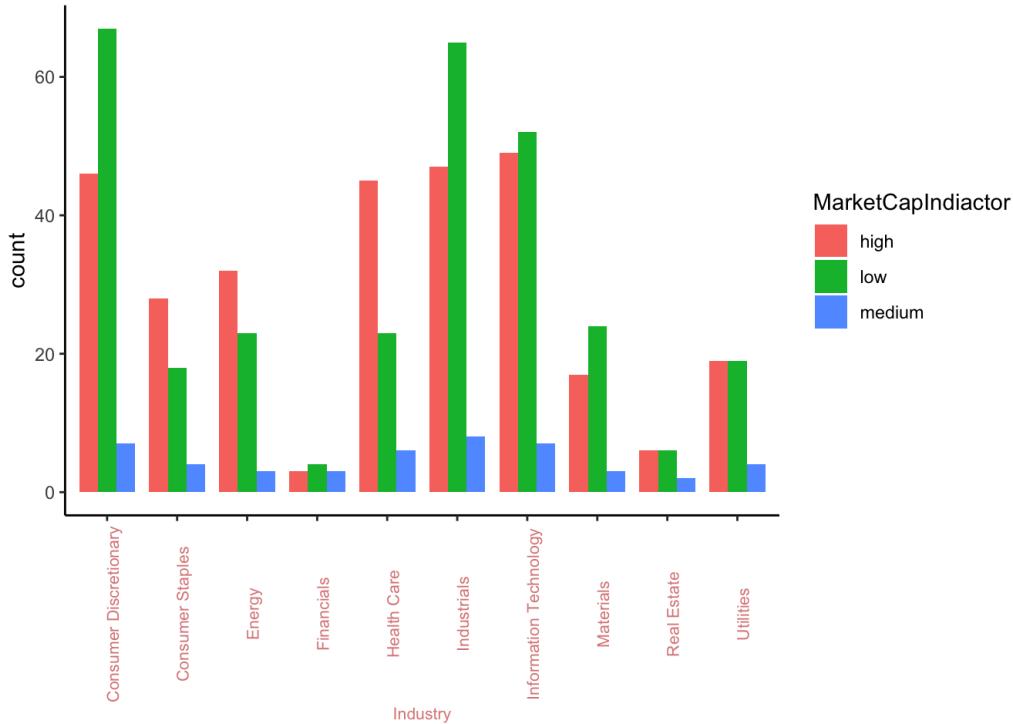
```
data: as.numeric(as.factor(s$Industry))[0:200]
W = 0.90934, p-value = 1.033e-09
```

###Graphical Summaries

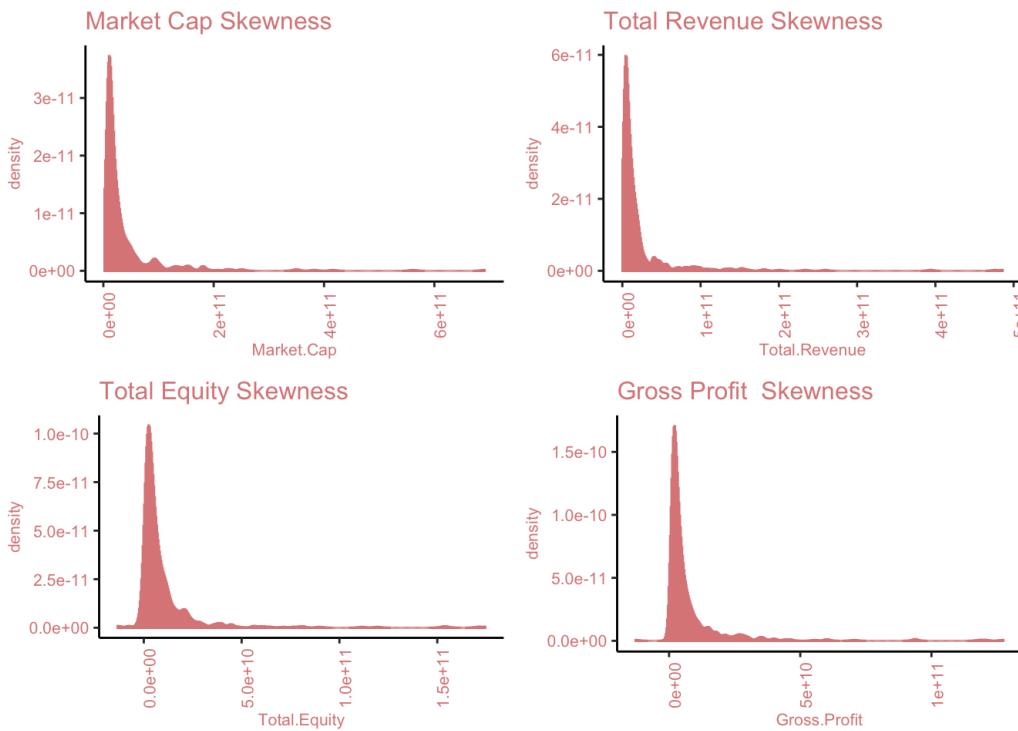
- Graphical Summary of categorical variable Industry & marketCapIndiactor



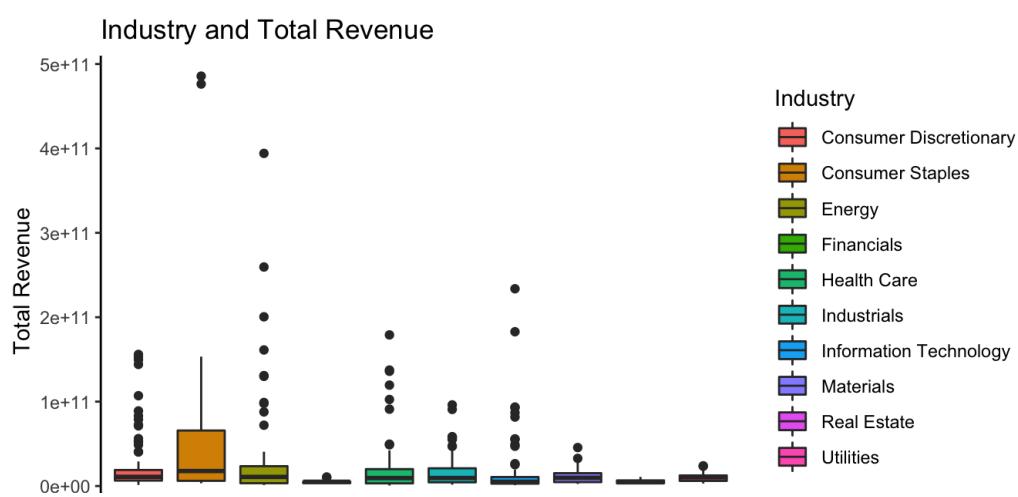
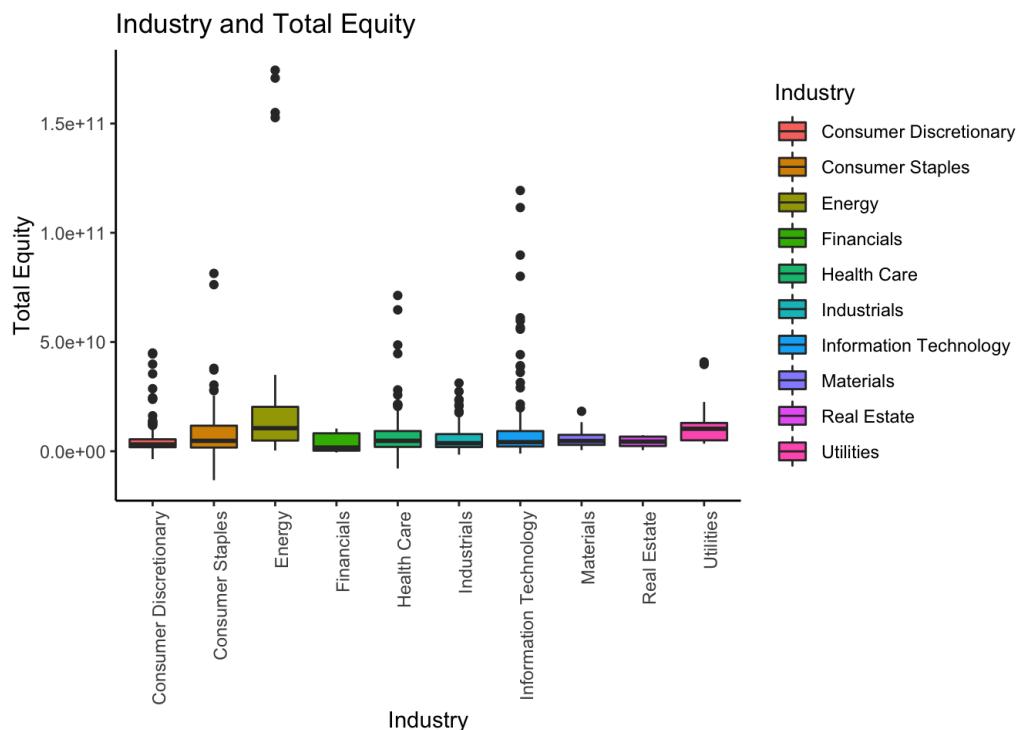
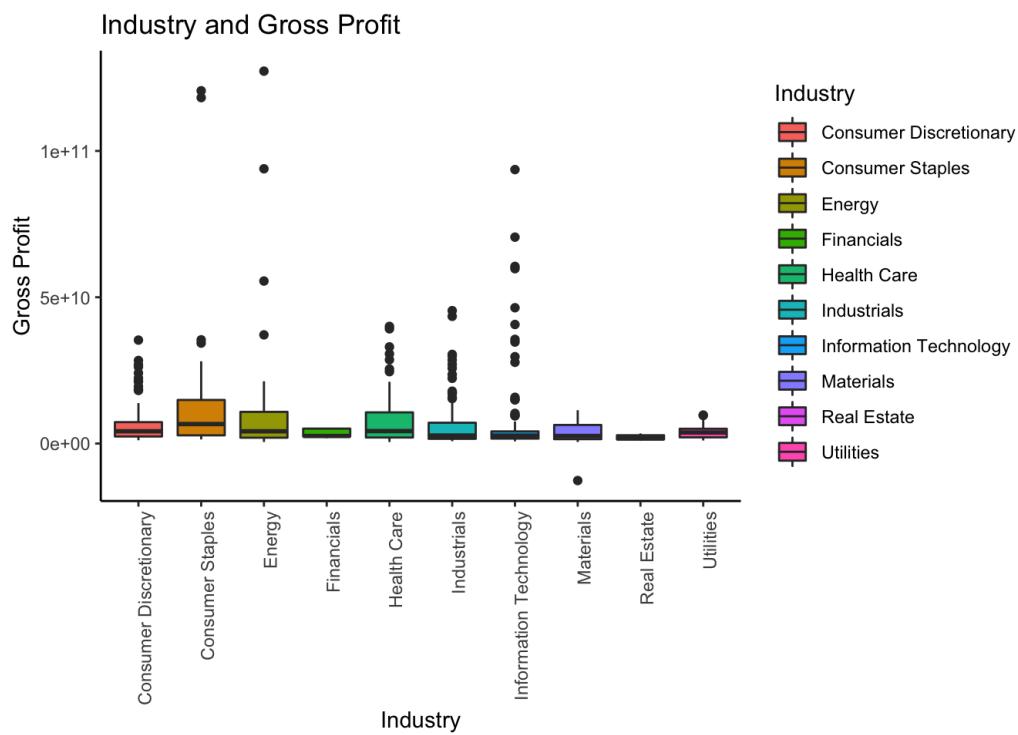
- Graphical Summary of categorical variable Industry grouped by MarketCapIndiactor

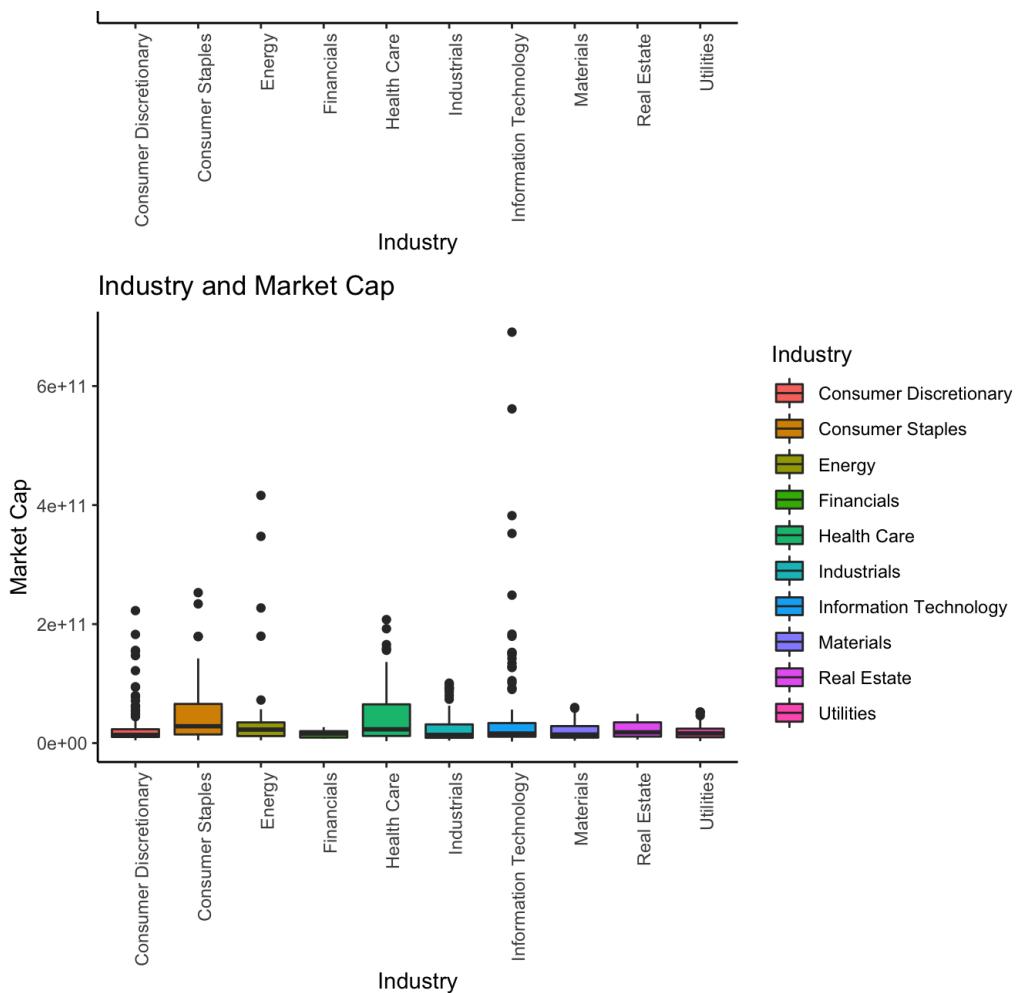


- Graphical summary of numerical variable Market.Cap, Total.Revenue, Total.Equity, Gross.Profit



- Graphical summary of numerical variable Market.Cap, Total.Revenue, Total.Equity, Gross.Profit grouped by categorical variable Industry



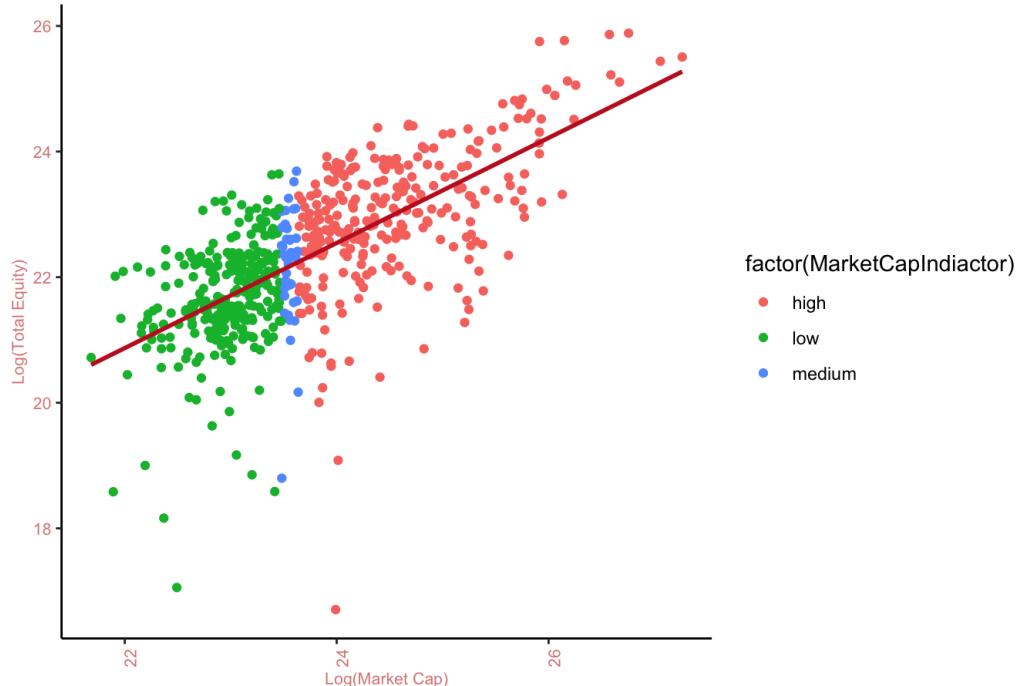


## Correlation and Regression

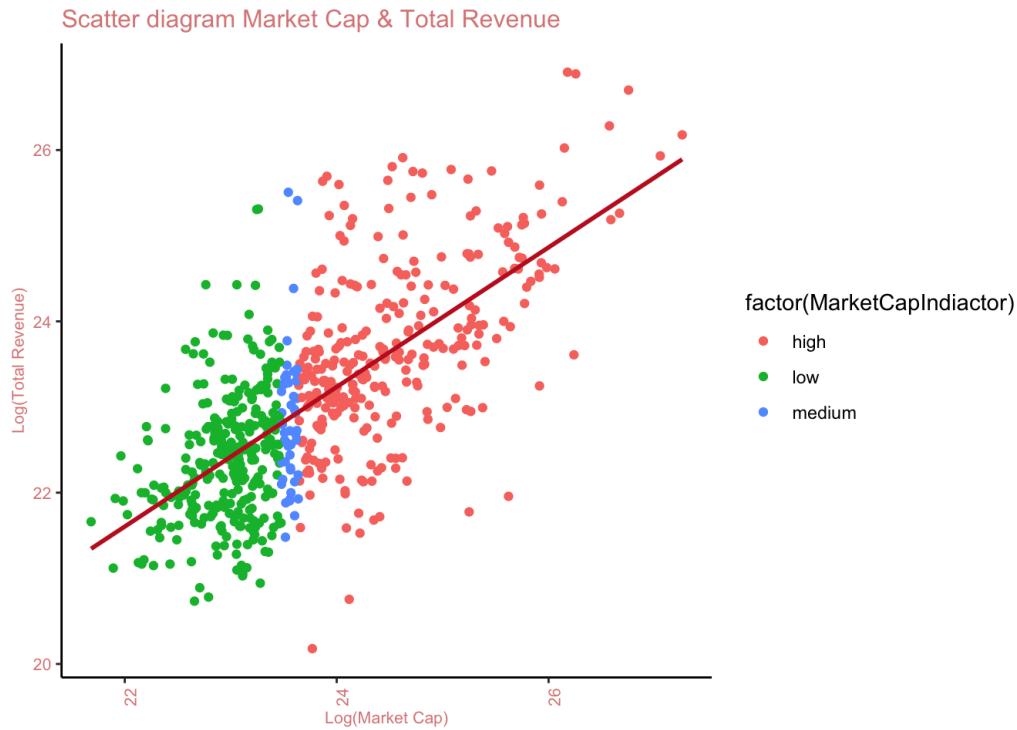
- Scatter Plot for Market.Cap & Total.Equity.

```
'data.frame': 640 obs. of 29 variables:
 $ Ticker.Symbol           : Factor w/ 320 levels "AAL", "AAPL", "ABBV", ... : 1 1 2 2 3 3 4 4 5 5 ...
 $ Capital.Expenditures    : num -5.31e+09 -6.15e+09 -9.57e+09 -1.12e+10 -6.12e+08 ...
 $ Cash.Ratio               : num 60 51 40 52 74 77 10 10 43 67 ...
 $ Cost.of.Revenue          : num 1.56e+10 1.11e+10 1.12e+11 1.40e+11 4.43e+09 ...
 $ Depreciation              : num 1.34e+09 1.49e+09 7.95e+09 1.13e+10 7.86e+08 ...
 $ Earnings.Before.Interest.and.Tax: num 4.10e+09 5.50e+09 5.35e+10 7.25e+10 2.76e+09 ...
 $ Fixed.Assets             : num 2.31e+10 2.75e+10 2.06e+10 2.25e+10 2.48e+09 ...
 $ Gross.Profit              : num 2.70e+10 2.99e+10 7.05e+10 9.36e+10 1.55e+10 ...
 $ Investments                : num 1.80e+09 4.43e+08 -9.03e+09 -4.44e+10 3.08e+08 ...
 $ Liabilities                : num -1.03e+09 -6.33e+08 1.34e+10 1.52e+10 -1.93e+08 ...
 $ Long.Term.Debt            : num 1.60e+10 1.83e+10 2.90e+10 5.33e+10 1.05e+10 ...
 $ Net.Cash.Flow              : num -1.46e+08 -6.04e+08 -4.15e+08 7.28e+09 -1.25e+09 ...
 $ Pre.Tax.ROE                 : num 159 82 48 61 136 168 34 44 12 15 ...
 $ Profit.Margin              : num 7 19 22 23 9 23 0 0 11 22 ...
 $ Retained.Earnings           : num -8.56e+09 -1.23e+09 8.72e+10 9.23e+10 5.35e+08 ...
 $ Total.Current.Assets       : num 1.18e+10 9.98e+09 6.85e+10 8.94e+10 1.61e+10 ...
 $ Total.Current.Liabilities  : num 1.34e+10 1.36e+10 6.34e+10 8.06e+10 1.14e+10 ...
 $ Total.Equity                : num 2.02e+09 5.64e+09 1.12e+11 1.19e+11 1.74e+09 ...
 $ Total.Liabilities           : num 4.12e+10 4.28e+10 1.20e+11 1.71e+11 2.58e+10 ...
 $ Total.Revenue                : num 4.26e+10 4.10e+10 1.83e+11 2.34e+11 2.00e+10 ...
 $ Earnings.Per.Share           : num 4.02 11.39 6.49 9.28 1.11 ...
 $ Estimated.Shares.Outstanding: num 7.17e+08 6.68e+08 6.09e+09 5.75e+09 1.60e+09 ...
 $ Industry                     : Factor w/ 10 levels "Consumer Discretionary", ... : 6 6 7 7 5 5 5 5 5 5 5
 ...
 $ endyear                     : Factor w/ 2 levels "2014", "2015": 1 2 1 2 1 2 1 2 1 2 ...
 $ quarter                      : Factor w/ 4 levels "1", "2", "3", "4": 4 4 3 3 4 4 3 3 4 4 ...
 $ Stock.p                       : num 38.9 45.1 92.3 120 55.5 ...
 $ Year                          : Factor w/ 2 levels "2014", "2015": 1 2 1 2 1 2 1 2 1 2 ...
 $ Market.Cap                    : num 2.79e+10 3.01e+10 5.62e+11 6.91e+11 8.86e+10 ...
 $ MarketCapIndiactor           : chr "high" "high" "high" "high" ...
```

Scatter diagram Market Cap & Total Equity



- Scatter Plot for Market.Cap & Total.Revenue



- Statistical Hypothesis Testing can be categorized into two types as below:

- t-test command prints out the p-value for the data. The p-value can then be readily compared to the significance level. The null hypothesis is then rejected if the p-value is less than the significance level.

```
Welch Two Sample t-test
```

```
data: visualization_data$Market.Cap and visualization_data$Total.Revenue
t = 4.908, df = 1179.8, p-value = 1.049e-06
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 8411277703 19614850898
sample estimates:
 mean of x   mean of y
 35337997368 21324933067
```

```
"P value for Market cap & Total revenue 0.000001"
```

```
"Group Means for Market cap & Total revenue 35337997367.965248"
"Group Means for Market cap & Total revenue 21324933067.187500"
```

```
"Confidence interval for difference for Market cap & Total revenue 8411277703.442698"
"Confidence interval for difference for Market cap & Total revenue 19614850898.112801"
```

```
"Confidence level for Market cap & Total revenue 0.950000"
```

## Inferential Statistics

### Hypothesis Test, Mean Change Price

- We want to do a hypothesis about the daily change of price of stock trading. Price may be volatile for stocks which are trending, but it might also be stable over the long-haul.

- Testing to see if daily price change has changed would make sense if we wanted to evaluate stability over two time frames. For example, if we wanted to do a pre-post analysis after a sales / marketing event, then we would test two samples (one before and one after).
- Test is two-tailed, meaning we want to know if price increased or decreased.
- Once again, we have a very small p-value (near zero), meaning we need to reject the null hypothesis. Mean daily price change is not equal to the baseline price change. Obviously, changing the alpha level makes no difference, because p is so close to zero.

#### One Sample t-test

```
data: DataSetGoogle$Price2012
t = -33.718, df = 501, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 6.241242
95 percent confidence interval:
-0.1912902 0.5170744
sample estimates:
mean of x
0.1628921
```

## Paired Tests

- To conduct a matched t-test, we simply need to compare prices of google between 2012 & 2014.
  - From the results of the F test, we can see that the null hypothesis that the variances are equal is rejected, because the p-value is below .05. This means we should use the unequal variance t-test (also known as the Welch.)
  - The results of this test are also clear. Reject the null hypothesis that that years 2012 adjusted closing prices are less than or equal to years 2014. That is nice to know. We would definitely want to investigate further, but we would surmise that if a company invested five years ago in a long-term position, they would not have greater gains!

#### F test to compare two variances

```
data: DataSetGoogle$Price2012 and DataSetGoogle$Price2014
F = 0.32317, num df = 501, denom df = 501, p-value < 2.2e-16
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
0.2711993 0.3851034
sample estimates:
ratio of variances
0.3231715
```

#### Two Sample t-test

```
data: DataSetGoogle$Price2012 and DataSetGoogle$Price2014
t = 1.6032, df = 1002, p-value = 0.1092
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
-0.1310177 1.3005862
sample estimates:
mean of x mean of y
0.1628921 -0.4218921
```

```

Welch Two Sample t-test

data: DataSetGoogle$Price2012 and DataSetGoogle$Price2014
t = 1.6032, df = 794.2, p-value = 0.1093
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
-0.1312443 1.3008127
sample estimates:
mean of x mean of y
0.1628921 -0.4218921

```

## Independent Tests

- The independent-samples test can take one of three forms, depending on the structure of your data and the equality of their variances. The general form of the test is `t.test(y1, y2, paired=FALSE)`.
  - Here for Independent test, we will conduct price comparison between Amazon & Google.
  - Test for equality of variances in the data prior to running an independent-samples t-test, can be done by `var.test()` function:

```

Two Sample t-test

data: pairedtestdataset$PriceGoogle and pairedtestdataset$PriceAmazon
t = -0.41406, df = 1002, p-value = 0.6789
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
-0.3699250 0.2410143
sample estimates:
mean of x mean of y
-0.12885399 -0.06439866

```

```

Welch Two Sample t-test

data: pairedtestdataset$PriceGoogle and pairedtestdataset$PriceAmazon
t = -0.41406, df = 828.45, p-value = 0.6789
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
-0.3700024 0.2410917
sample estimates:
mean of x mean of y
-0.12885399 -0.06439866

```

```

F test to compare two variances

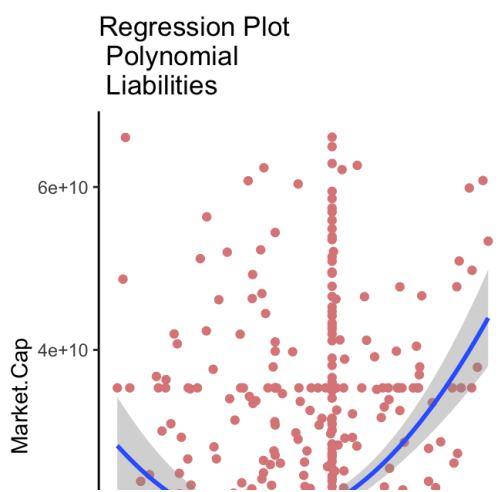
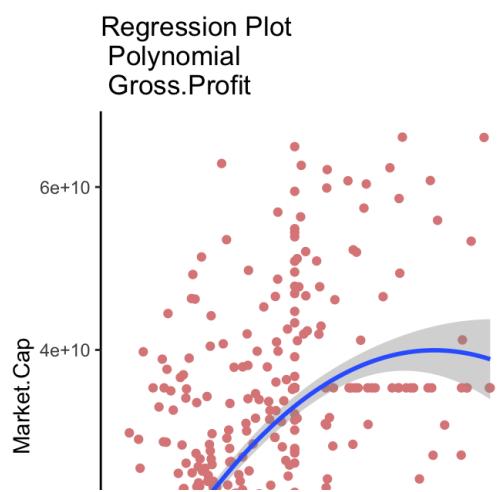
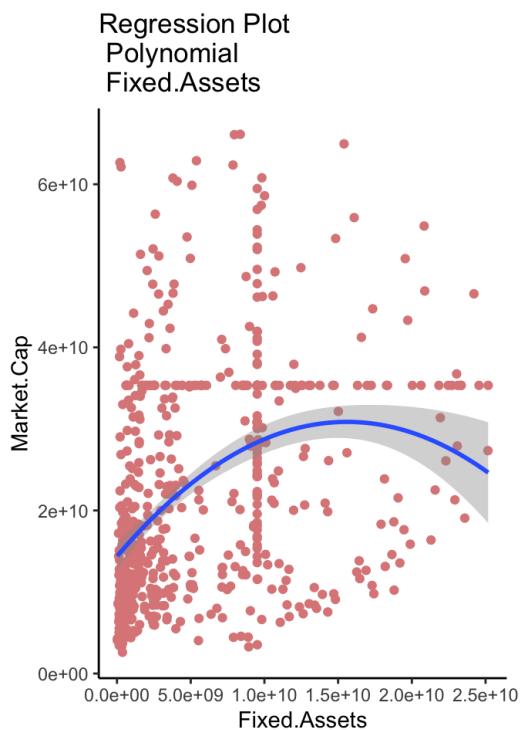
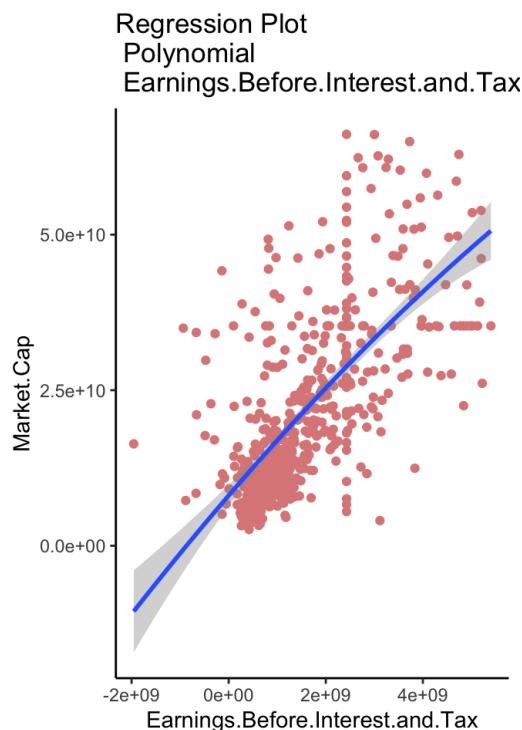
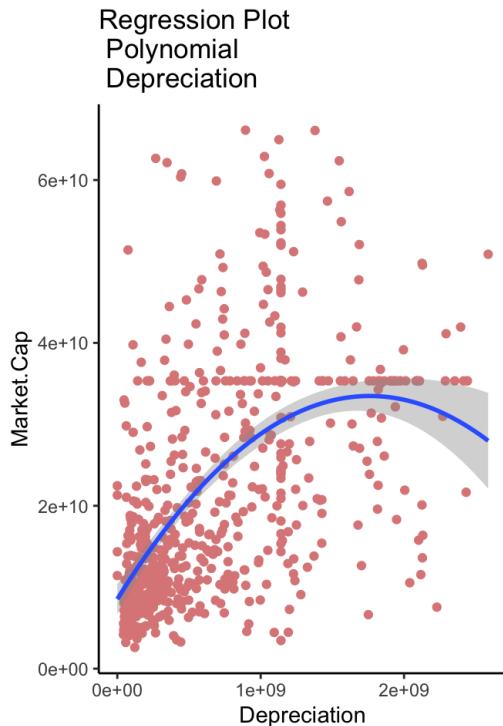
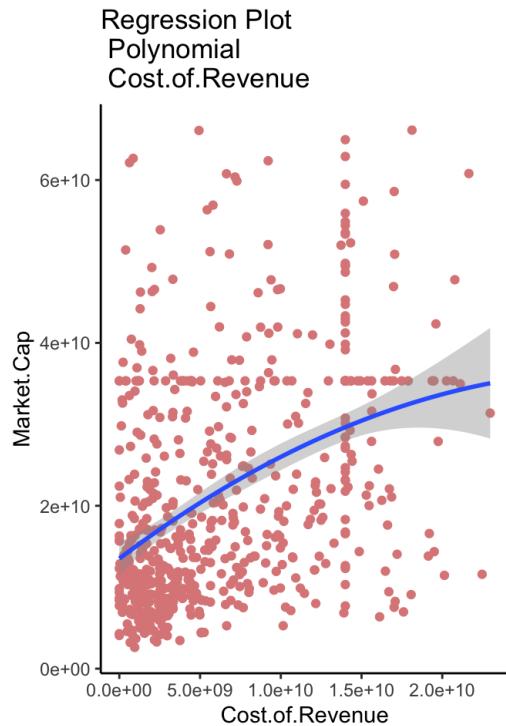
data: pairedtestdataset$PriceGoogle and pairedtestdataset$PriceAmazon
F = 2.688, num df = 501, denom df = 501, p-value < 2.2e-16
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
2.255708 3.203109
sample estimates:
ratio of variances
2.687988

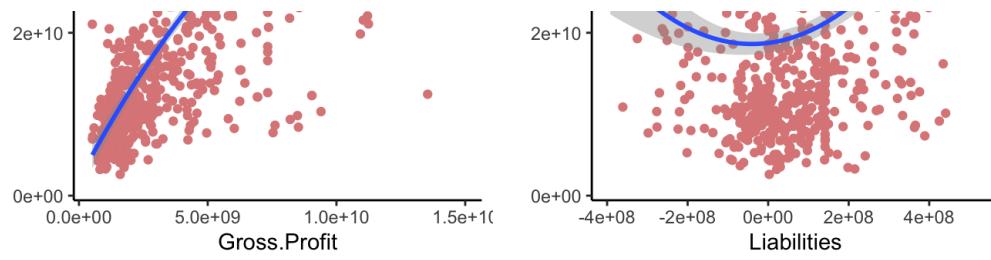
```

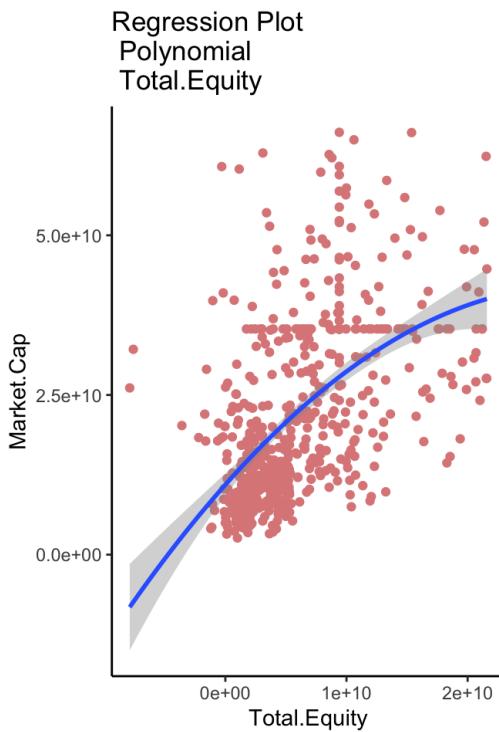
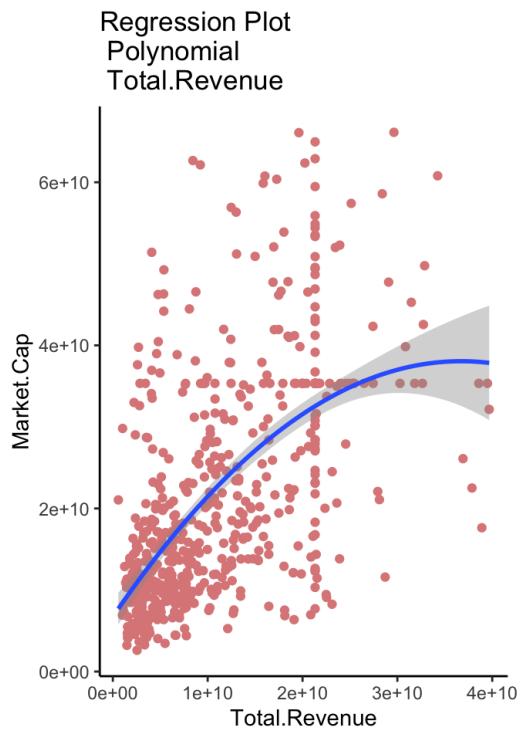
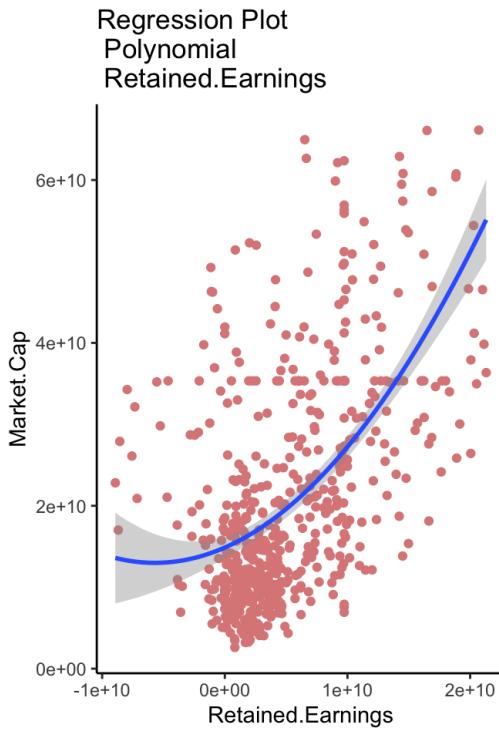
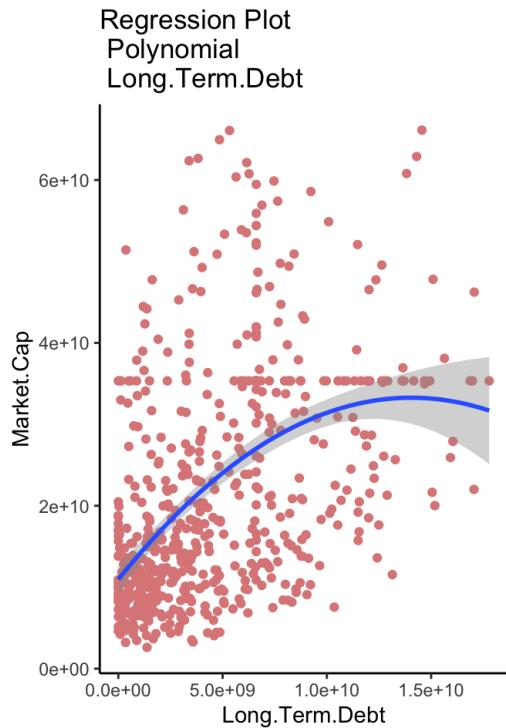
## Regression Diagnostic Plots

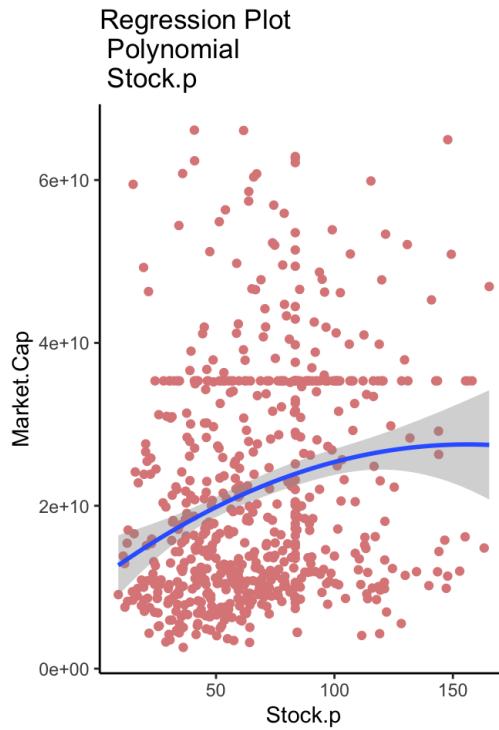
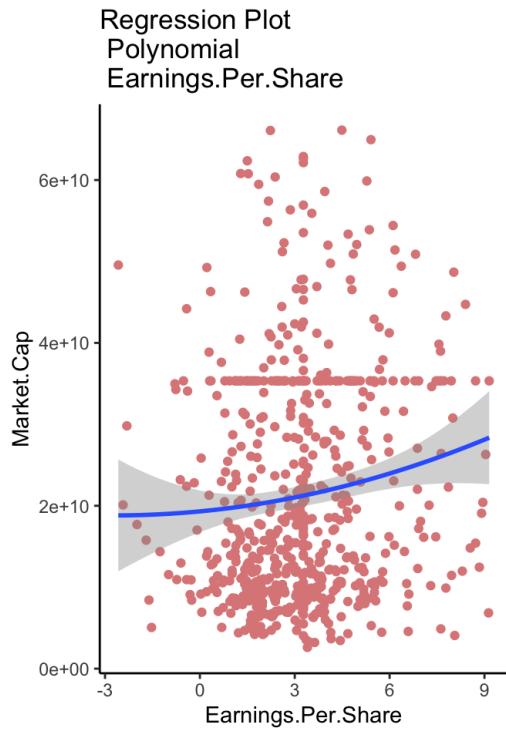
- Regression plot for Market.Cap with other features



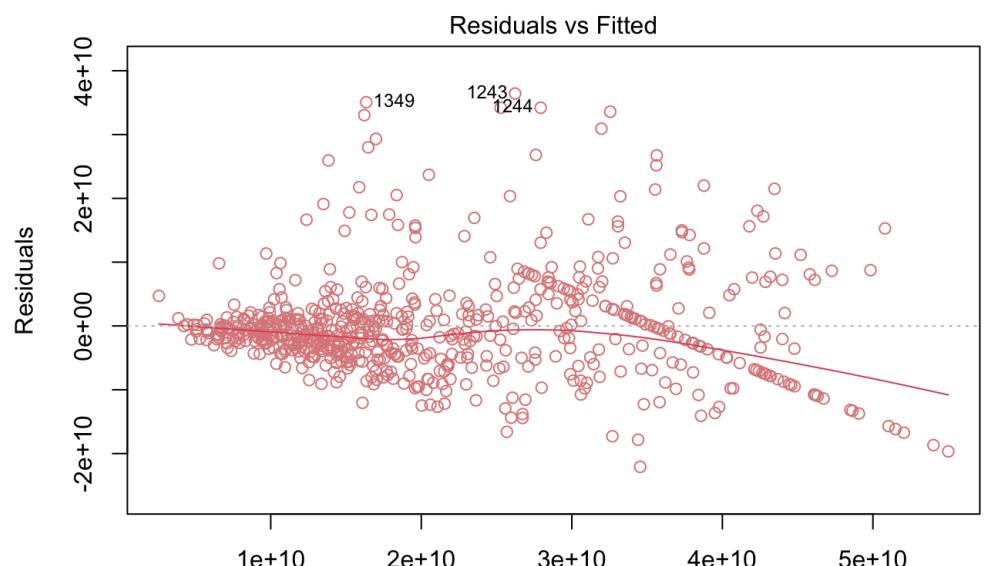




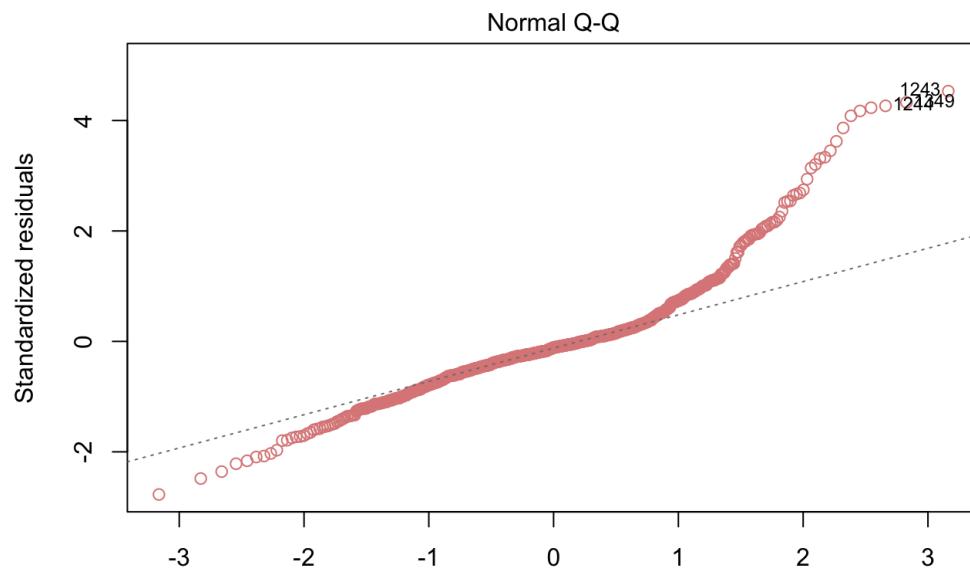




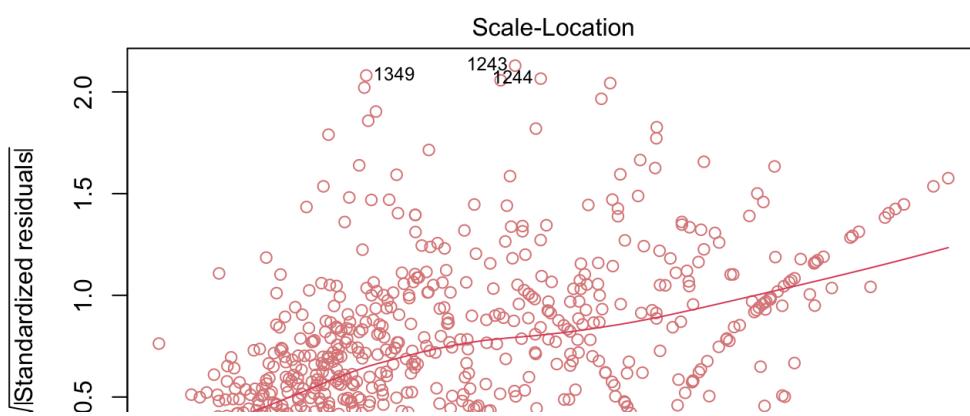
- The diagnostic plots of availability with price show residuals in four different ways:
- Residuals vs Fitted: This diagnostic plot is an indicator of the Linearity or Non Linearity of the relationship. If there is no perceivable pattern around the central horizontal curve then the relationship is linear.
- Normal Q-Q: This diagnostic plot ascertains whether the residuals are normally distributed. If the residuals are aligned to the central diagonal then the residuals follow a straight line.
- Scale-Location: This diagnostic is to evaluate the homogeneity of variance of the residuals .If the residuals are spread uniformly around the central line then the residuals are homoscedastic.
- Residuals vs Leverage. Used to identify influential cases, that is extreme values that might influence the regression results when included or excluded from the analysis.

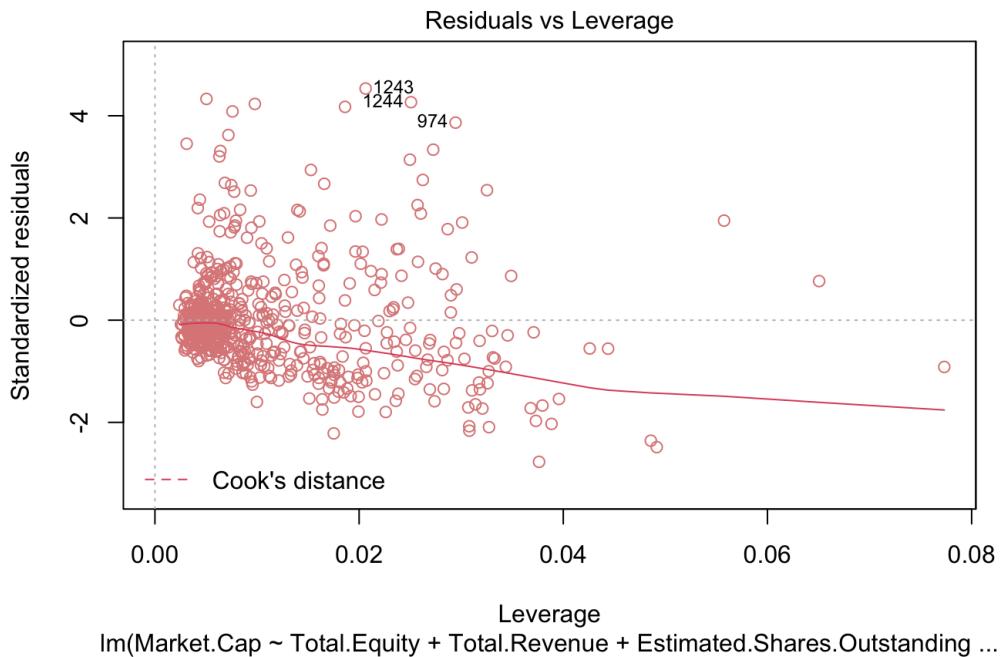
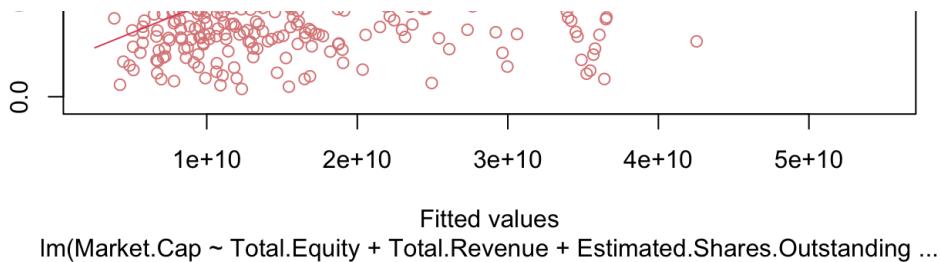


Im(Market.Cap ~ Total.Equity + Total.Revenue + Estimated.Shares.Outstanding ...)



Im(Market.Cap ~ Total.Equity + Total.Revenue + Estimated.Shares.Outstanding ...)





## MultiCollinearity

- Collinearity implies two variables are near perfect linear combinations of one another. Multicollinearity involves more than two variables. In the presence of multicollinearity, regression estimates are unstable and have high standard errors.
- Variance inflation factors measure the inflation in the variances of the parameter estimates due to collinearities that exist among the predictors.
- The general rule of thumb is that VIFs exceeding 4 warrant further investigation, while VIFs exceeding 10 are signs of serious multicollinearity requiring correction.
- In our example, the VIF score all the Total.Equity, Total.Revenue , Estimated.Shares.Outstanding, Earnings.Before.Interest.and.Tax , Gross.Profit is very high . This might be problematic.

"VIF score for all features"

Total.Equity	Total.Revenue
3.198600	3.939829
Estimated.Shares.Outstanding	Earnings.Before.Interest.and.Tax
2.732357	3.040498
Gross.Profit	Stock.p
7.184466	1.062775

## Regression- Find if Market.Cap is affected by trading technical indicators

### Linear Regression

- Objective: Check if Market.Cap is affected by "Total.Equity", "Total.Revenue", "Estimated.Shares.Outstanding", "Earnings.Before.Interest.and.Tax", "Gross.Profit", "Stock.p".
- Import the dataset, and clean the dataset for target and feature variables.
- Data slicing:
  - Dataset is split into 75 percent of training data, 25 % of test set.

- Preprocessing:
  - We scale and center the predictor & feature variables before passing to models.
- Training the Multiple Linear Regression:
  - Training the Multiple Linear Regression with metric as “RMSE”.
  - train() method is passed with repeated cross-validation resampling method for 10 number of resampling iterations repeated for 3 times. The default method for optimizing tuning parameters in train is to use a grid search. Hyperparameter Tuning using Grid Search Cross Validation
- Trained classifier results
  - We can check the result of our train() method by a print fit.LR variable.

```

Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
    Min         1Q     Median        3Q       Max
-2.233e+10 -4.448e+09 -9.596e+08  2.379e+09  3.628e+10

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 2.139e+10 2.102e+08 101.771 < 2e-16 ***
Total.Equity 9.649e+08 2.724e+08   3.542 0.000409 ***
Total.Revenue -5.398e+07 3.301e+08  -0.164 0.870121
Estimated.Shares.Outstanding 4.801e+09 3.153e+08 15.226 < 2e-16 ***
Earnings.Before.Interest.and.Tax 3.977e+09 3.150e+08 12.623 < 2e-16 ***
Gross.Profit 3.104e+09 3.696e+08   8.397 < 2e-16 ***
Stock.p      3.654e+09 2.338e+08 15.632 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.237e+09 on 1529 degrees of freedom
Multiple R-squared:  0.6548,    Adjusted R-squared:  0.6534
F-statistic: 483.3 on 6 and 1529 DF,  p-value: < 2.2e-16

```

- Prediction
  - We are ready to predict classes for our test set using predict() method, and calculate error rate and rmse. We get Multiple R-squared value as 0.99499149, and error rate as 0.2309932.

```
RMSE:  4968177423 Error:  0.2309932 R2:  0.9949914
```

- Interpretation of results
  - R-squared has the useful property that its scale is intuitive: it ranges from zero to one, with zero indicating that the proposed model does not improve prediction over the mean model, and one indicating perfect prediction. Improvement in the regression model results in proportional increases in R-squared.
  - The RMSE is the square root of the variance of the residuals. It indicates the absolute fit of the model to the data—how close the observed data points are to the model's predicted values. Whereas R-squared is a relative measure of fit, RMSE is an absolute measure of fit. Lower values of RMSE indicate better fit.
  - Since error percentage is 23, R<sup>2</sup>=0.99, RMSE is , it is safe to assume that market cap is affected by “Total.Equity”, “Total.Revenue”, “Estimated.Shares.Outstanding”, “Earnings.Before.Interest.and.Tax”, “Gross.Profit”, “Stock.p”.

## Polynomial Regression

- Polynomial regression is a special case of linear regression where we fit a polynomial equation on the data with a curvilinear relationship between the target variable and the independent variables.
- In a curvilinear relationship, the value of the target variable changes in a non-uniform manner with respect to the predictor (s).
- poly function returns or evaluates orthogonal polynomials of degree 1 to degree over the specified set of points x: these are all orthogonal to the constant polynomial of degree 0.

```

Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
    Min         1Q     Median        3Q       Max
-1.693e+10 -3.719e+09 -1.128e+09  1.712e+09  4.125e+10

Residual standard error: 7.42e+09 on 1523 degrees of freedom
Multiple R-squared:  0.721, Adjusted R-squared:  0.7188
F-statistic: 327.9 on 12 and 1523 DF,  p-value: < 2.2e-16

```

- Polynomial Regression Prediction & Accuracy

```
RMSE: 5761789130 Error: 0.2309932 R2: 0.9938427
```

## Spline Regression

- Spline is a special function defined piece-wise by polynomials. The term “spline” is used to refer to a wide class of functions that are used in applications requiring data interpolation and/or smoothing. The data may be either one-dimensional or multi-dimensional.
- Spline Regression is one of the non-parametric regression technique. In this technique the dataset is divided into bins at intervals or points which we called as knots. Also this bin has its separate fit.
- The disadvantages of the polynomial regression can be overcome by using Spline Regression. Polynomial regression only captures a certain amount of curvature in a nonlinear relationship. An alternative, and often superior, approach to modeling nonlinear relationships is to use splines.
- Splines provide a way to smoothly interpolate between fixed points, called knots. Polynomial regression is computed between knots. In other words, splines are series of polynomial segments strung together, joining at knots.
- The generic function quantile produces sample quantiles corresponding to the given probabilities.
- bs {splines} Generate the B-spline basis matrix for a polynomial spline.bs uses knots which is the internal breakpoints that define the spline.Typical values are the mean or median for one knot, quantiles for more knots.

```
'data.frame': 640 obs. of 23 variables:
 $ Capital.Expenditures      : num  -5.31e+09 -6.15e+09 -9.57e+09 -1.12e+10 -6.12e+08 ...
 $ Cash.Ratio                  : num  60 51 40 52 74 77 10 10 43 67 ...
 $ Cost.of.Revenue              : num  1.56e+10 1.11e+10 1.40e+10 1.40e+10 4.43e+09 ...
 $ Depreciation                 : num  1.34e+09 1.49e+09 1.14e+09 1.14e+09 7.86e+08 ...
 $ Earnings.Before.Interest.and.Tax: num  4.10e+09 2.43e+09 2.43e+09 2.43e+09 2.76e+09 ...
 $ Fixed.Assets                 : num  2.31e+10 9.51e+09 2.06e+10 2.25e+10 2.48e+09 ...
 $ Gross.Profit                 : num  7.33e+09 7.33e+09 7.33e+09 7.33e+09 7.33e+09 ...
 $ Investments                   : num  1.80e+09 4.43e+08 -9.03e+09 -4.44e+10 3.08e+08 ...
 $ Liabilities                   : num  1.41e+08 1.41e+08 1.41e+08 1.41e+08 -1.93e+08 ...
 $ Long.Term.Debt               : num  1.60e+10 6.61e+09 6.61e+09 6.61e+09 1.05e+10 ...
 $ Net.Cash.Flow                 : num  -1.46e+08 -6.04e+08 -4.15e+08 7.28e+09 -1.25e+09 ...
 $ Pre.Tax.ROE                   : num  159 82 48 61 136 168 34 44 12 15 ...
 $ Profit.Margin                 : num  7 19 22 23 9 23 0 0 11 22 ...
 $ Retained.Earnings              : num  -8.56e+09 -1.23e+09 9.73e+09 9.73e+09 5.35e+08 ...
 $ Total.Current.Assets          : num  1.18e+10 9.98e+09 6.85e+10 8.94e+10 1.61e+10 ...
 $ Total.Current.Liabilities     : num  1.34e+10 1.36e+10 6.34e+10 8.06e+10 1.14e+10 ...
 $ Total.Equity                   : num  2.02e+09 5.64e+09 9.41e+09 9.41e+09 1.74e+09 ...
 $ Total.Liabilities              : num  4.12e+10 4.28e+10 1.69e+10 1.69e+10 2.58e+10 ...
 $ Total.Revenue                  : num  2.13e+10 2.13e+10 2.13e+10 2.13e+10 2.00e+10 ...
 $ Earnings.Per.Share              : num  4.02 3.27 6.49 3.27 1.11 ...
 $ Estimated.Shares.Outstanding : num  7.17e+08 6.68e+08 5.64e+08 5.64e+08 5.64e+08 ...
 $ Stock.p                         : num  38.9 45.1 92.3 120 55.5 ...
 $ Market.Cap                      : num  2.79e+10 3.01e+10 3.53e+10 3.53e+10 3.53e+10 ...

```

```

Call:
lm(formula = Market.Cap ~ bs(Total.Equity, knots = knots) + bs(Total.Revenue,
  knots = knots) + bs(Estimated.Shares.Outstanding, knots = knots) +
  bs(Earnings.Before.Interest.and.Tax, knots = knots) + bs(Gross.Profit,
  knots = knots), data = splineDatasetSubTrain)

Residuals:
    Min          1Q      Median          3Q         Max
-2.800e+10 -4.215e+09 -6.977e+08  2.852e+09  3.430e+10

Residual standard error: 8.429e+09 on 1512 degrees of freedom
Multiple R-squared:  0.6425,   Adjusted R-squared:  0.637
F-statistic: 118.1 on 23 and 1512 DF,  p-value: < 2.2e-16

```

- Spline Regression Prediction & Accuracy

```
RMSE: 4626494209 Error: 0.2151068 R2: 0.8751914
```

## Penalized Cubic Regression Spline

- Cubic Regression Spline is specified by bs="cr". These have a cubic spline basis defined by a modest sized set of knots spread evenly through the covariate values. They are penalized by the conventional integrated square second derivative cubic spline penalty.

```

Family: gaussian
Link function: identity

Formula:
Market.Cap ~ s(Total.Equity, bs = "cr") + s(Total.Revenue, bs = "cr") +
  s(Earnings.Before.Interest.and.Tax, bs = "cr")

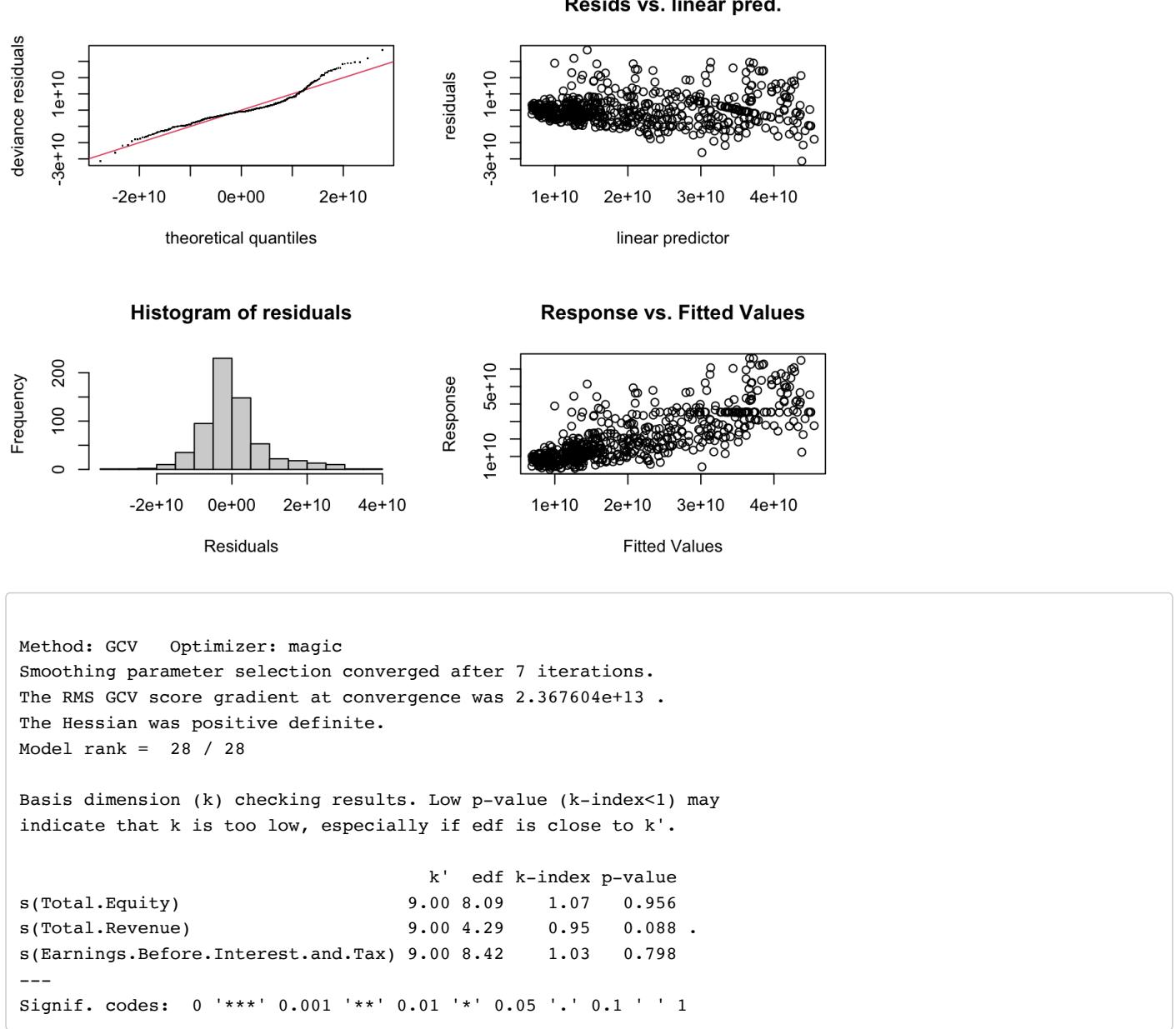
Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 2.144e+10  3.458e+08     62    <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

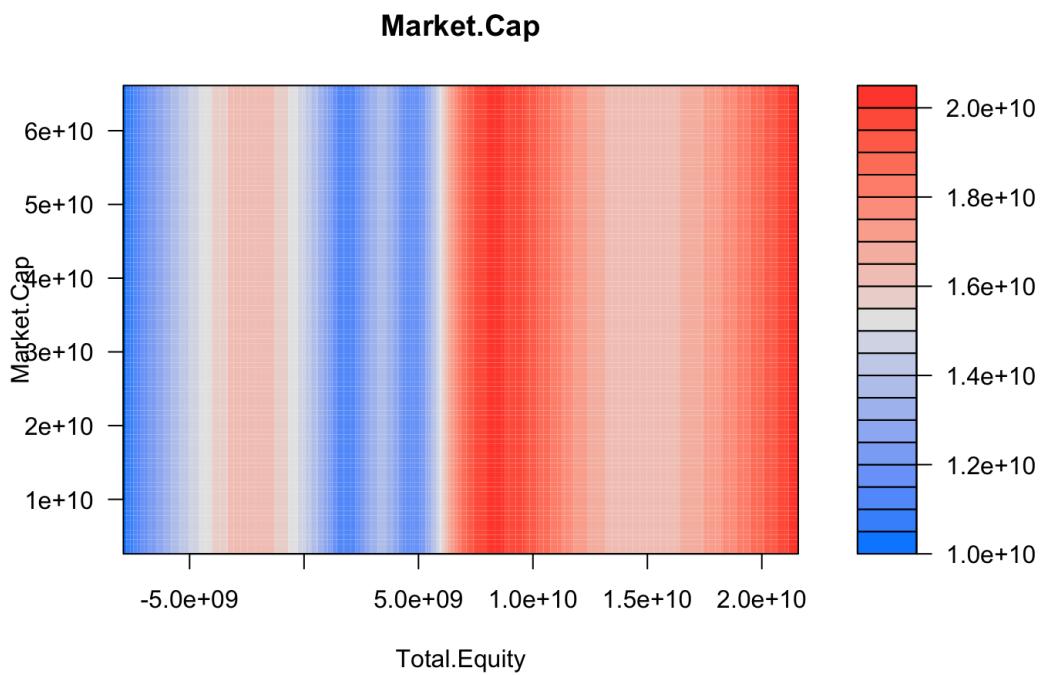
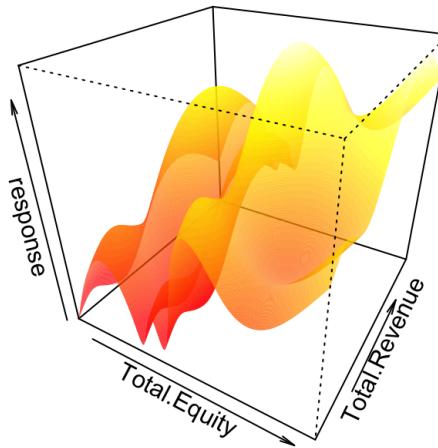
Approximate significance of smooth terms:
              edf Ref.df      F  p-value
s(Total.Equity) 8.091  8.625  6.858 < 2e-16 ***
s(Total.Revenue) 4.292  5.027  4.697 0.000331 ***
s(Earnings.Before.Interest.and.Tax) 8.418  8.829 22.214 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.607  Deviance explained =  62%
GCV = 7.9235e+19  Scale est. = 7.6536e+19  n = 640

```

- Diagnosing Penalized Cubic Regression Spline GAM model issues





- Cubic Regression Spline Prediction & Accuracy

```
RMSE: 5328801306 Error: 0.2722922 R2: 0.9062193
```

## Compare Regression Models

X.Algorithm.	X.RMSE.	X.R2.	X.Error.
Algorithm	RMSE	R2	Error
ExtremeGradinetBoosting	3988987596.3155	0.870134218809127	0.203830128687742
Cubic Regression Spline	5328801306.209320.9062193180252090	0.272292211938517	
GAM	6937151742.56324	0.651536114257158	0.354476041419414
Spline Regression	4626494208.874580.875191434305949	0.215106804884498	
PolynomialRegression	0.993842702528730.2309932149928095761789129.92617		

## Conclusion

As such, we provide evidence suggesting that stock technical indicator has influence in stock market market cap. PolynomialRegression outperforms other model with lesser error and with R<sub>2</sub>=0.994 the model completely fit.

## Generalized Linear Model

Why Use GAM? Relationships between the individual predictors and the dependent variable follow smooth patterns that can be linear or nonlinear. + Mathematically speaking, GAM is an additive modeling technique where the impact of the predictive variables is captured through smooth functions which—depending on the underlying patterns in the data—can be nonlinear. + GAM can capture common nonlinear patterns that a classic linear model would miss. + GAM framework allows us to control smoothness of the predictor functions to prevent overfitting. By controlling the wigginess of the predictor functions, we can directly tackle the bias/variance tradeoff.

```
Family: gaussian
Link function: identity

Formula:
Market.Cap ~ Total.Equity + Earnings.Before.Interest.and.Tax

Parametric coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept) 2.173e-09 5.850e-11 37.14 <2e-16 ***
Total.Equity 1.052e+00 5.873e-02 17.92 <2e-16 ***
Earnings.Before.Interest.and.Tax 8.172e+00 2.327e-01 35.11 <2e-16 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Rank: 2/3
R-sq.(adj) = 0.461 Deviance explained = 43.4%
GCV = 1.1162e+20 Scale est. = 1.1147e+20 n = 1536
```

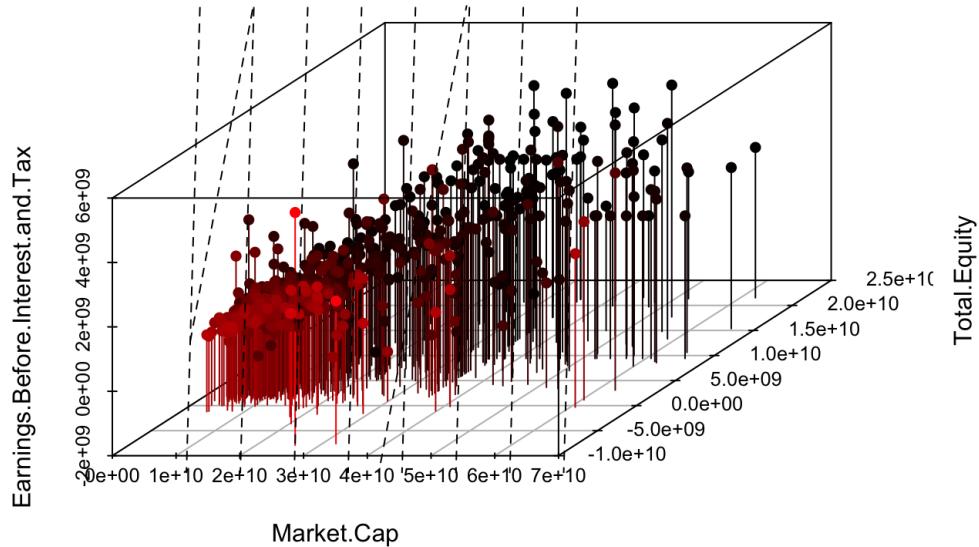
## GAM Regression Prediction & Accuracy

```
RMSE: 6937151743 Error: 0.354476 R2: 0.6515361
```

## GAM Scatter Plot 3D Visualization

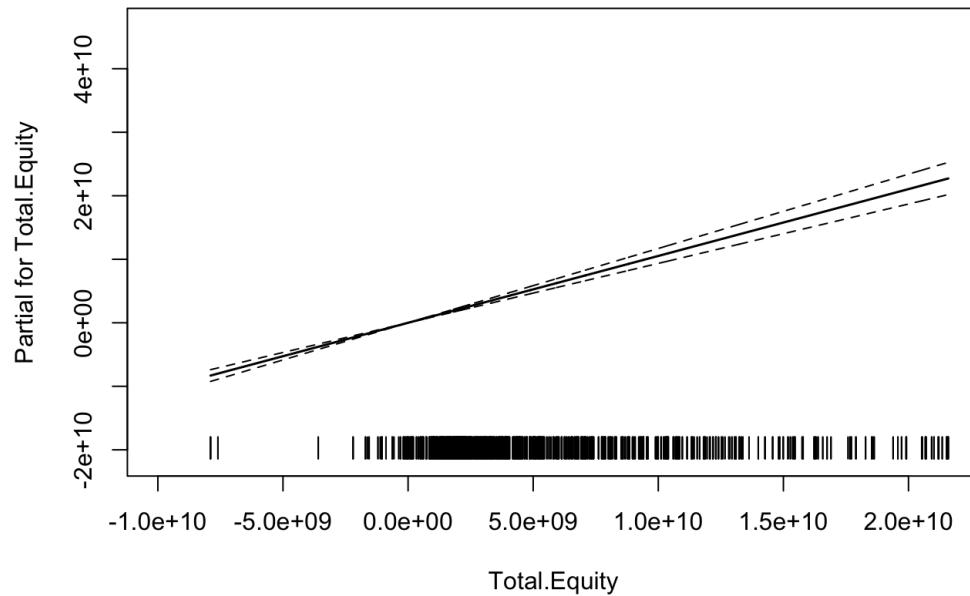
## Multi-Variable Regression

Market.Cap ~ Total.Equity + Earnings.Before.Interest.and.Tax

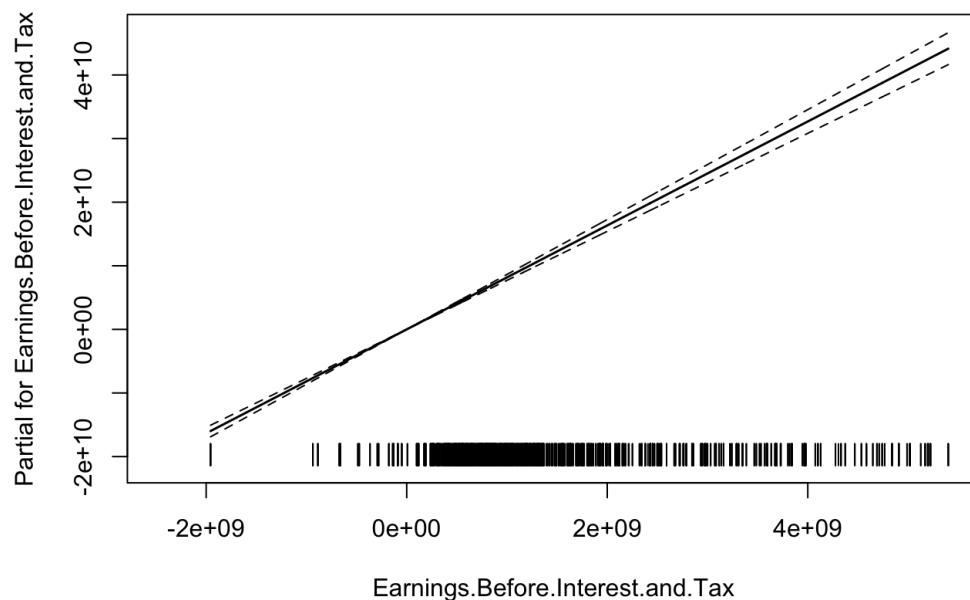


- Visualization part of gam model

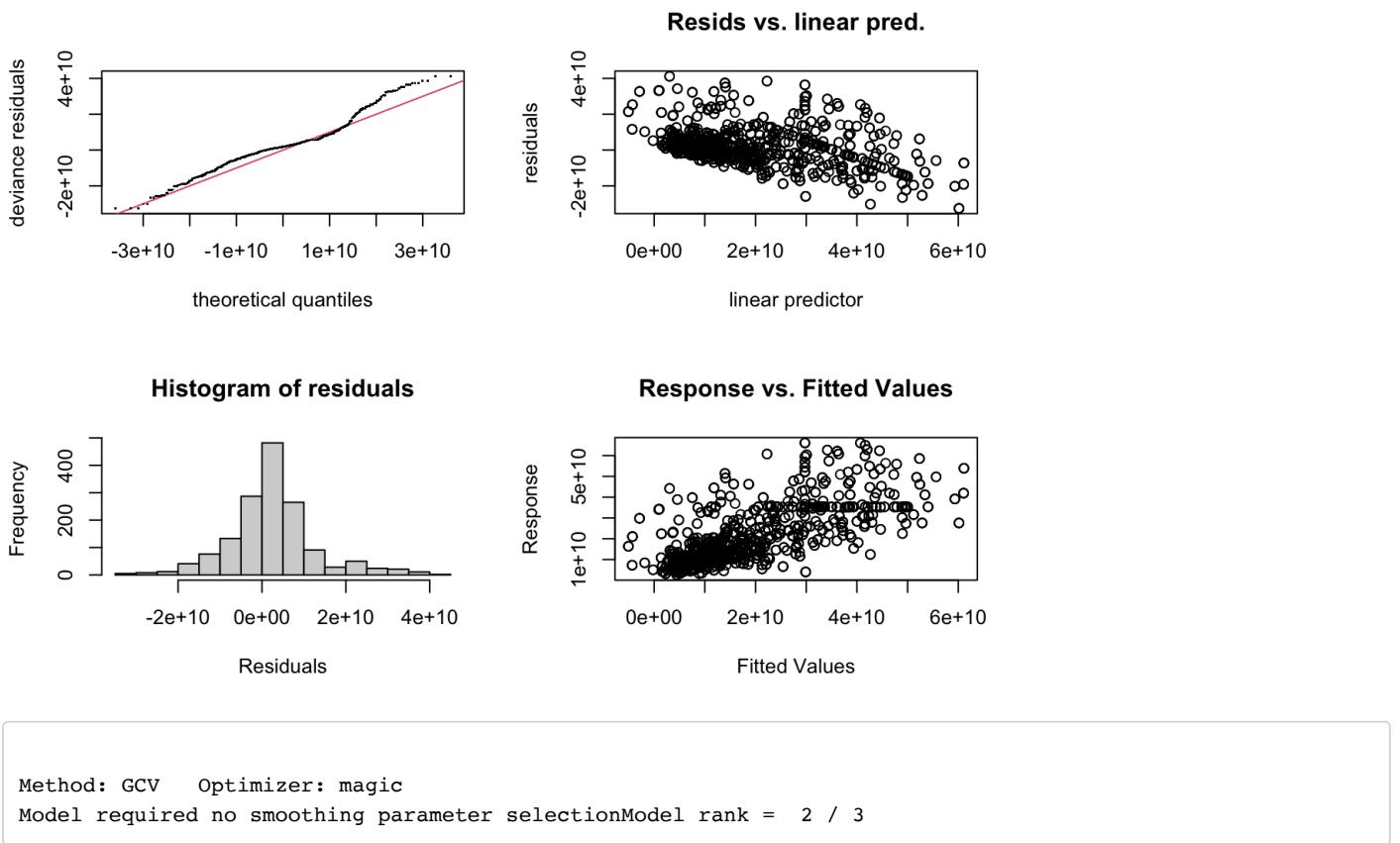
### Total.Equity



### Earnings.Before.Interest.and.Tax



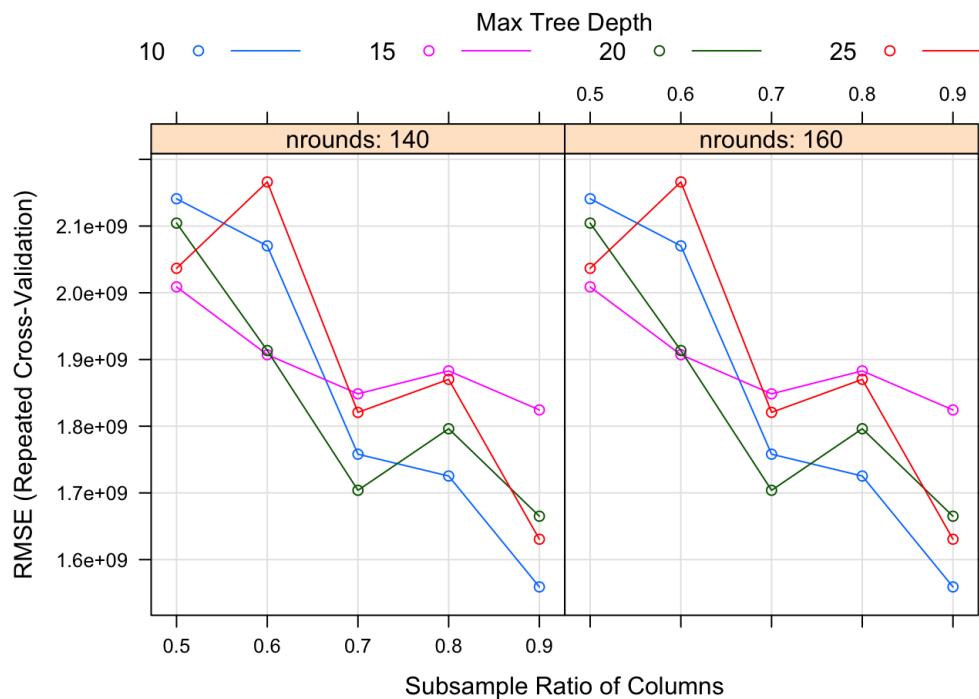
- Diagnosing GAM model issues



## Extreme Gradient Boosting

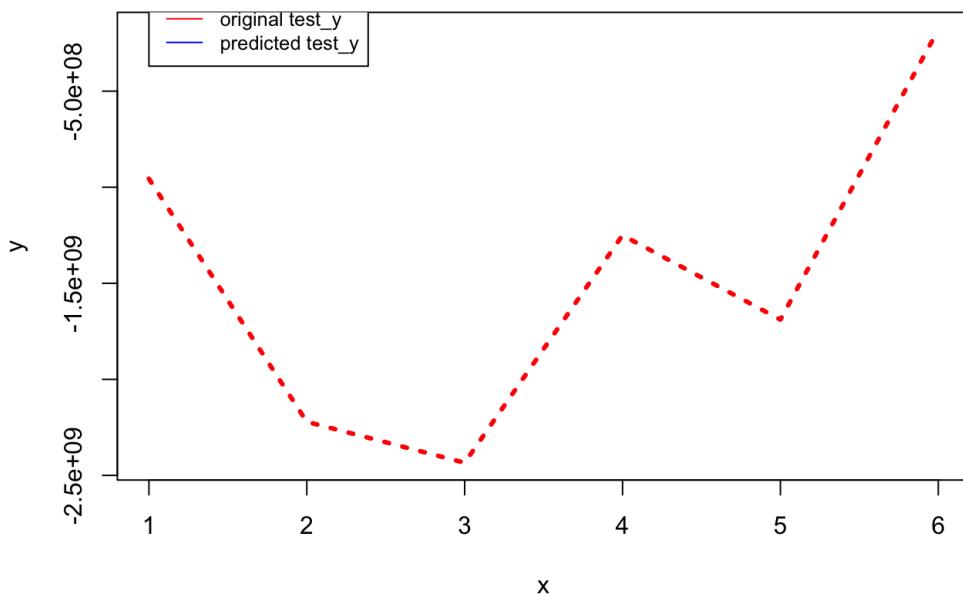
- XGBoost is an implementation of the Gradient Boosted Decision Trees algorithm.
- XGBoost supports various objective functions, including regression, classification and ranking.
- XGBoost gives among the best performances in many machine learning applications. It is optimized gradient-boosting machine learning library. The core algorithm is parallelizable and hence it can use all the processing power of your machine and the machines in your cluster. In R, according to the package documentation, since the package can automatically do parallel computation on a single machine, it could be more than 10 times faster than existing gradient boosting packages.
- XGBoost shines when we have lots of training data where the features are numeric or mixture of numeric and categorical fields. It is also important to note that xgboost is not the best algorithm out there when all the features are categorical or when the number of rows is less than the number of fields (columns).
- The data argument in the xgboost R function is for the input features dataset. It accepts a matrix, dgCMatrix, or local data file. The nrounds argument refers to the max number of iterations (i.e. the number of trees added to the model).
- There are different hyperparameters that we can tune and the parameters are different from baselearner to baselearner. In tree based learners, which are the most common ones in xgboost applications, the following are the most commonly tuned hyperparameters:
  - learning rate: learning rate/eta- governs how quickly the model fits the residual error using additional base learners. If it is a smaller learning rate, it will need more boosting rounds, hence more time, to achieve the same reduction in residual error as one with larger learning rate. Typically, it lies between 0.01 - 0.3
  - The three hyperparameters below are regularization hyperparameters.
  - gamma: min loss reduction to create new tree split. default = 0 means no regularization.
  - lambda: L2 reg on leaf weights. Equivalent to Ridge regression.
  - alpha: L1 reg on leaf weights. Equivalent to Lasso regression.
  - max\_depth: max depth per tree. This controls how deep our tree can grow. The Larger the depth, more complex the model will be and higher chances of overfitting. Larger data sets require deep trees to learn the rules from data. Default = 6.

- subsample: % samples used per tree. This is the fraction of the total training set that can be used in any boosting round. Low value may lead to underfitting issues. A very high value can cause over-fitting problems.
- colsample\_bytree: % features used per tree. This is the fraction of the number of columns that we can use in any boosting round. A smaller value is an additional regularization and a larger value may be cause overfitting issues.
- n\_estimators: number of estimators (base learners). This is the number of boosting rounds.



## Extreme Gradient Boosting Prediction & Accuracy

MSE: 3.852715e+20 MAE: 18121969088 RMSE: 19628334514



3988987596

0.2038301

0.8701342

RMSE: 3988987596 Error: 0.2038301 R2: 0.8701342

## Decision Tree to Trade Bank of America Stock (Real Time Value Analysis)

- Decision trees are one of the more popular machine-learning algorithms for their ability to model noisy data, easily pick up non-linear trends, and capture relationships between your indicators; they also have the benefit of being easy to interpret.
- Decision trees take a top-down, “divide-and-conquer” approach to analyzing data. They look for the indicator, and indicator value, that best splits the data into two distinct groups.
- The algorithm then repeats this process on each subsequent group until it correctly classifies every data point or a stopping criteria is reached.
- Each split, known as a “node”, tries to maximize the purity of the resulting “branches”. The purity is basically the probability that a data point falls in a given class, in our case “up” or “down”, and is measured by the “information gain” of each split.
- In this model, we are going to use real time data of bofa stock from 2000- 2020. For this we will use quantmod package

### Building a Strategy

- Let's see how we can quickly build a strategy using 4 technical indicators to see whether today's price of BoA's stock is going to close up or down. The 4 technical indicators are:
  - RSI -> Calculate a 3-period relative strength index (RSI) off the open price
  - EMA -> Calculate a 5-period exponential moving average (EMA)
  - MACD -> Calculate a MACD with standard parameters
  - SMI -> Stochastic Oscillator with standard parameters
- Then we calculate the variable we are looking to predict and build our data sets.
  - Class -> Calculate the difference between the close price and open price. If PriceChange>0, “UP”, “DOWN”

"BAC" "RSI3" "EMAcross" "MACDsignal" "Stochastic" "Class"

- Data slicing is a step to split data into train and test set. Training data set can be used specifically for our model building. Test dataset should not be mixed up while building model. Even during standardization, we should not standardize our test set. 75 percent contributes to training & 25 percent of data contributes to test.
- Training the Decision Tree classifier with criterion as information gain
- We are setting 3 parameters of trainControl() method. The “method” parameter holds the details about resampling method. We can set “method” to use repeatedcv i.e, repeated cross-validation.
- The “number” parameter holds the number of resampling iterations. The “repeats” parameter contains the complete sets of folds to compute for our repeated cross-validation. We are using setting number =10 and repeats =3. This trainControl() methods returns a list. We are going to pass this on our train() method.
- To select the specific strategy for decision tree, we need to pass a parameter “parms” in our train() method. It should contain a list of parameters for our rpart method. For splitting criterions, we need to add a “split” parameter with values either “information” for information gain & “gini” for gini index. We are using information gain as a criterion.

### Trained Decision Tree classifier results:

- We can check the result of our train() method by a print dtree\_fit variable. It is showing us the accuracy metrics for different values of cp. Here, cp is complexity parameter for our dtree.

CART

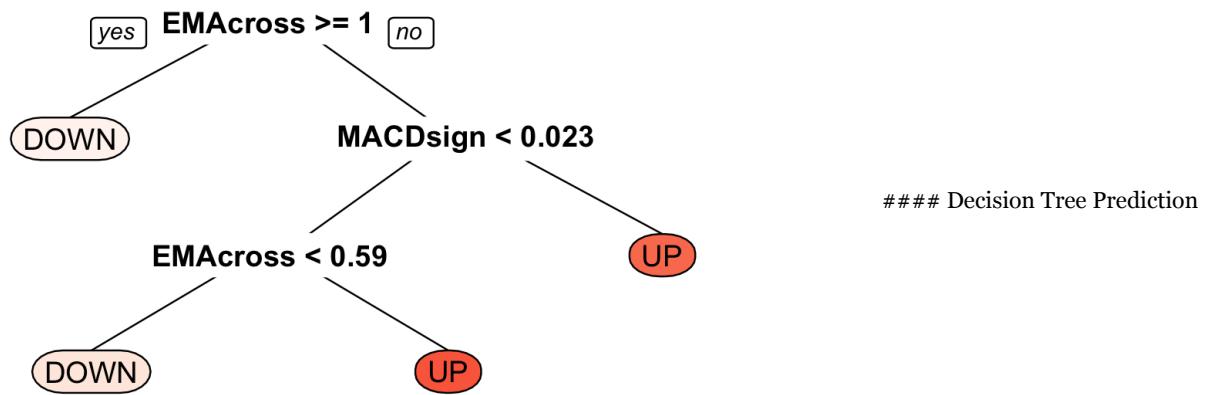
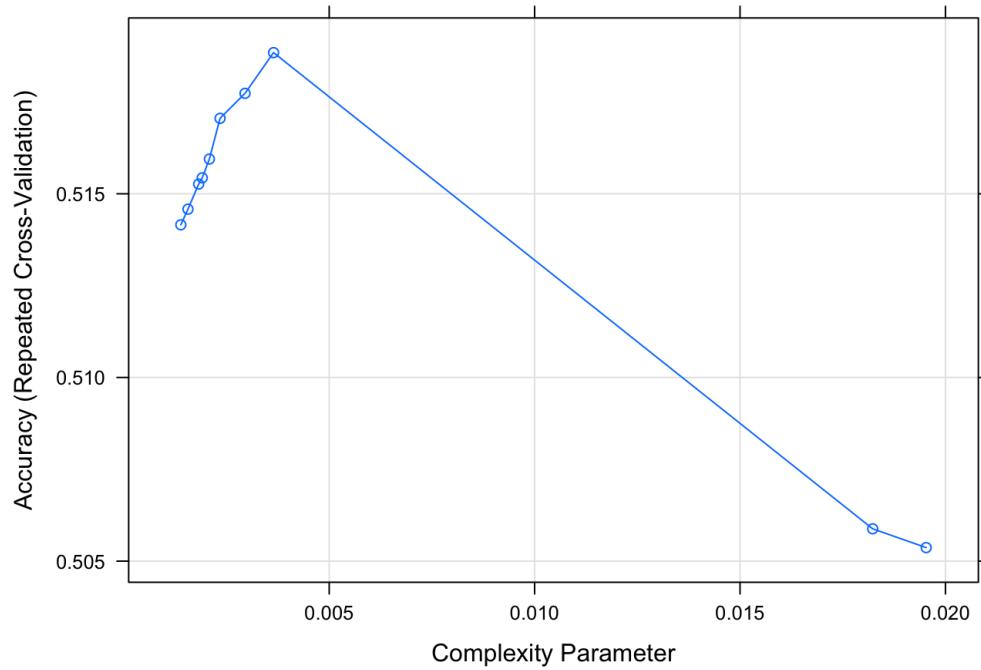
```
3908 samples
  4 predictor
  2 classes: 'DOWN', 'UP'

Pre-processing: centered (4), scaled (4)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 3517, 3518, 3517, 3517, 3517, ...
Resampling results across tuning parameters:
```

cp	Accuracy	Kappa
0.001388889	0.5141566	0.0301167276
0.001562500	0.5145822	0.0308771188
0.001822917	0.5152644	0.0322216484
0.001909722	0.5154349	0.0325509692
0.002083333	0.5159460	0.0335094058
0.002343750	0.5170542	0.0345975203
0.002951389	0.5177349	0.0358283997
0.003645833	0.5188436	0.0377622413
0.018229167	0.5058795	0.0006184276
0.019531250	0.5053680	-0.0004969420

```
Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.003645833.
```

## Plot Decision Tree



Now, our model is trained with  $cp = 0.003645833$ . Accuracy of the model is 0.5141566.

## Confusion Matrix and Statistics

```
Reference
Prediction DOWN UP
    DOWN 317 279
    UP     345 360

Accuracy : 0.5204
95% CI : (0.4928, 0.5478)
No Information Rate : 0.5088
P-Value [Acc > NIR] : 0.210682

Kappa : 0.0422

McNemar's Test P-Value : 0.009266

Sensitivity : 0.4789
Specificity : 0.5634
Pos Pred Value : 0.5319
Neg Pred Value : 0.5106
Prevalence : 0.5088
Detection Rate : 0.2437
Detection Prevalence : 0.4581
Balanced Accuracy : 0.5211

'Positive' Class : DOWN
```

## KNN Classifier to predict if daily stock price will increase

- kNN algorithm is a non-parametric algorithm that can be used for either classification or regression. Non-parametric means that it makes no assumption about the underlying data or its distribution.
- For each data point, the algorithm finds the k closest observations, and then classifies the data point to the majority. Usually, the k closest observations are defined as the ones with the smallest Euclidean distance to the data point under consideration.
- K-Nearest Neighbors computes the likelihood of each share closing price based on other technical indicator.

### Building KNN model

- Data Model which we will use for ML is listed below. Here, the 3 columns represent the closing price, opening price, High price, volume of the stock on the given date.
- The Increase column represents whether the price of stock rose or fell as compared to the previous day.
- We are using caret's KNN to sample the data (for training and testing), preprocessing, evaluating the model etc.,
- **Step 1: Data Splicing**
  - KNN algorithm is applied to the training data set and the results are verified on the test data set.
  - I used 25% to test data and 75% to train the data.
  - After obtaining training and testing data sets, then we will create a separate data frame from testing data set which has values to be compared with actual final values
  - Predictor variables are Open, Increase, Volume, High. Target variable is Close
- **Step 2: Data Pre-Processing With Caret**
  - We need to pre-process our data before we can use it for modeling.
  - The caret package in R provides a number of useful data transforms.
  - Training transforms can be prepared and applied automatically during model evaluation.
  - Transforms applied during training are prepared using the `preProcess()` and passed to the `train()` function via the `preProcess` argument.
  - Combining the scale and center transforms for preprocessing will standardize the data.
  - The scale transform calculates the standard deviation for an attribute and divides each value by that standard deviation.
  - The center transform calculates the mean for an attribute and subtracts it from each value.
  - Attributes after preprocessing will have a mean value of 0 and a standard deviation of 1.

- **Step 3:Model Training and Tuning**

- To control parameters for train, trainControl function is used. “trainControl” allows estimation of parameter coefficients through resampling methods like cross validation, repeatedcv,boosting etc.
- The option “repeatedcv” method which we used in traincontrol method controls the number of repetitions for resampling used in repeated K-fold cross-validation.

```
colnames(bny_raw)
1             Date
2             Open
3      increase
4            Volume
5            Close
6            High
```

- **Step 4:How to choose value for K to improve performance**

- Time to fit a knn model using caret with preprocessed values.
- From the output of the knn model ,maximum RSquared(0.9983847) is achieved by k = 5
- Now we can plot our KNN , and view summary of the model

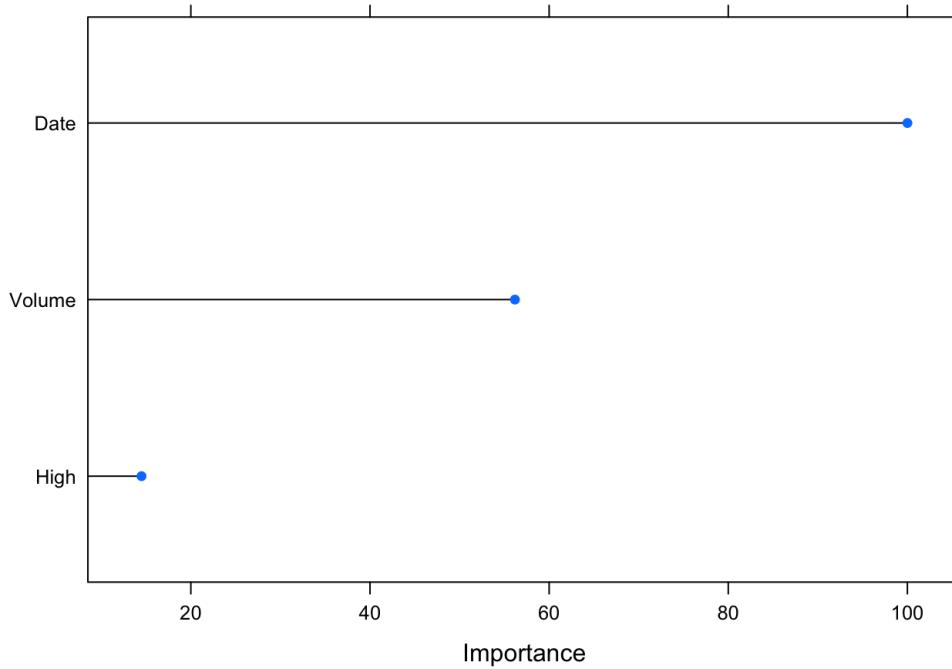
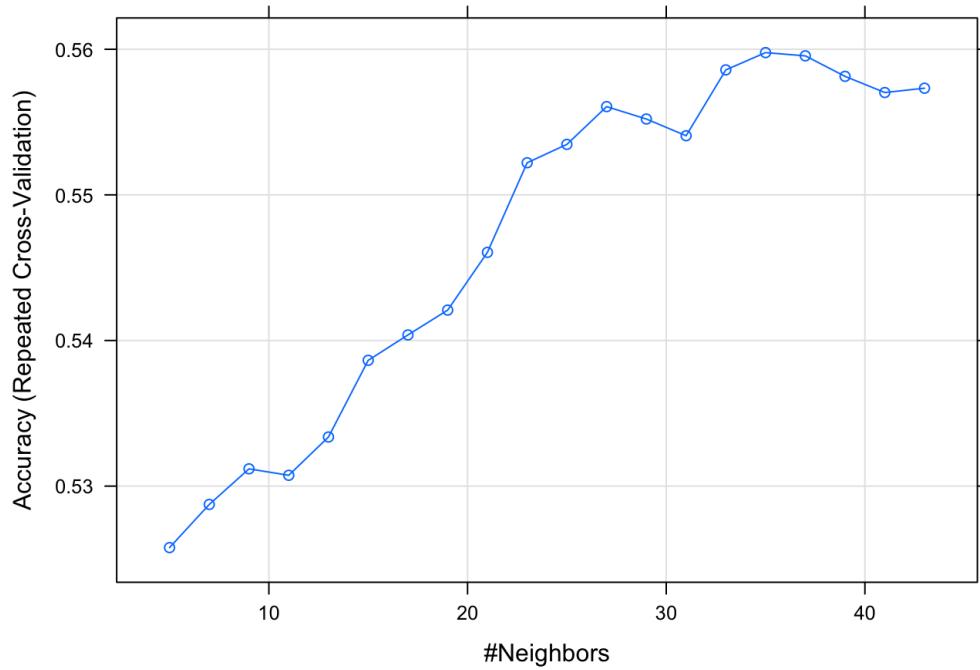
```
k-Nearest Neighbors

8993 samples
 5 predictor
 2 classes: 'FALSE', 'TRUE'

Pre-processing: centered (5), scaled (5)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 8093, 8094, 8093, 8094, 8094, 8095, ...
Resampling results across tuning parameters:
```

k	Accuracy	Kappa
5	0.5257777	0.02578255
7	0.5287424	0.02899955
9	0.5311870	0.03274320
11	0.5307430	0.02950820
13	0.5333738	0.03257598
15	0.5386393	0.04107065
17	0.5403822	0.04288474
19	0.5420874	0.04482209
21	0.5460535	0.05178726
23	0.5522058	0.06325600
25	0.5534678	0.06464222
27	0.5560607	0.06852801
29	0.5552093	0.06542168
31	0.5540608	0.06233631
33	0.5585827	0.07069766
35	0.5597692	0.07214944
37	0.5595463	0.07104593
39	0.5581399	0.06758583
41	0.5570294	0.06480353
43	0.5573246	0.06410129

Accuracy was used to select the optimal model using the largest value.  
The final value used for the model was k = 35.



## KNN Accuracy

- **Step 5: Making predictions**
  - We build knn by using training & test data sets. After building the model, then we can check the accuracy of forecasting using confusion matrix.
- **Step 6: Interpretation of the results and prediction accuracy achieved.**
  - The accuracy of our model on the testing set is 55%.
  - We can visualise the model's performance using a confusion matrix.
  - We can also evaluate the accuracy, precision and recall on the training and validation sets in confusion matrix to evaluate the performance of knn algorithm.

## Confusion Matrix and Statistics

```
Reference
Prediction FALSE TRUE
  FALSE    426   446
  TRUE     895  1229

  Accuracy : 0.5524
  95% CI  : (0.5344, 0.5703)
  No Information Rate : 0.5591
  P-Value [Acc > NIR] : 0.7748

  Kappa : 0.0583

  Mcnemar's Test P-Value : <2e-16

  Sensitivity : 0.3225
  Specificity  : 0.7337
  Pos Pred Value : 0.4885
  Neg Pred Value : 0.5786
  Prevalence   : 0.4409
  Detection Rate : 0.1422
  Detection Prevalence : 0.2911
  Balanced Accuracy : 0.5281

  'Positive' Class : FALSE
```

Accuracy: 0.5524032

## Overall insights obtained from the implemented project

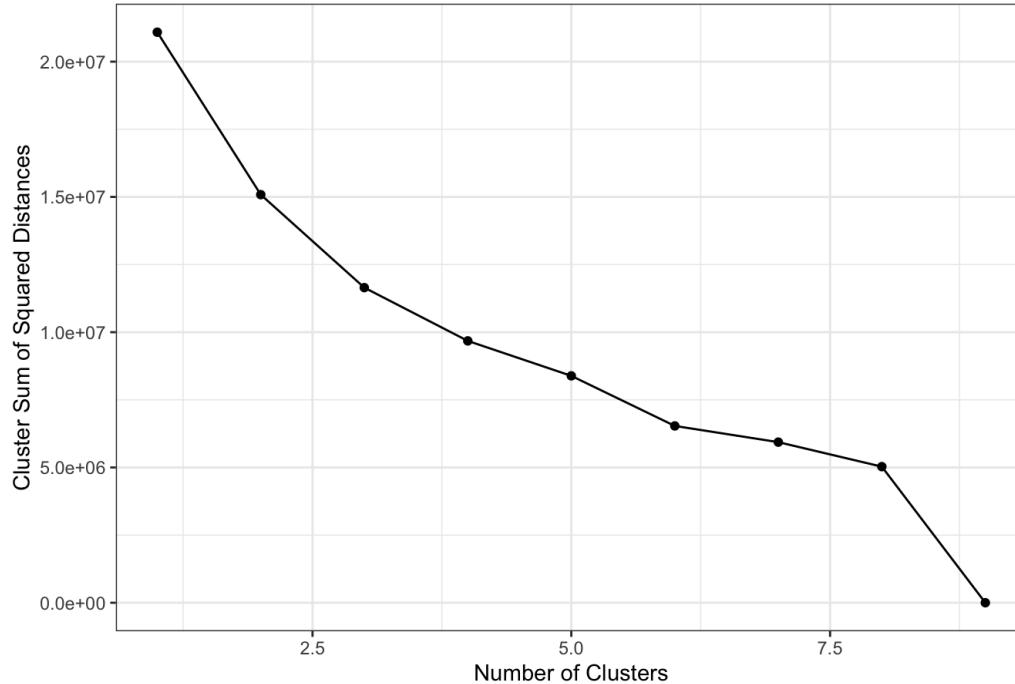
- As we can see, the model has the highest accuracy of 55 when k = 5. While this may not seem any good, it is often extremely hard to predict the price of stocks. Even the 2.5% improvement over random guessing can make a difference given the amount of money at stake. After all, if it was that easy to predict the prices, wouldn't we all be trading in stocks for the easy money instead of learning these algorithms?
- Sensitivity for price increase is 0.3225 .
- Specificity for price increase is 0.7337.

## K-means Clustering NYSE trades.

- K-means clustering is an unsupervised learning method used to uncover non-random structures in your data.
- It does so by creating labels for your training data, where each label is a cluster.
- An ideal cluster analysis groups points together on a scatterplot, with maximized space (separation) between groups, and minimized space (cohesion) between points in the same group.
- Groups can be centered around a non-real point (k-means).
- **Challenge: Group stock based on performances**
- Perform k-means clustering of the New York Stock Exchange (NYSE) dataset that has more than 9,211,031 NYSE trade data. Columns from 1 to 7 are:
  - ID INTEGER record ID
  - OPEN\_P DOUBLE open price
  - HIGH\_P DOUBLE highest price
  - LOW\_P DOUBLE lowest price
  - CLOSE\_P DOUBLE close price
  - VOLUME DOUBLE volume
  - CLOSE\_ADJ\_P DOUBLE close adjusted price
- **Choose K Value**

- Use columns 2 from the input data and perform the k-means clustering with k = 0 to 15. Set the maximum number of iterations to 10,000.
- Use Within Cluster Sum of Squares as a measure to choose best K , which measures the squared average distance of all the points within a cluster to the cluster centroid. To calculate WCSS, you first find the Euclidean distance (see figure below) between a given point and the centroid to which it is assigned. You then iterate this process for all points in the cluster, and then sum the values for the cluster and divide by the number of points. Finally, you calculate the average across all clusters. This will give you the average WCSS.
- From the below plot we can see K=7 will not overfit/underfit the model.

Trading Training Data Scree Plot



### Modelling

- We'll now use the k-means algorithm to fit a model.
- It will divide all datapoints into 7 clusters, and allocate each point to a cluster group using a distance (euclidean) measurement.
- We'll set it to iteratively determine which point belongs to each cluster 10,000 times, allowing for each iteration to recalculate where the center of the groups is and which points are closest to those centers.
- We can visualize the coefficients with the following plot.

```

      Length Class  Mode
cluster      33333 -none- numeric
centers        14 -none- numeric
totss          1 -none- numeric
withinss       7 -none- numeric
tot.withinss   1 -none- numeric
betweenss      1 -none- numeric
size           7 -none- numeric
iter           1 -none- numeric
ifault         1 -none- numeric

```

```

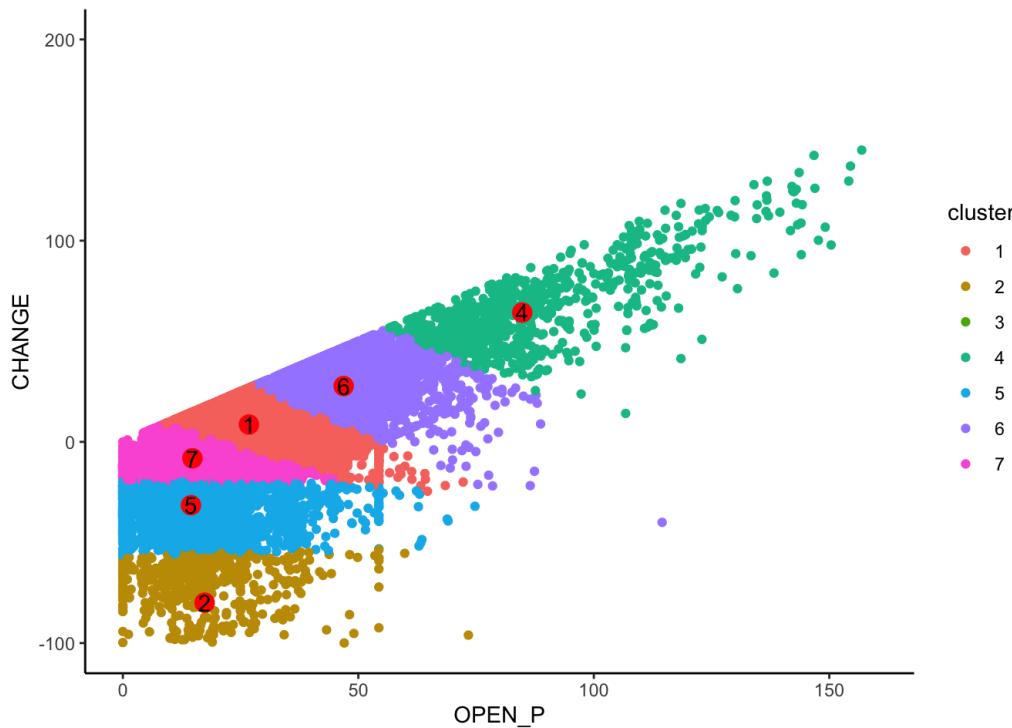
Centers:  26.77171 17.36502 298.0009 84.80548 14.4589 46.8826 14.77272 8.621914 -79.9289 275.3313 64.2736
4 -31.47357 27.74873 -8.16676

```

```
No Of Records in Cluster: 10464 681 23 1032 5088 5087 10958
```

### KNN Cluster Plotting

- Plot to see how opening prices have been distributed in clusters.



## Conclusion

- KNN is a lazy learner where generalization of the training data is delayed until a query is made to the system. Which means knn starts working only when you trigger it to, thus lazy learning methods can construct a different approximation or result to the target function for each encountered query.
- It is a good method for online learning but it requires a possibly large amount of memory to store the data, and each request involves starting the identification of a local model from scratch.

## ARIMA Forecasting Expedia Stock Price incorporating COVID-19 (Real Time)

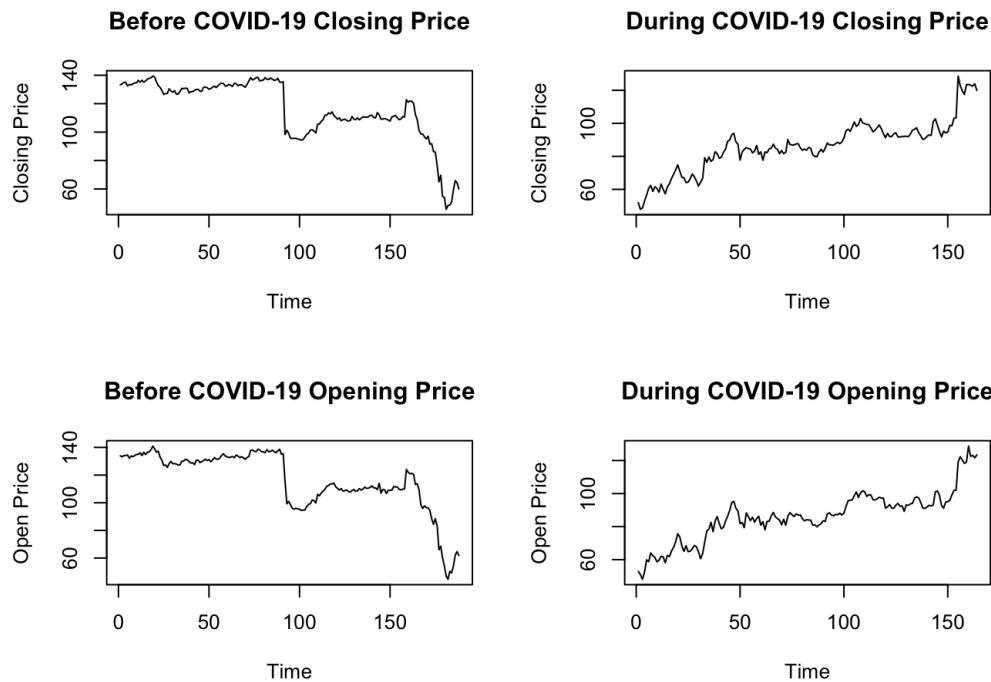
- The goal of this project is to predict the future stock price of Expedia using various predictive forecasting models and then analysing the models using ARIMA.
- The dataset for Expedia stocks is obtained from Yahoo Finance using Quantmod package in R.
- The timeline of the data is from 2019 till present day(11/26/2020).
- We shall also try and understand the impact of COVID-19 disaster on the stock prices of Expedia.
- ***Forecasting***
  - A forecasting algorithm is a process that seeks to predict future values based on the past and present data.
  - This historical data points are extracted and prepared trying to predict future values for a selected variable of the dataset.
- ***Data Preparation***
  - Importing the data : We obtain the data of Expedia from “2019-07-01” to “2020-11-26” of Expedia Stock price for our analysis using the quantmod package. To analyse the impact of COVID-19 on the Expedia Stock price, we take two sets of data from the quantmod package.
  - Data from “2019-07-01” - “2020-03-28” is data before covid.
  - Data from “2020-04-01” - “till date” is data before covid.
  - All the analysis and the models will be made on both the datasets to analyse the impact of COVID-19, if any.

```

colnames(google_data_after_covid)
1 EXPE.Open
2 EXPE.High
3 EXPE.Low
4 EXPE.Close
5 EXPE.Volume
6 EXPE.Adjusted

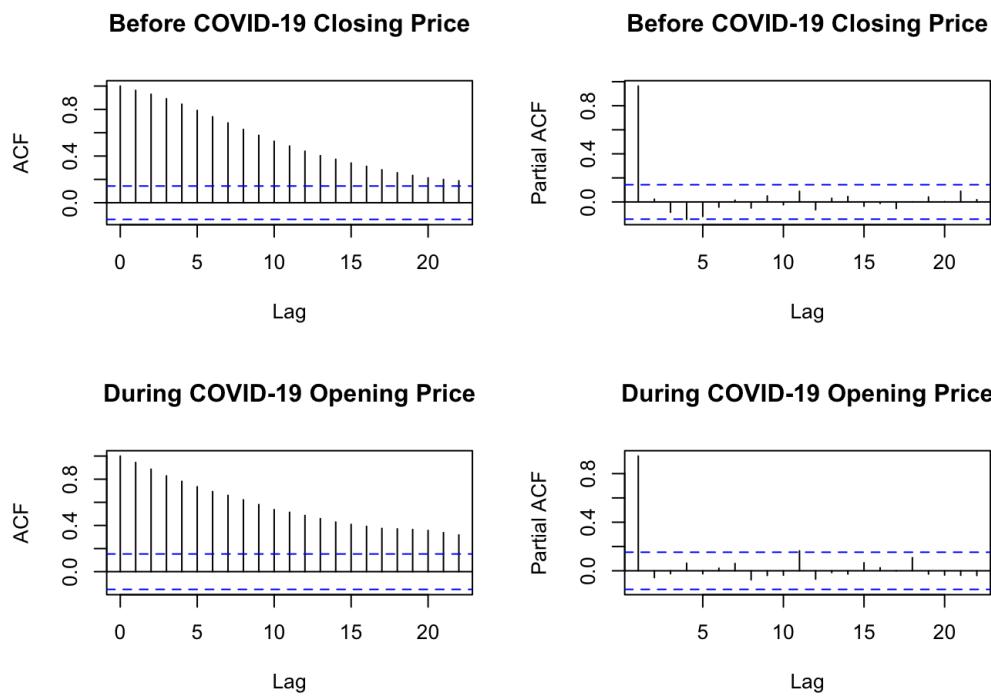
```

## Graphical Representation of Data



## ARIMA Model

- Let us first analyse the ACF and PACF Graph of each of the two datasets.



- We then use the auto.arima function to determine the time series model for each of the datasets.

```

Series: tsData_before_covid_close
ARIMA(3,1,2)
Box Cox transformation: lambda= -0.5522943

Coefficients:
      ar1      ar2      ar3      ma1      ma2
    0.9436   -0.2552   -0.2793  -0.9659   0.7154
s.e.  0.1333   0.1520   0.0991   0.1174   0.1094

sigma^2 estimated as 1.264e-05:  log likelihood=791.34
AIC=-1570.69  AICc=-1570.22  BIC=-1551.3

Training set error measures:
      ME      RMSE      MAE      MPE      MAPE      MASE
Training set -0.3468986 3.779995 2.064975 -0.4697368 2.119611 1.117624
      ACF1
Training set -0.01611792

```

```

Series: tsData_after_covid_close
ARIMA(1,1,0) with drift
Box Cox transformation: lambda= 1.049021

Coefficients:
      ar1      drift
    -0.0993   0.5238
s.e.  0.0785   0.3099

sigma^2 estimated as 19.12:  log likelihood=-470.79
AIC=947.57  AICc=947.73  BIC=956.86

Training set error measures:
      ME      RMSE      MAE      MPE      MAPE      MASE
Training set -0.002349432 3.471361 2.361182 -0.07665327 2.827689 0.9868023
      ACF1
Training set 0.002088858

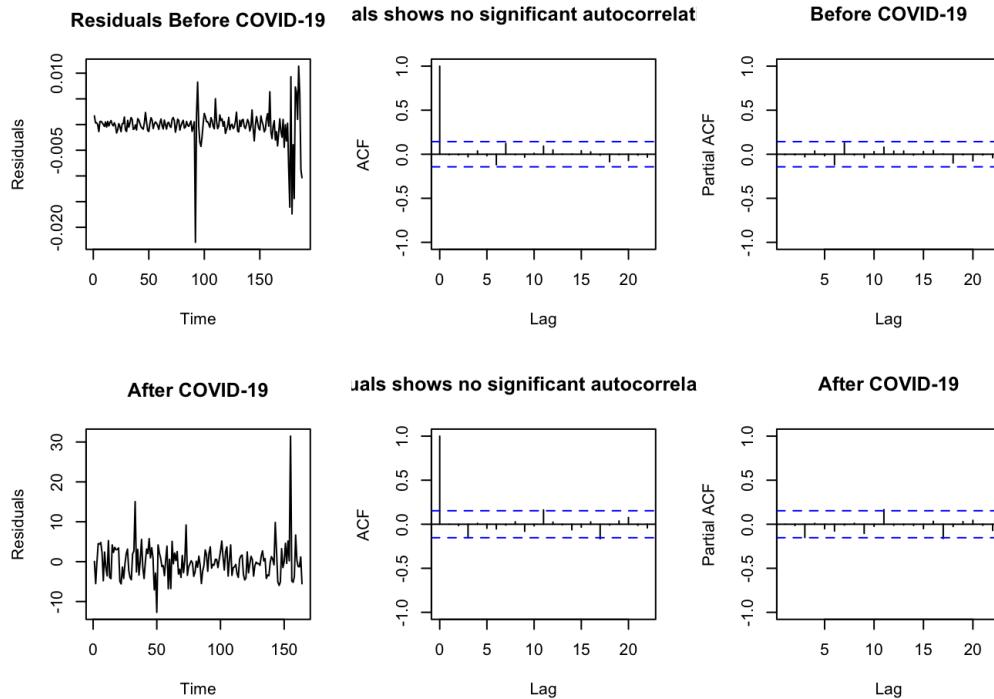
```

- From the auto.arima function, we conclude the following models for the two datasets:
  - Before COVID-19: ARIMA(3,1,2)
  - After COVID-19: ARIMA(1,1,0)
- After obtaining the model, we then perform residual diagnostics for each of the fitted models.
- From the residual plot , we can confirm that the residual has a mean of 0 and the variance is constant as well . The ACF is 0 for lag>0 , and the PACF is 0 as well.
- So, we can say that the residual behaves like white noise and conclude that the models ARIMA(3,1,2) and ARIMA(1,1,0) fits the data well. Alternatively, we can also test at a significance level using the Box-Ljung Test.

## Diagnostic measures

- Try to find out the pattern in the residuals of the chosen model by plotting the ACF of the residuals, and doing a portmanteau test. We need to try modified models if the plot doesn't look like white noise.
- Once the residuals look like white noise, calculate forecasts.
- Box-Ljung test is a test of independence at all lags up to the one specified. Instead of testing randomness at each distinct lag, it tests the “overall” randomness based on a number of lags, and is therefore a portmanteau test. It is applied to the residuals of a fitted ARIMA model, not the original series, and in such applications the hypothesis actually being tested is that the residuals from the ARIMA model have no autocorrelation.
- The ACF of the residuals shows no significant autocorrelations.

- The p-values for the Ljung-Box Q test all are well above 0.05, indicating “non-significance.” The values are normal as they rest on a line and aren’t all over the place.



## Augmented Dickey-Fuller & Kwiatkowski-Phillips-Schmidt-Shin

- We then conduct an ADF (Augmented Dickey-Fuller) test and KPSS (Kwiatkowski-Phillips-Schmidt-Shin) test to check for the stationarity of the time series data for both the datasets closing price.

### Augmented Dickey-Fuller Test

```
data: tsData_before_covid_close
Dickey-Fuller = -2.1153, Lag order = 5, p-value = 0.5279
alternative hypothesis: stationary
```

### Augmented Dickey-Fuller Test

```
data: tsData_after_covid_close
Dickey-Fuller = -2.4665, Lag order = 5, p-value = 0.3817
alternative hypothesis: stationary
```

- From the above ADF tests, we can conclude the following:
  - For the dataset before COVID-19, the ADF tests gives a p-value of 0.5279 which is greater than 0.05, thus implying that the time series data is not stationary.
  - For the dataset after COVID-19, the ADF tests gives a p-value of 0.3817 which is lesser than 0.05, thus implying that the time series data is not stationary.

### KPSS Test for Level Stationarity

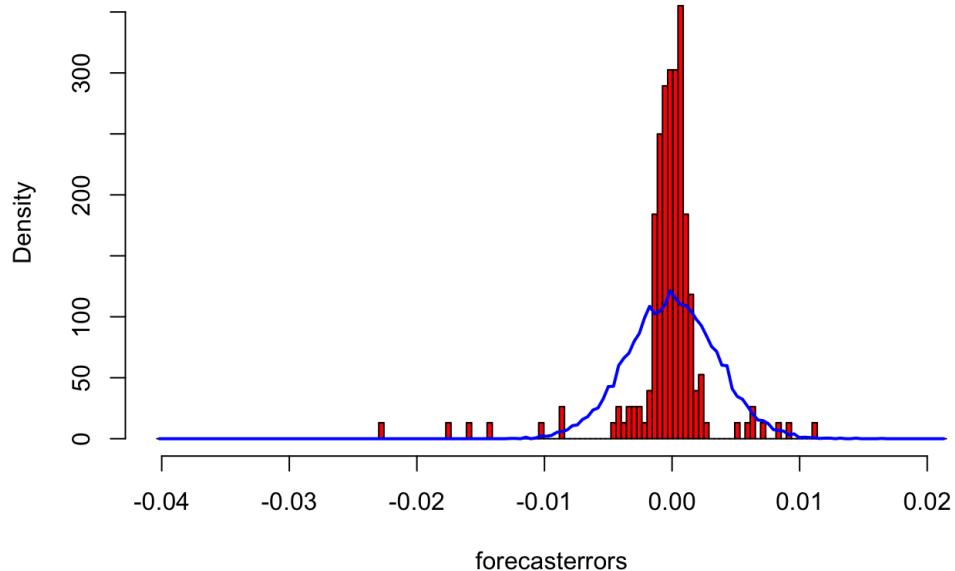
```
data: tsData_before_covid_close
KPSS Level = 2.3907, Truncation lag parameter = 4, p-value = 0.01
```

```
KPSS Test for Level Stationarity
```

```
data: tsData_after_covid_close
KPSS Level = 2.6532, Truncation lag parameter = 4, p-value = 0.01
```

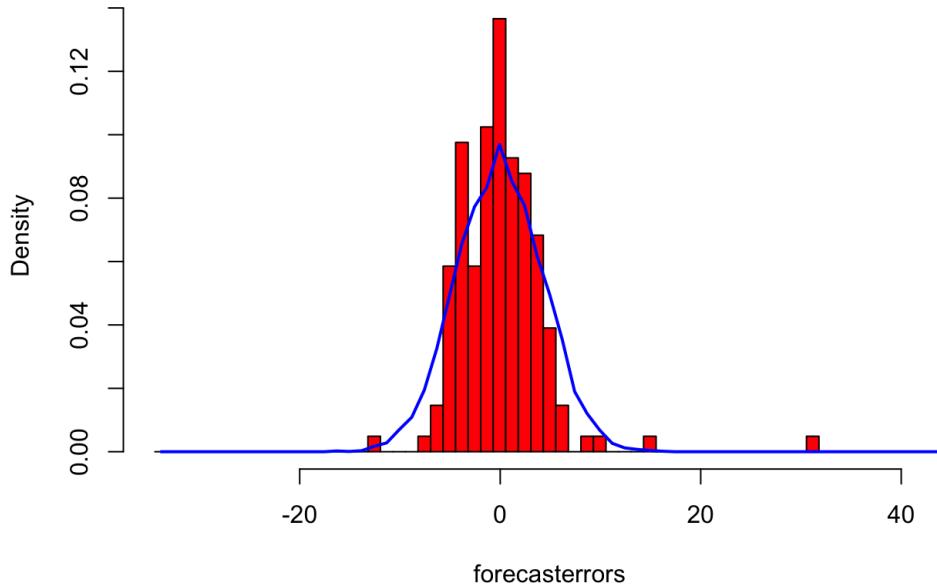
- From the above KPSS tests, we can conclude the following:
  - For the dataset before COVID-19, the KPSS tests gives a p-value of 0.01 which is less than 0.05, thus implying that the time series data is not stationary.
  - For the dataset after COVID-19, the KPSS tests gives a p-value of 0.01 which is less than 0.05, thus implying that the time series data is not stationary.
  - Thus, we can conclude from the above tests that the time series data is not stationary.
- ***Forecasting with ARIMA Models***
- Forecast errors for before covid dataset

**Histogram of forecasterrors**



- Forecast errors for after covid dataset

## Histogram of forecasterrors



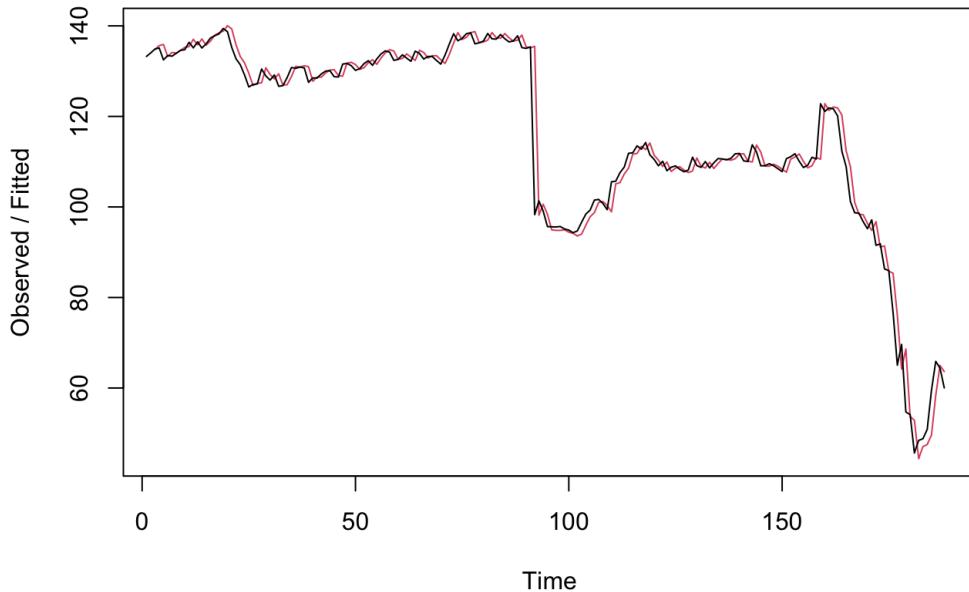
- **Holt Winters**

- To make forecasts using simple exponential smoothing in R, we can fit a simple exponential smoothing predictive model using the “HoltWinters()” function in R. To use HoltWinters() for simple exponential smoothing, we need to set the parameters beta=FALSE and gamma=FALSE in the HoltWinters() function (the beta and gamma parameters are used for Holt’s exponential smoothing, or Holt-Winters exponential smoothing, as described below).
- HoltWinters() function returns a list variable, that contains several named elements. The output of HoltWinters() tells us that the estimated value of the alpha parameter is about 0.982813.
- We can plot the original time series.

```
Holt-Winters exponential smoothing with trend and without seasonal component.
```

```
Call:  
HoltWinters(x = tsData_before_covid_close, gamma = FALSE)  
  
Smoothing parameters:  
alpha: 0.982813  
beta : 0.02381423  
gamma: FALSE  
  
Coefficients:  
[,1]  
a 60.091954  
b -0.869578
```

## Holt-Winters filtering



- As a measure of the accuracy of the forecasts, we can calculate the sum of squared errors for the in-sample forecast errors, that is, the forecast errors for the time period covered by our original time series. The sum-of-squared-errors is stored in a named element of the list variable “rainseriesforecasts” called “SSE”, so we can get its value by typing:

```
"skirtsseriesforecasts$SSE"
```

```
2798.291
```

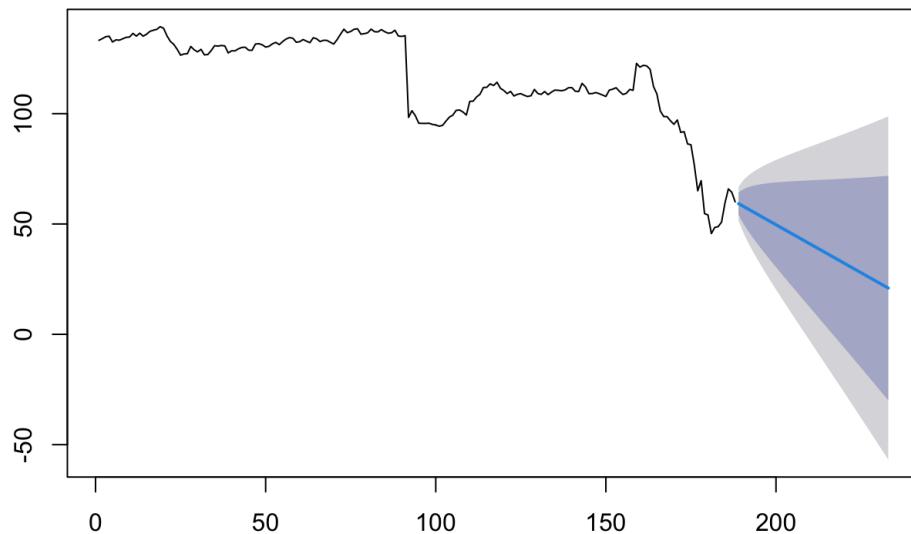
## Forecasting with HoltWinters

- We can make forecasts for further time points by using the “forecast.HoltWinters()” function in the R “forecast” package.
- When using the forecast.HoltWinters() function, as its first argument (input), you pass it the predictive model that you have already fitted using the HoltWinters() function.
- You specify how many further time points you want to make forecasts for by using the “h” parameter in forecast. Here we are going to predict price for 45 days.
- forecast.HoltWinters() function gives you the forecast for a year, a 80% prediction interval for the forecast, and a 95% prediction interval for the forecast. For example, the forecasted stock price for 233 day is about 20.96095, with a 95% prediction interval of (-56.8291332 98.75102).

Point	Forecast	Lo 80	Hi 80	Lo 95	Hi 95
189	59.22238	54.261714	64.18304	51.6356981	66.80905
190	58.35280	51.315518	65.39008	47.5902068	69.11539
191	57.48322	48.787761	66.17868	44.1846635	70.78178
192	56.61364	46.469767	66.75752	41.0999238	72.12736
193	55.74406	44.280562	67.20757	38.2121512	73.27598
194	54.87449	42.178858	67.57011	35.4581988	74.29077
195	54.00491	40.140333	67.86948	32.8008698	75.20895
196	53.13533	38.149294	68.12137	30.2161653	76.05449
197	52.26575	36.194958	68.33655	27.6875932	76.84391
198	51.39617	34.269562	68.52279	25.2032811	77.58907
199	50.52660	32.367315	68.68588	22.7543724	78.29882
200	49.65702	30.483773	68.83026	20.3340704	78.97997
201	48.78744	28.615447	68.95943	17.9370389	79.63784
202	47.91786	26.759545	69.07618	15.5590078	80.27672
203	47.04828	24.913796	69.18277	13.1965057	80.90006
204	46.17871	23.076330	69.28108	10.8466714	81.51074
205	45.30913	21.245588	69.37267	8.5071196	82.11114
206	44.43955	19.420256	69.45884	6.1758412	82.70326
207	43.56997	17.599217	69.54073	3.8511288	83.28882
208	42.70039	15.781515	69.61927	1.5315205	83.86927
209	41.83082	13.966326	69.69531	-0.7842448	84.44588
210	40.96124	12.152934	69.76954	-3.0972610	85.01974
211	40.09166	10.340716	69.84261	-5.4084829	85.59180
212	39.22208	8.529124	69.91504	-7.7187474	86.16291
213	38.35250	6.717675	69.98733	-10.0287917	86.73380
214	37.48293	4.905945	70.05991	-12.3392679	87.30512
215	36.61335	3.093553	70.13314	-14.6507544	87.87745
216	35.74377	1.280164	70.20738	-16.9637667	88.45131
217	34.87419	-0.534523	70.28291	-19.2787648	89.02715
218	34.00461	-2.350779	70.36001	-21.5961611	89.60539
219	33.13504	-4.168844	70.43892	-23.9163256	90.18640
220	32.26546	-5.988937	70.51985	-26.2395913	90.77051
221	31.39588	-7.811255	70.60302	-28.5662585	91.35802
222	30.52630	-9.635974	70.68858	-30.8965985	91.94920
223	29.65672	-11.463254	70.77670	-33.2308566	92.54431
224	28.78715	-13.293243	70.86754	-35.5692552	93.14355
225	27.91757	-15.126070	70.96121	-37.9119963	93.74713
226	27.04799	-16.961857	71.05784	-40.2592630	94.35524
227	26.17841	-18.800712	71.15754	-42.6112221	94.96805
228	25.30884	-20.642734	71.26040	-44.9680254	95.58570
229	24.43926	-22.488014	71.36653	-47.3298109	96.20832
230	23.56968	-24.336634	71.47599	-49.6967045	96.83606
231	22.70010	-26.188669	71.58887	-52.0688212	97.46902
232	21.83052	-28.044188	71.70523	-54.4462656	98.10731
233	20.96095	-29.903253	71.82514	-56.8291332	98.75102

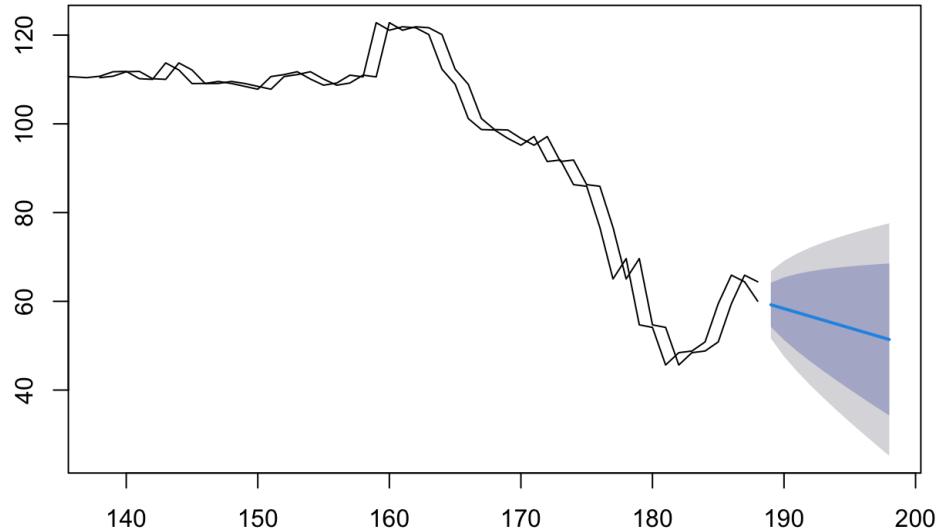
- To plot the predictions made by `forecast.HoltWinters()`, we can use the “`plot.forecast()`” function:

**Forecasts from HoltWinters**



- Prediction for next ten days.

**Forecasts from HoltWinters**



- Ljung-Box Tests**

- Box Test for `modelfit_after_covid$residuals`.
- To check for correlations between successive forecast errors, we can make a correlogram and use the Ljung-Box test.

```
Box-Ljung test
```

```
data: modelfit_after_covid$residuals
X-squared = 1.6058e-05, df = 1, p-value = 0.9968
```

- Box Test for `modelfit_before_covid$residuals`

```

Box-Ljung test

data: modelfit_before_covid$residuals
X-squared = 0.0047812, df = 1, p-value = 0.9449

```

## Conclusion

Here, the p value for both the models is greater than 0.05 . Hence, at a significance level of 0.05 we fail to reject the null hypothesis and conclude that the residual follows white noise. This means that the model fits the data well.

# Using a Neural Network to Model the Amazon Stock Price Change (Real Time Analysis)

- Artificial neural networks are very powerful and popular machine-learning algorithms that mimic how a brain works in order find patterns in your data.
  - In this project, we will build a basic neural network to model the price change of Amazon stocks in real Time.
  - ANNs make predictions by sending the inputs (in our case, the indicators) through the network of neurons, with the neurons firing off depending on the weights of the incoming signals. The final output is determined by the strength of the signals coming from the previous layer of neurons.
- **Data Import**
  - Use “quantmod” package to download information for Amazon stocks.
  - Let’s see how we can quickly build a strategy using 4 technical indicators to see whether today’s price of Amazon’s stock .The 4 technical indicators are:
    - RSI -> Calculate a 3-period relative strength index (RSI) off the open price
    - EMA -> Calculate a 5-period exponential moving average (EMA)
    - MACD -> Calculate a MACD with standard parameters
    - SMI -> Stochastic Oscillator with standard parameters

```
"AMZN"
```

## Normalize Data

- One of the most important procedures when forming a neural network is data normalization. This involves adjusting the data to a common scale so as to accurately compare predicted and actual values. Failure to normalize the data will typically result in the prediction value remaining the same across all observations, regardless of the input values.
- Max-Min Normalization: For this method, we invoke the following function to normalize our data:
- Data slicing:
  - Data slicing is a step to split data into train and test set. Training data set can be used specifically for our model building. Test dataset should not be mixed up while building model. Even during standardization, we should not standardize our test set.75 percent contributes to training & 25 percent of data contributes to test.

```
'data.frame': 5209 obs. of 5 variables:
 $ RSI3      : num  0.0853 0.0409 0.0402 0.6396 0.4188 ...
 $ EMAcross   : num  0.538 0.534 0.539 0.553 0.545 ...
 $ MACDsignal: num  0.727 0.705 0.681 0.661 0.643 ...
 $ BollingerB: num  0.423 0.353 0.362 0.477 0.408 ...
 $ Price      : num  0.417 0.424 0.444 0.422 0.433 ...
```

- TrainingParameters :
  - train() method is passed with repeated cross-validation resampling method for 10 number of resampling iterations repeated for 3 times.

## Machine Learning: Classification using Neural Networks

- Model Training

- We can use neuralnet() to train a NN model. Also, the train() function from caret can help us tune parameters. We can plot the result to see which set of parameters is fit our data the best.
- nnnet package by default uses the Logistic Activation function.
- Data Pre-Processing With Caret: The scale transform calculates the standard deviation for an attribute and divides each value by that standard deviation.
- The center transform calculates the mean for an attribute and subtracts it from each value.
- Combining the scale and center transforms will standardize your data.
- Attributes will have a mean value of 0 and a standard deviation of 1.
- Training transforms can be prepared and applied automatically during model evaluation.
- Transforms applied during training are prepared using the preProcess() and passed to the train() function via the preProcess argument.
- Backpropagation algorithm is a supervised learning method for multilayer feed-forward networks from the field of Artificial Neural Networks.
- The principle of the backpropagation approach is to model a given function by modifying internal weightings of input signals to produce an expected output signal. The system is trained using a supervised learning method, where the error between the system's output and a known expected output is presented to the system and used to modify its internal state.
- We use Backpropagation as algorithm in neural network package.

```
nnetGrid <- expand.grid(size = seq(from = 1, to = 5, by = 1)
                         ,decay = seq(from = 0.1, to = 0.2, by = 0.1))
str(subTrain)
```

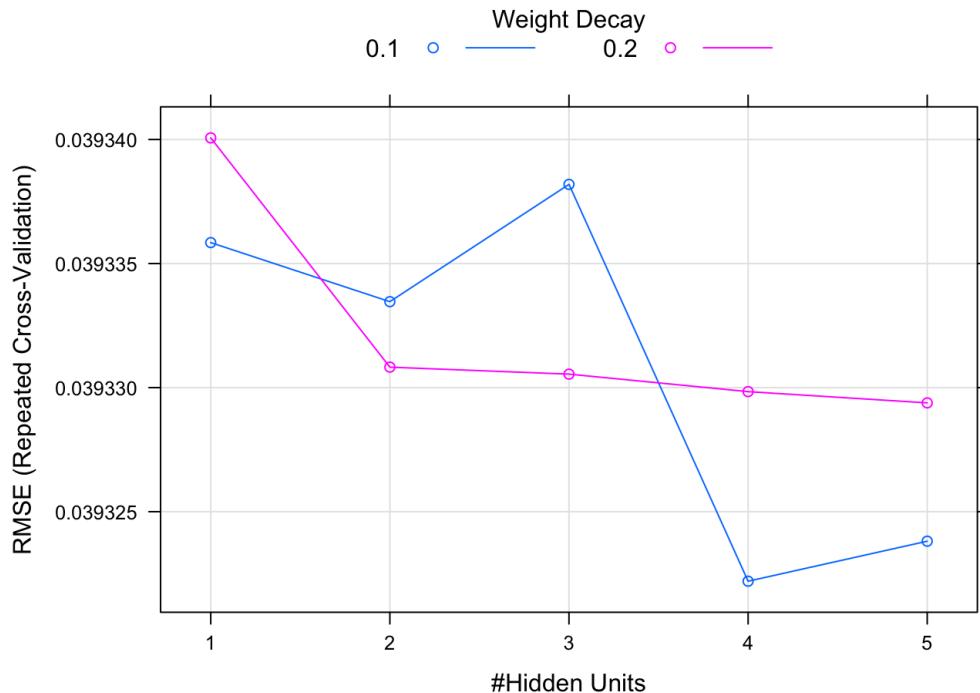
```
'data.frame': 12507 obs. of 5 variables:
 $ RSI3      : num  0.0853 0.0409 0.0402 0.4188 0.3651 ...
 $ EMAcross   : num  0.538 0.534 0.539 0.545 0.544 ...
 $ MACDsignal: num  0.727 0.705 0.681 0.643 0.625 ...
 $ BollingerB: num  0.423 0.353 0.362 0.408 0.386 ...
 $ Price      : num  0.417 0.424 0.444 0.433 0.426 ...
```

```
nn_model <- train(Price~RSI3+EMAcross+MACDsignal+BollingerB, subTrain,
                  method = "nnet", algorithm = 'backprop',
                  trControl= TrainingParameters,
                  preProcess=c("scale","center"),
                  na.action = na.omit,
                  tuneGrid = nnetGrid,
                  trace=FALSE,
                  verbose=FALSE)
```

- Based on the caret neural network model, train sets hidden layer.caret neural network picks the best neural network based on size, decay.

- We can visualize accuracy for different hidden layers below:

	size	decay	RMSE	Rsquared	MAE	RMSESD	RsquaredSD
1	1	0.1	0.03933584	0.01238798	0.01586244	0.002504076	0.01443134
2	1	0.2	0.03934006	0.01239107	0.01583777	0.002507265	0.01451383
3	2	0.1	0.03933346	0.01268185	0.01586449	0.002503906	0.01468389
4	2	0.2	0.03933083	0.01281302	0.01584319	0.002506894	0.01488929
5	3	0.1	0.03933819	0.01287279	0.01586899	0.002504096	0.01467428
6	3	0.2	0.03933054	0.01293707	0.01584766	0.002504418	0.01501688
7	4	0.1	0.03932220	0.01314280	0.01584623	0.002496180	0.01459575
8	4	0.2	0.03932984	0.01300564	0.01584778	0.002504590	0.01508909
9	5	0.1	0.03932381	0.01361986	0.01584481	0.002522508	0.01473453
10	5	0.2	0.03932938	0.01304931	0.01584684	0.002504771	0.01512491
		MAESD					
1		0.0007305646					
2		0.0007313067					
3		0.0007312625					
4		0.0007335536					
5		0.0007326783					
6		0.0007321606					
7		0.0007377006					
8		0.0007321347					
9		0.0007322239					
10		0.0007327730					



- Prediction
  - Now, our model is trained with accuracy = 0.8889 We are ready to predict classes for our test set.

## Accuracy of Neural Network model:

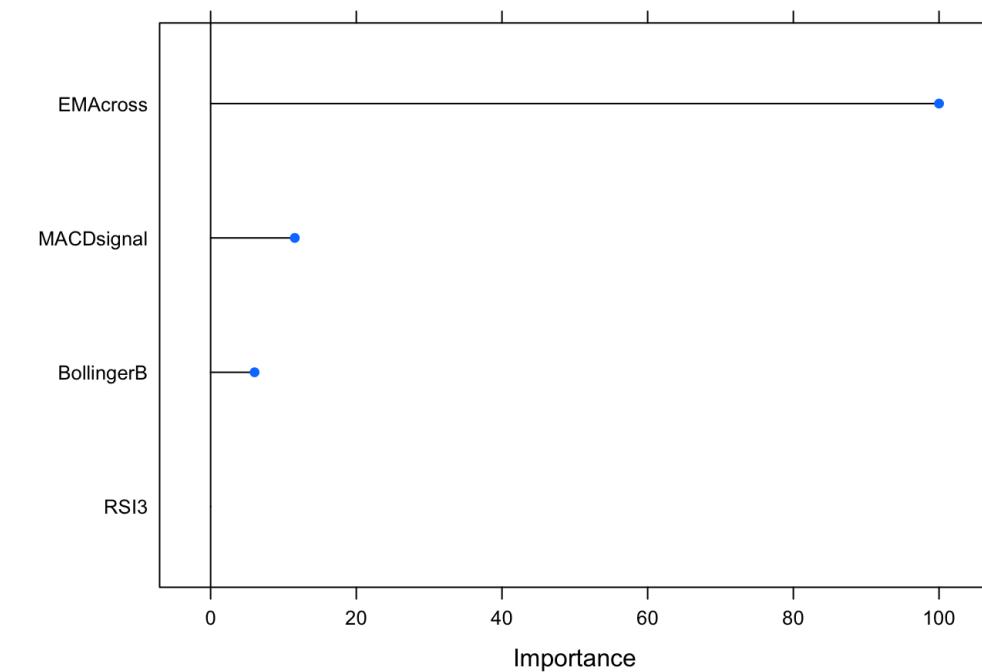
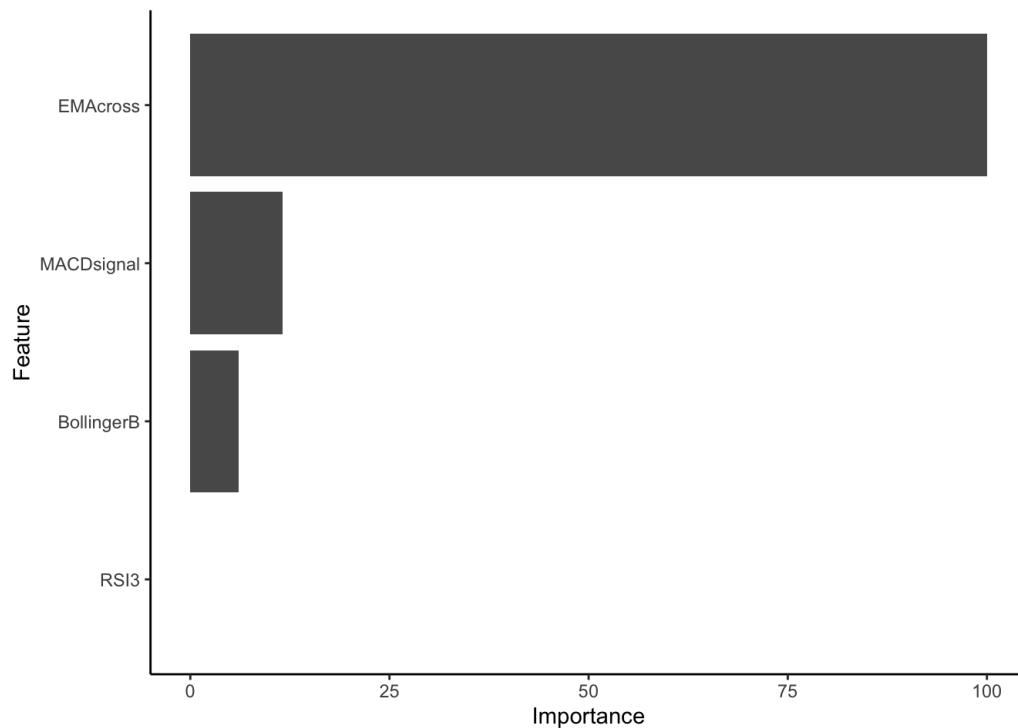
99% accuracy is achieved to predict the price using the stock technical indicators.

0.9974791

- Plotting nnet variable importance

```
nnet variable importance
```

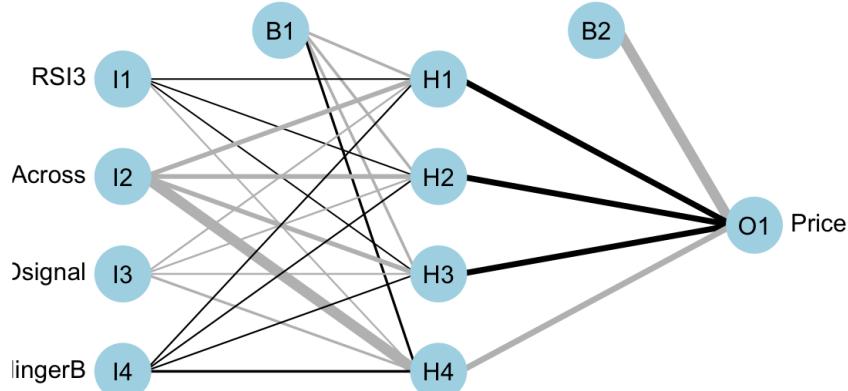
	Overall
EMAcross	100.000
MACDsignal	11.556
BollingerB	6.033
RSI3	0.000



#### Graphical Representation of

our Neural Network

## Graphical Representation of our Neural Network



## Conclusion

From the above implementation, the results are impressive(99% accuracy) and convincing in terms of using a machine learning algorithm to decide on the price of the stock Majority of the attributes in the dataset contribute significantly to the building of a predictive model.

## Using SVM to predict price change for WALMART (Real Time Analysis)

### Machine Learning: Classification using SVM

- SVM is another classification method that can be used to predict if a client falls into either 'yes' or 'no' class.
  - The linear, polynomial and RBF or Gaussian kernel in SVM are simply different in case of making the hyperplane decision boundary between the classes.
  - The kernel functions are used to map the original dataset (linear/nonlinear ) into a higher dimensional space with view to making it linear dataset.
  - Usually linear and polynomial kernels are less time consuming and provides less accuracy than the rbf or Gaussian kernels.
  - The k cross validation is used to divide the training set into k distinct subsets. Then every subset is used for training and others k-1 are used for validation in the entire trainging phase. This is done for the better training of the classification task.Overall, if you are unsure which kernel method would be best, a good practice is use of something like 10-fold cross-validation for each training set and then pick the best algorithm.

```
colnames(DataSetWalmart)
1          Class
2          RSI
3      EMAcross
4          MACD
5          SMI
6          WPR
7          ADX
8          CCI
9          CMO
10         ROC
```

### SVM Classifier using Linear Kernel

- Caret package provides train() method for training our data for various algorithms. We just need to pass different parameter values for different algorithms. Before train() method, we will first use trainControl() method.

- We are setting 3 parameters of trainControl() method. The “method” parameter holds the details about resampling method. We can set “method” with many values like “boot”, “boot632”, “cv”, “repeatedcv”, “LOOCV”, “LGOCV” etc. For this project, let’s try to use repeatedcv i.e, repeated cross-validation.
- The “number” parameter holds the number of resampling iterations. The “repeats” parameter contains the complete sets of folds to compute for our repeated cross-validation. We are using setting number =10 and repeats =3. This trainControl() methods returns a list. We are going to pass this on our train() method.
- Before training our SVM classifier, set.seed().
- For training SVM classifier, train() method should be passed with “method” parameter as “svmLinear”. We are passing our target variable Term\_Deposit. The “Term\_Deposit.” denotes a formula for using all attributes in our classifier and Term\_Deposit. as the target variable. The “trControl” parameter should be passed with results from our trianControl() method. The “preProcess” parameter is for preprocessing our training data.
- As discussed earlier for our data, preprocessing is a mandatory task. We are passing 2 values in our “preProcess” parameter “center” & “scale”. These two help for centering and scaling the data. After preProcessing these convert our training data with mean value as approximately “0” and standard deviation as “1”. The “tuneLength” parameter holds an integer value. This is for tuning our algorithm.

#### Support Vector Machines with Linear Kernel

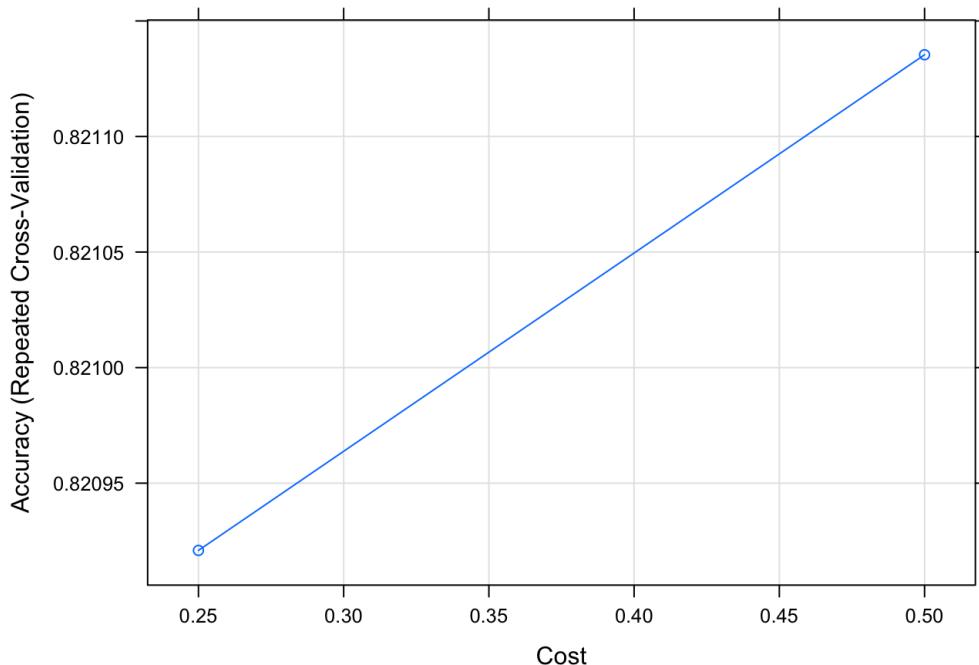
```
18651 samples
  7 predictor
  2 classes: 'DOWN', 'UP'

Pre-processing: centered (7), scaled (7)
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 16786, 16786, 16786, 16786, 16786, 16786, ...
Resampling results across tuning parameters:

  C      Accuracy   Kappa
  0.25  0.8209209  0.6412469
  0.50  0.8211354  0.6416789
```

Accuracy was used to select the optimal model using the largest value.  
The final value used for the model was C = 0.5.

- The above model is showing that our classifier is giving best accuracy on C = 0.25 Let’s try to make predictions using this model for our test set and check its accuracy.



```

predictionsvm DOWN UP
    DOWN    28   8
    UP      5  28

```

- Accuracy on the test set by train control is 81% using C=0.25.

```
0.8115942
```

#### Confusion Matrix and Statistics

```

        Reference
Prediction DOWN UP
    DOWN    28   8
    UP      5  28

        Accuracy : 0.8116
        95% CI  : (0.6994, 0.8957)
No Information Rate : 0.5217
P-Value [Acc > NIR] : 5.253e-07

        Kappa : 0.6239

McNemar's Test P-Value : 0.5791

        Sensitivity : 0.8485
        Specificity : 0.7778
Pos Pred Value : 0.7778
Neg Pred Value : 0.8485
        Prevalence : 0.4783
Detection Rate : 0.4058
Detection Prevalence : 0.5217
Balanced Accuracy : 0.8131

'Positive' Class : DOWN

```

## SVM Classifier using Non-Linear Kernel

- Now, we will try to build a model using Non-Linear Kernel like Radial Basis Function. For using RBF kernel, we just need to change our train() method's "method" parameter to "svmRadial". In Radial kernel, it needs to select proper value of Cost "C" parameter and "sigma" parameter.

```

Support Vector Machines with Radial Basis Function Kernel

18651 samples
  7 predictor
  2 classes: 'DOWN', 'UP'

Pre-processing: centered (7), scaled (7)
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 16786, 16786, 16786, 16786, 16786, ...
Resampling results across tuning parameters:

      C      Accuracy     Kappa
0.25  0.8565757  0.7127826
0.50  0.8618302  0.7233515

Tuning parameter 'sigma' was held constant at a value of 0.5
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.5 and C = 0.5.

```

- SVM-RBF kernel calculates variations and will present us best values of sigma & C. Based on the output best values of sigma= 0.5 & C=0.5 Let's check our trained models' accuracy on the test set.

0.7971014

- Final prediction accuracy on the test set is 0.7971014

## Comparision between SVM models

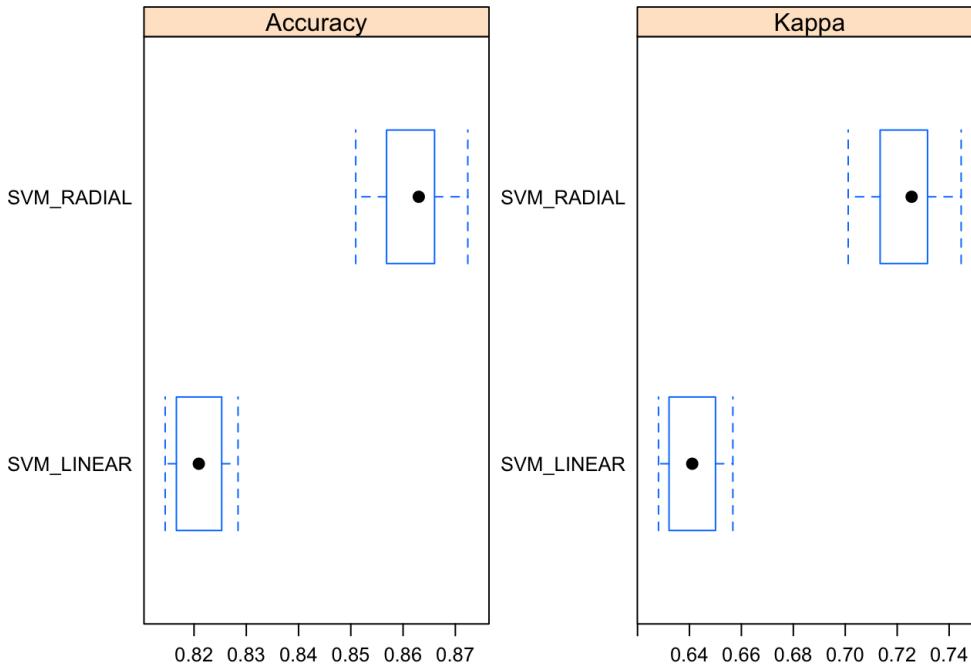
- Comparision between SVM Linear and Radial Models.

```
Call:
summary.resamples(object = algo_results)

Models: SVM_RADIAL, SVM_LINEAR
Number of resamples: 10

Accuracy
      Min.   1st Qu.     Median       Mean   3rd Qu.   Max. NA's
SVM_RADIAL 0.8509383 0.8571046 0.8630027 0.8618302 0.8657375 0.8723861 0
SVM_LINEAR 0.8144772 0.8171582 0.8209115 0.8211354 0.8248692 0.8284182 0

Kappa
      Min.   1st Qu.     Median       Mean   3rd Qu.   Max. NA's
SVM_RADIAL 0.7011356 0.7139651 0.7255579 0.7233515 0.7312067 0.7446252 0
SVM_LINEAR 0.6280912 0.6334363 0.6410743 0.6416789 0.6492305 0.6567207 0
```



## Conclusion

From the above implementation, the results are impressive and convincing in terms of using a machine learning algorithm to decide on the price change of walmart . Majority of the attributes in the dataset contribute significantly to the building of a predictive model. All the two SVM approach achieves good accuracy rate(>80%) and are easier to implement.

## Naive Bayes Sentimental Analysis of Twitter Data for Apple Stock

- We will be trying to understand sentiment of tweets about the company Apple, By using the twitter for better understand public perception, Apple wants to monitor how people feel over time and how people receive new announcements.

- Our challenge in this project is to see if we can correctly classify tweets as being negative, positive, or neither about Apple.

- **Sentiment Mining - Apple Stock**

- Apple wants to monitor how people feel about them over time, and how people receive new announcements.
- Challenge: Can we correctly classify tweets as being negative, positive, or neither about Apple?

- **Preprocessing of Text data**

- Change all the words in words in lower or upper
- remove punctuation
- remove stop words
- stemming

- **Data**

To collect the data needed for this task, we had to perform two steps.

- Collect Twitter data
  - Construct the outcome variable for these tweets, which means that we have to label them as positive, negative, or neutral sentiment. These outcomes were represented as a number on the scale from -2 to 2.

The following graph shows the distribution of the number of tweets classified into each of the categories. We can see here that the majority of tweets were classified as neutral, with a small number classified as strongly negative or strongly positive.

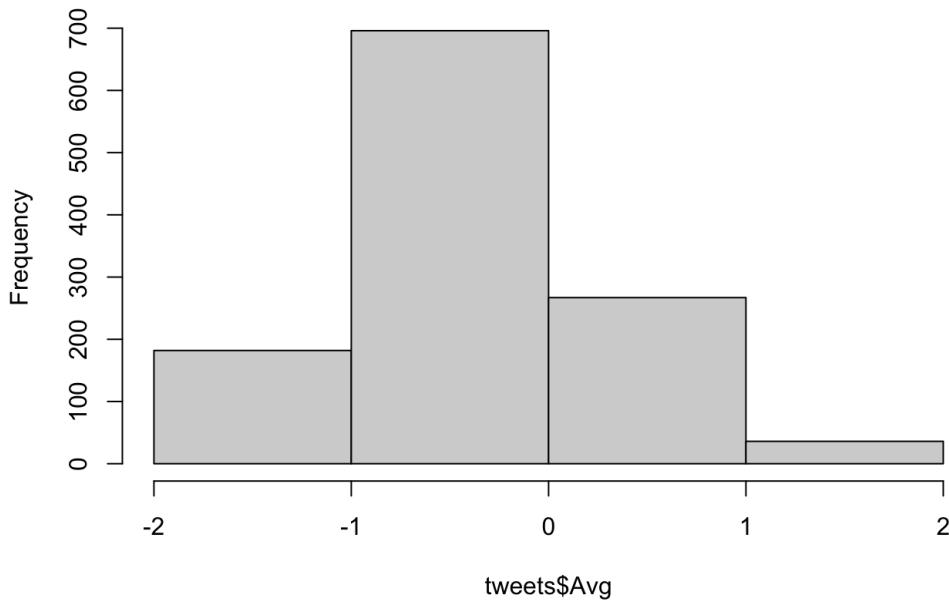
```

Tweet
1 I have to say, Apple has by far the best customer care service I have ever received! @Apple @AppStore
2                               iOS 7 is so fricking smooth & beautiful!! #ThanxApple @Apple
3                                         LOVE U @APPLE
4 Thank you @apple, loving my new iPhone 5S!!!! #apple #iphone5S pic.twitter.com/XmHJCU4pcb
5 .@apple has the best customer service. In and out with a new phone in under 10min!
6 @apple ear pods are AMAZING! Best sound from in-ear headphones I've ever had!

Avg
1 2.0
2 2.0
3 1.8
4 1.8
5 1.8
6 1.8

```

**Histogram of tweets\$Avg**



## Creating Corpus

- One of fundamental concepts in text analysis, implemented in the package tm as well, is that of a corpus. A corpus is a collection of documents.
- We will need to convert our tweets to a corpus for pre-processing. Various function in the tm package can be used to create a corpus in many different ways.
- We will create it from the tweet column of our data frame using two functions, Corpus() and VectorSource(). We feed to this latter the Tweets variable of the tweets data frame.

- **Pre-processing steps**

- To deal with text data following pre-processing is required. Follow the standard steps to build and pre-process the corpus:

- Build a new corpus variable called corpus.
- Using tm\_map, convert the text to lowercase.
- Using tm\_map, remove all punctuation from the corpus.
- Using tm\_map, remove all English stopwords from the corpus.
- Using tm\_map, stem the words in the corpus.

- Build a document term matrix from the corpus, called dtm. Each operation, like stemming or removing stop words, can be done with one line in R, where we use the tm\_map() function which takes as its first argument the name of a corpus and as second argument a function performing the transformation that we want to apply to the text.

## Positive Bag Of Words

- Create a Document Term Matrix
- We are now ready to extract the word frequencies to be used in our prediction problem. The tm package provides a function called DocumentTermMatrix() that generates a matrix where:
  - the rows correspond to documents, in our case tweets, and
  - the columns correspond to words in those tweets.
  - The values in the matrix are the number of times that word appears in each document.
- Create word cloud for positive tweets

```
'data.frame': 61 obs. of 1 variable:
 $ text: chr " say appl far best custom care servic ever receiv appl appstor" "io frick smooth beauti th
anxappl appl" "love u appl" "thank appl love new iphon s appl iphon pictwittercomxmhjcupcb" ...
```



## Negative Bag Of Words & wordcloud for negative



- ***Building machine learning model***

- Before Building the machine learning model, we need to split our data in training and testing sets
- Lastly, let's split our data into a training set and a testing set, putting 70% of the data in the training set.

- ***Making predictions***

- We build Naive Bayes by using training & test data sets.
- We apply Laplace smoothing , which is a technique for smoothing categorical data.
- A small-sample correction, or pseudo-count, will be incorporated in every probability estimate. Consequently, no probability will be zero. this is a way of regularizing Naive Bayes, and when the pseudo-count is zero, it is called Laplace smoothing.

```
model <- naive_bayes(as.data.frame(as.matrix(dtm_train)), dataset_train$positive, laplace = 1)
```

## Interpretation of the results and prediction accuracy achieved

- ***Evaluate the model performance using confusionMatrix***

- The accuracy of our model on the testing set is 78%.
- We can visualise the model's performance using a confusion matrix.
- We can evaluate the accuracy, precision and recall on the training and validation sets to evaluate the performance of naive bayes algorithm.

## Confusion Matrix and Statistics

```
Reference
Prediction FALSE TRUE
  FALSE    237     0
  TRUE      65     0

  Accuracy : 0.7848
  95% CI : (0.7341, 0.8298)
No Information Rate : 1
P-Value [Acc > NIR] : 1

  Kappa : 0

Mcnemar's Test P-Value : 2.051e-15

  Sensitivity : 0.7848
  Specificity :      NA
  Pos Pred Value :      NA
  Neg Pred Value :      NA
  Prevalence : 1.0000
  Detection Rate : 0.7848
Detection Prevalence : 0.7848
  Balanced Accuracy :      NA

'Positive' Class : FALSE
```

- Evaluate the model performance using CrossTable

```
Cell Contents
+-----+
|          N |
+-----|
```

Total Observations in Table: 302

```
| dataset_test$positive
model_predict |      FALSE | Row Total |
+-----+-----+-----+
  FALSE |      237 |      237 |
+-----+-----+-----+
  TRUE |       65 |       65 |
+-----+-----+-----+
Column Total |      302 |      302 |
+-----+-----+-----|
```

- Find 25 frequent words

```
"appl"  "iphon" "new"   "phone" "whi"   "get"   "ipad"  "itun"  "ipod"
```

## Overall insights obtained from the implemented project

- Overall accuracy of the model is 78%. It is safe to assume that naive bayes models can be trained on to find the sentiment of the stocks.
- Sensitivity for finding ratings is 0.7848.

- Since the dataset was clean, and reviews are equally distributed between test & training set, adding laplace smoothing factor did not make much difference in the accuracy.

## Amazon Price Trend Predict - Random Forest (Real Time Analysis)

- Stock market prediction is an incredibly difficult task, due to the randomness and noisiness found in the market. Yet, predicting market behaviors is a very important task. Correctly predicting stock price directions can be used to maximize profits, as well as to minimize risk. There are two types of methods to predicting market behavior. One is predicting the future price of an asset. This is usually done using time series analysis to fit a specific model, like ARIMA or GARCH, to some historical data. The other is predicting the future trend of an asset. That is, whether one thinks it will go up or down in price, treating it as a classification problem.
- The goal of this project is to create an intelligent model, using the Random Forest model, that can correctly forecast the behavior of a stock's price n days out.

### ***Data Import***

- Use “quantmod” package to download information for Amazon stocks.
- The data used for this project consists of regular stock data (open, close, volume, etc.) from Yahoo finance, and ranges from the year 2000 to 2018.
- From this data, technical indicators were calculated for every stock. Below are all the technical indicators used for this model:

```

- Relative Strength Index
- Stochastic Oscillator
- William %R
- Moving Average Convergence Divergence
- Price Rate of Change
- On Balance Volume

```

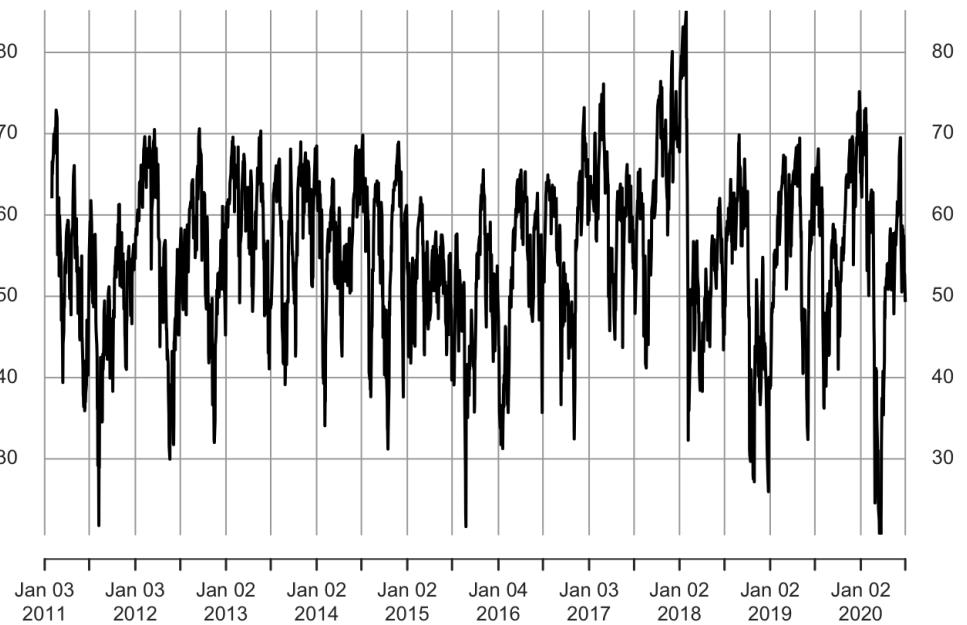
- The last step of pre-processing the data was calculating the response variable. Since we are treating this as a classification problem, the response variable was binary. The equation for calculating the response variable is below: Response=Close<sub>t+n</sub>-Close<sub>t</sub>
- It states that the adjusted close price at t+n, where n is the number of days out you want to predict, minus the current adjusted close price will map to a value that says the stock price went up from the point at time t, or that it went down.

## Methodology

```
"^GSPC"
```

**relativeStrengthIndex20**

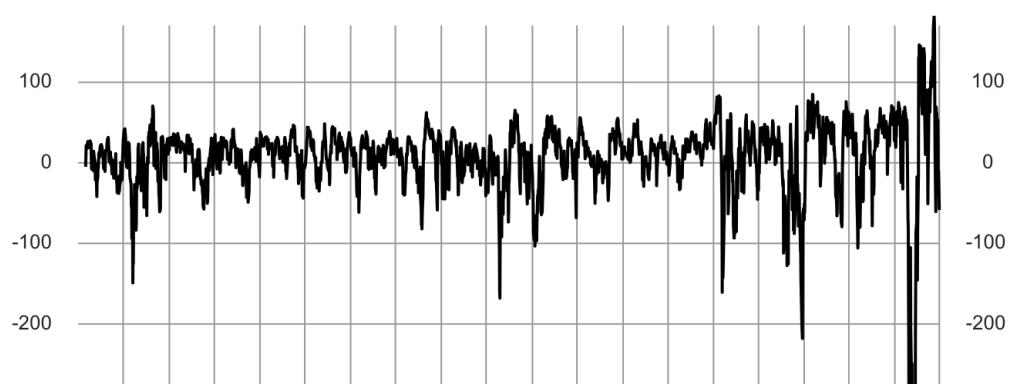
2011-01-03 / 2020-06-29

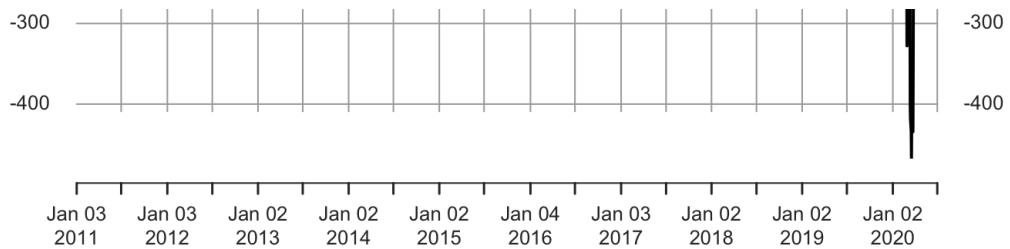
**exponentialMovingAverage20**

2011-01-03 / 2020-06-29

**exponentialMovingAverageDiff**

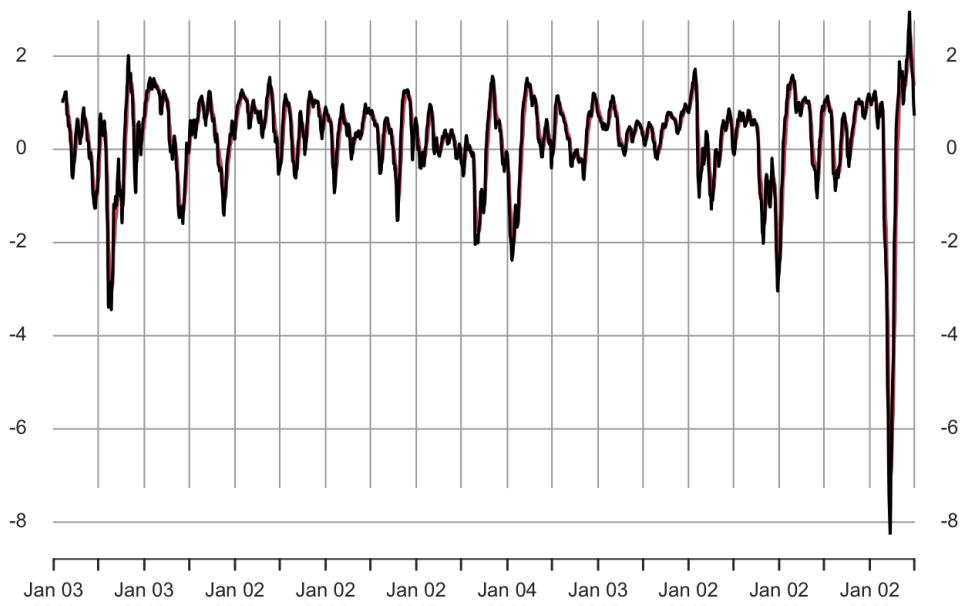
2011-01-03 / 2020-06-29





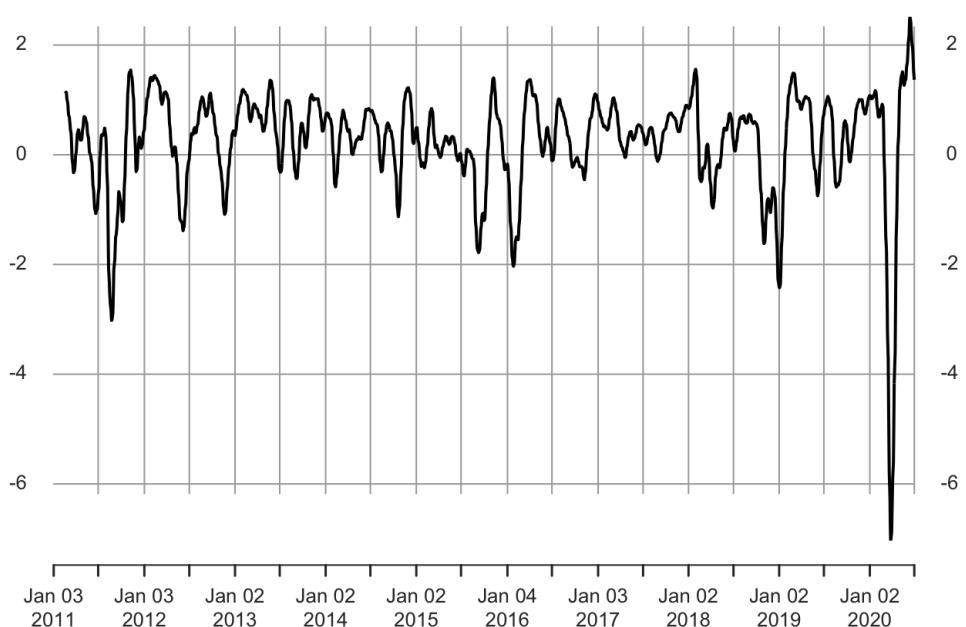
**MACD**

2011-01-03 / 2020-06-29



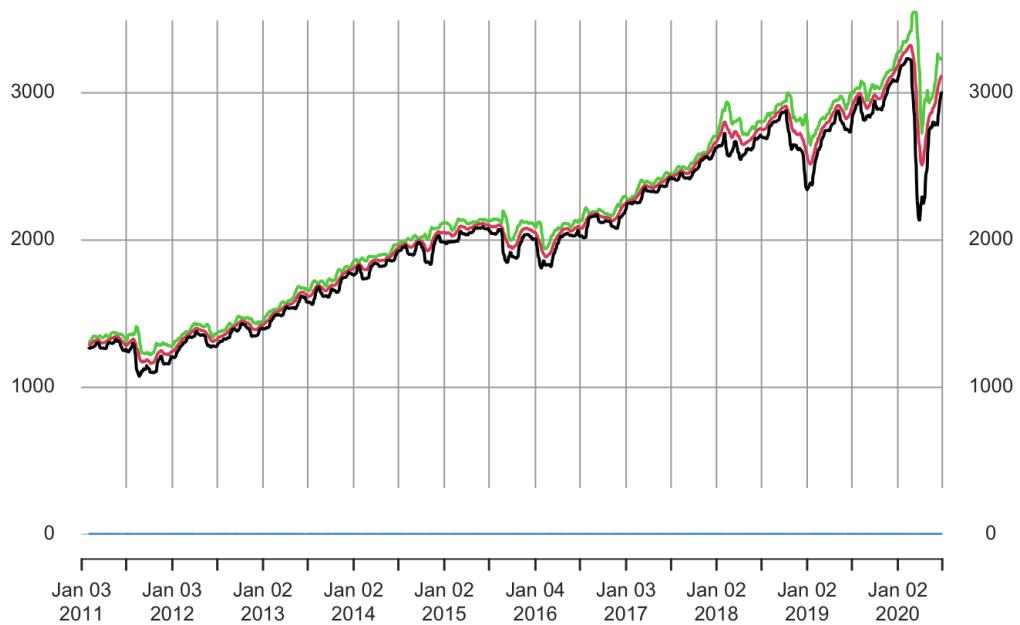
**MACDsignal**

2011-01-03 / 2020-06-29

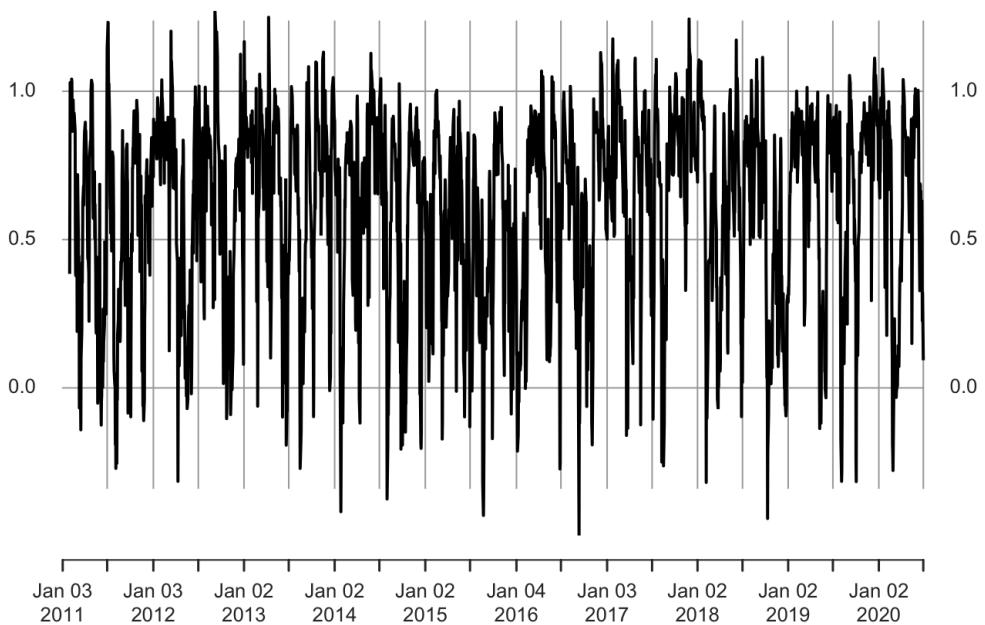


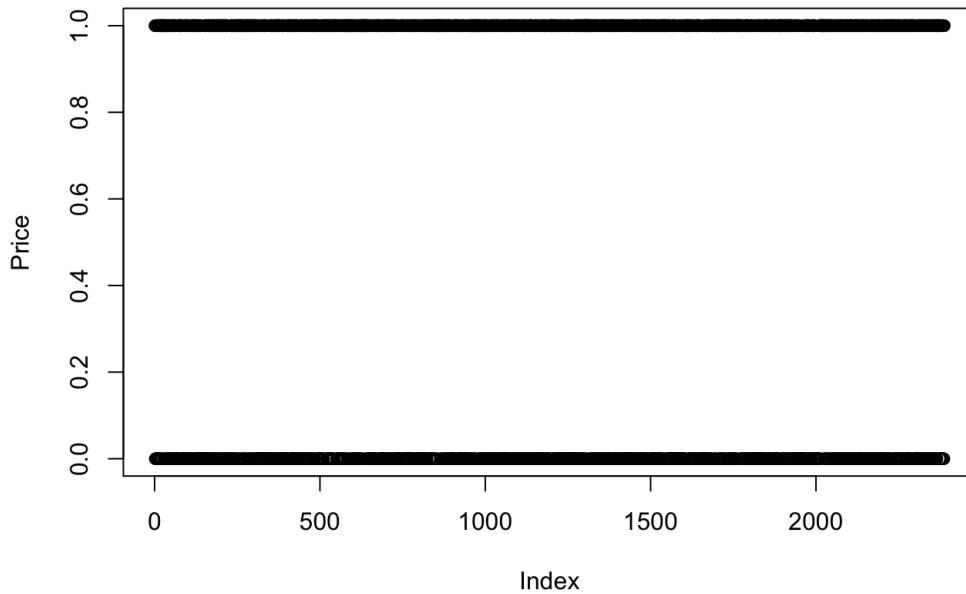
**BollingerBands**

2011-01-03 / 2020-06-29

**PercentageChngpctB**

2011-01-03 / 2020-06-29





- Dataset used for this project:

```
colnames(dataset1)
1           rsi
2           EMA
3       signal
4      pctB
5 GSPC.Close
```

- Check for missing data in relativeStrengthIndex20, exponentialMovingAverage20, MACDsignal, PercentageChngpctB, Price. Omit the n/a values in dataset.
- Print the number of missed value for each attribute in dataset.

```
"rsi"
20
"EMA"
19
"signal"
33
"pctB"
19
"GSPC.Close"
0
```

- Random Forest Machine Learning Model
  - Split the dataset into training & test dataset. 80 % of the data is training data. 20 % of the data is test dataset.
  - Feature Scaling -> Normalization /Scale and dropping the feature variables.

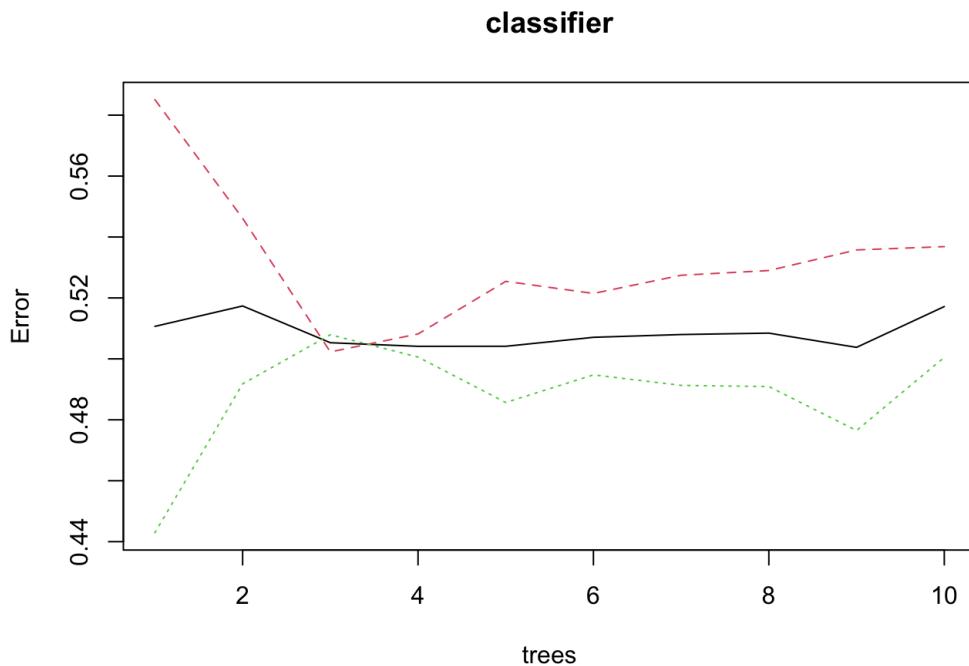
```
# Feature Scaling (Normalization and dropping the predicted variable)
training_set[-5] = scale(training_set[-5])
test_set[-5] = scale(test_set[-5])
```

- Applying Random Forest Model on the Training set.

```

Length Class Mode
call           4 -none- call
type           1 -none- character
predicted     1884 factor numeric
err.rate       30 -none- numeric
confusion      6 -none- numeric
votes          3768 matrix numeric
oob.times     1884 -none- numeric
classes         2 -none- character
importance      4 -none- numeric
importanceSD    0 -none- NULL
localImportance 0 -none- NULL
proximity       0 -none- NULL
ntree           1 -none- numeric
mtry            1 -none- numeric
forest          14 -none- list
y               1884 factor numeric
test            0 -none- NULL
inbag           0 -none- NULL

```



## Prediction & Accuracy.

- After building the model, then we can check the accuracy of forecasting using confusion matrix.
  - Final Model Accuracy of the model is 51 percent.

```

predict_val
 0   1
0  89 128
1 102 152

```

```
"Model Accuracy is"
```

```
0.5116773
```

## Ensemble Method for Random Forest

## Random Forest

```
1884 samples
  4 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 1257, 1256, 1255
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
  3     0.5058141 -0.003131317
  6     0.5021045 -0.009128875
  9     0.5132502  0.013836823

Kappa was used to select the optimal model using the largest value.
The final value used for the model was mtry = 9.
```

## Overall insights obtained from the implemented project

- As we can see, the model has the highest accuracy of 51 percent . While this may not seem any good, it is often extremely hard to predict the price of stocks. Even the 2.5% improvement over random guessing can make a difference given the amount of money at stake. After all, if it was that easy to predict the prices, wouldn't we all be trading in stocks for the easy money instead of learning these algorithms?

## Discussion and Conclusions

- This is a beginning of the use of ML algorithms for predicting the time series data or the stock prices. It can be modified and optimized in a lot of ways to produce much better and much more efficient and accurate results.
- Choosing the right technical indicators which will influence the price change is daunting. In this project we tried to predict the price change through variety of technical indicators in different ML algorithms.
- Although we have achieved accuracy of 99 percent in few ML algorithms, there are many other things that impact the prices of stocks such as: Political and social upheavals ,current affairs etc
- Thus, we can say stock market price change is quite a dynamic movement.