



Ministry of Communication
and Information Technology



Intensive Programs on Communications and Information Technology (3 Months)

Track
Big Data Science

Project Name
Stock Price Prediction
by Algorithms Trading

December 2018

Project Title
Stock Price Prediction
By Algorithms Trading

Team member
Eng. Mohamed Saber
Eng. Amir Zaghloul
Eng. Yara Sabry
Eng. Amr Abd Alaziz

Under The Supervision of
Dr. Doaa Hassan
Eng. Sayed Omer

Technical Report Outline

Chapter 1 Introduction

- 1.1 Machine Learning in Finance.
- 1.2 Stock price prediction.
- 1.3 Prediction methods
 - 1.3.1 Fundamental analysis
 - 1.3.2 Technical analysis (charting)
 - 1.3.3 Technological methods (Machine Learning)
- 1.4 Algorithmic Trading concept

Chapter 2 Background and Theoretical Overview

- 2.1 ARIMA Model for forecasting stock price
 - 2.1.1 Auto Regression (AR)
 - 2.1.2 Differencing (I)
 - 2.1.3 Moving Average (MA)
- 2.2 Box-Jenkins Model Approach

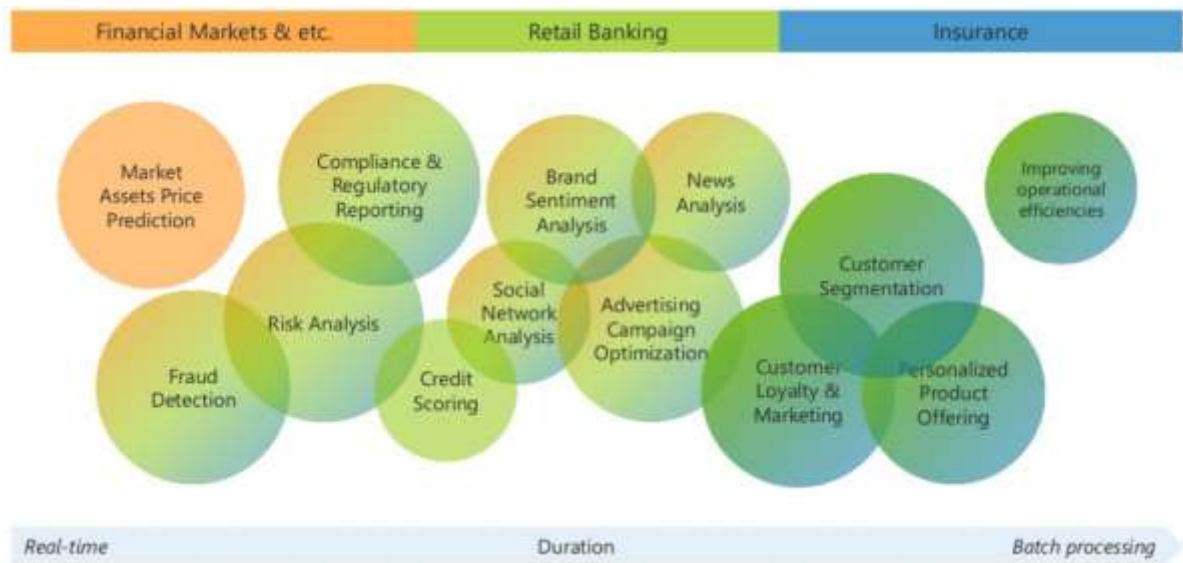
Chapter 3 Proposed Project Implementation

- 3.1 Framework architecture
- 3.2 Analysis of stock price data by Python
- 3.3 Predict of stock price data by R
- 3.4 Stock price by Shiny API
- 3.5 Algorithmic Trading (Quant Develop)

References

Chapter 1 Introduction

1.1 Machine Learning in Finance



Machine learning has had fruitful applications in finance well before the advent of mobile banking apps, proficient chat bots, or search engines. Given high volume, accurate historical records, and quantitative nature of the finance world, few industries are better suited for artificial intelligence. There are more uses cases of machine learning in finance than ever before, a trend perpetuated by more accessible computing power and more accessible machine learning tools (such as Google's Tensorflow). Today, machine learning has come to play an integral role in many phases of the financial ecosystem, from approving loans, to managing assets, to assessing risks. Yet, few technically-savvy professionals have an accurate view of just how many ways machine learning finds its way into their daily financial lives.

Machine Learning in Finance current Applications:

Portfolio Management	Algorithmic Trading
Fraud Detection	Loan / Insurance Underwriting
Customer Service	Security financial
Sentiment / News Analysis	
Sales / Recommendations of Financial Products	

1.2 Stock price prediction intro.

Stock price prediction is the act of trying to determine the future value of a company stock or other instrument traded on an exchange. The successful prediction of a stock's future price could yield significant profit. The efficient-market hypothesis suggests that stock prices reflect all currently available information and any price changes that are not based on newly revealed information thus are inherently unpredictable. Others disagree and those with this viewpoint possess myriad methods and technologies which purportedly allow them to gain future price information.

While the efficient market hypothesis finds favor among financial academics, its critics point to instances in which actual market experience differs from the prediction-of-unpredictability the hypothesis implies. A large industry has grown up around the implication proposition that some analysts can predict stocks better than others; ironically that would be impossible under the Efficient Markets Hypothesis if the stock prediction industry did not offer something its customers believed to be of value.

1.3 Prediction methods.

Prediction methodologies fall into three broad categories which can (and often do) overlap. They are fellow:

Fundamental analysis

Technical analysis (charting)

Technological methods (Machine Learning)



1.3.1 Fundamental analysis:



Fundamental Analysts are concerned with the company that underlies the stock itself. They evaluate a company's past performance as well as the credibility of its accounts. Many performance ratios are created that aid the fundamental analyst with assessing the validity of a stock, such as the P/E ratio. Warren Buffett is perhaps the most famous of all Fundamental Analysts.

Fundamental analysis is built on the belief that human society needs capital to make progress and if a company operates well, it should be rewarded with additional capital and result in a surge in stock price. Fundamental analysis is widely used by fund managers as it is the most reasonable, objective and made from publicly available information like **financial statement analysis**. Another meaning of fundamental analysis is beyond bottom-up company analysis, it refers to top-down analysis from first analyzing the global economy, followed by country analysis and then sector analysis, and finally the company level analysis.

1.3.2 Technical analysis (Charting)

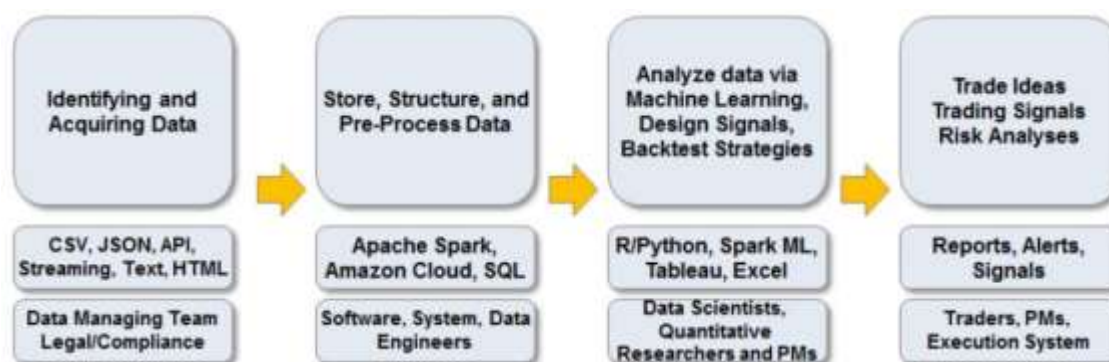


Technical analysts or chartists are not concerned with any of the company's fundamentals. They seek to determine the future price of a stock based solely on the trends of the past price (a form of **time series analysis**). Numerous patterns are employed such as the head and shoulders or cup and saucer. Alongside the patterns, techniques are used such as the exponential **moving average** (EMA). **Candle stick** patterns, believed to have been first developed by Japanese rice merchants, are nowadays widely used by technical analysts.

1.3.3 Technological methods (Machine Learning)

With the advent of the digital computer, stock market prediction has since moved into the technological realm. The most prominent technique involves the use of **artificial neural networks** (ANNs) and **Genetic Algorithms** (GA). Scholars found bacterial chemotaxis optimization method may perform better

than GA. ANNs can be thought of as mathematical function approximations. The most common form of ANN in use for stock market prediction is the feed forward network utilizing the backward propagation of errors algorithm to update the network weights. These networks are commonly referred to as **Backpropagation networks**. Another form of ANN that is more appropriate for stock prediction is the time **recurrent neural network** (RNN) or **time delay neural network** (TDNN). Examples of RNN and TDNN are the Elman, Jordan, and Elman-Jordan networks.



For stock prediction with ANNs, there are usually two approaches taken for **forecasting different time horizons**: independent and joint. The independent approach employs a single ANN for each time horizon, for example, 1-day, 2-day, or 5-day. The advantage of this approach is that network forecasting error for one horizon won't impact the error for another horizon since each time horizon is typically a unique problem. The joint approach, however, incorporates multiple time horizons together so that they are determined simultaneously. In this approach, forecasting error for one time horizon may share its error with that of another horizon, which can decrease performance. There are also more parameters required for a joint model, which increases the risk of overfitting. of late, the majority of academic research groups studying ANNs for stock forecasting seem to be using an ensemble of independent ANNs methods more frequently, with greater success. An ensemble of ANNs would use low price and time lags to predict future lows, while another network would use lagged highs to predict future highs. The predicted low and high predictions are then used to form stop prices for buying or selling. Outputs from the individual "low" and "high" networks can also be input into a final network that would also incorporate volume, intermarket data or statistical summaries of prices, leading to a final

ensemble output that would trigger buying, selling, or market directional change. A major finding with ANNs and stock prediction is that a classification approach (vs. function approximation) using outputs in the form of buy($y=+1$) and sell($y=-1$) results in better predictive reliability than a quantitative output such as low or high price. since NNs require training and can have a large parameter space; it is useful to optimize the network for optimal predictive ability.

1.4 Algorithmic Trading concept

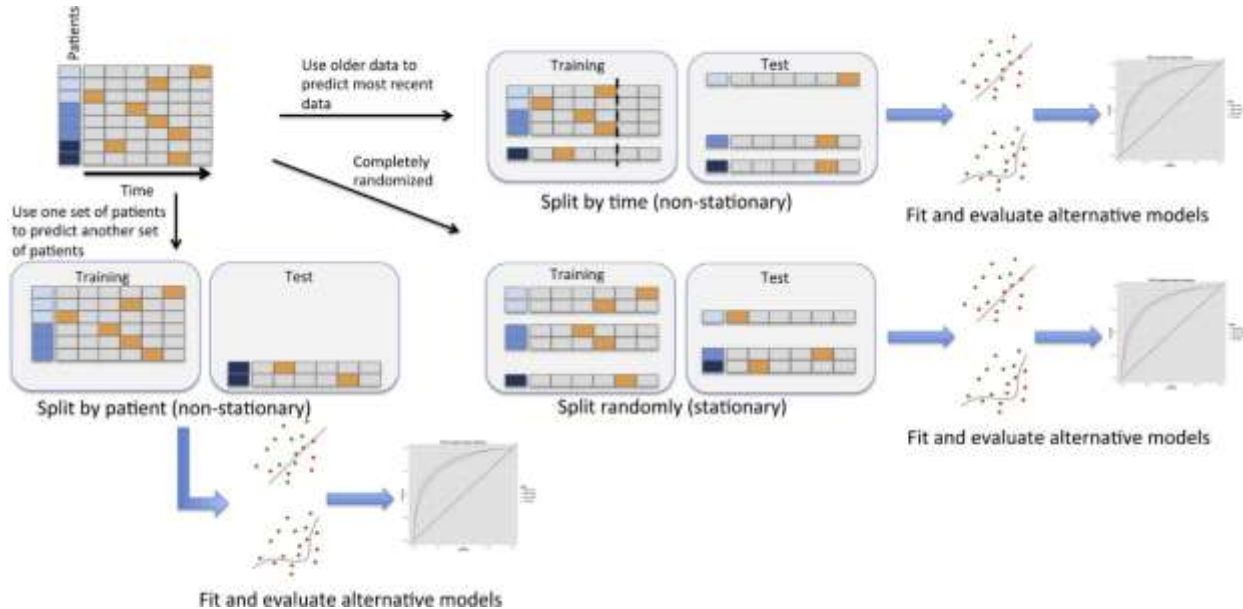
Algorithmic trading (automated trading, black-box trading or simply algo-trading) is the process of using computers programed to follow a defined set of instructions (an algorithm) for placing a trade in order to generate profits at a speed and frequency that is impossible for a human trader. The defined sets of rules are based on timing, price, quantity or any mathematical model. Apart from profit opportunities for the trader, algo-trading makes markets more liquid and makes trading more systematic by ruling out the impact of human emotions on trading activities.

Algorithmic Trading Strategies:

1. Trend-following Strategies
2. Arbitrage Opportunities
3. Mathematical Model Based Strategies
4. Trading Range (Mean Reversion)
5. Volume Weighted Average Price (VWAP)
6. Time Weighted Average Price (TWAP)
7. Percentage of Volume (POV)
8. Implementation Shortfall

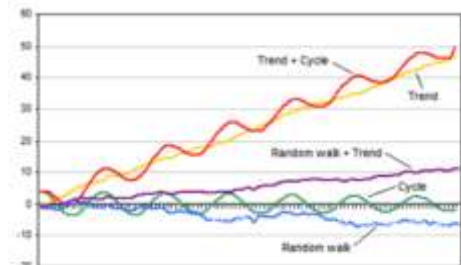
Chapter 2 Background and Theoretical Overview

2.1 ARIMA Model for forecasting stock price in r



Forecasting involves predicting values for a variable using its historical data points or it can also involve predicting the change in one variable given the change in the value of another variable. Forecasting approaches are primarily categorized into qualitative forecasting and quantitative forecasting. Time series forecasting falls under the category of quantitative forecasting wherein statistical principals and concepts are applied to a given historical data of a variable to forecast the future values of the same variable. Some time series forecasting techniques used **include**:

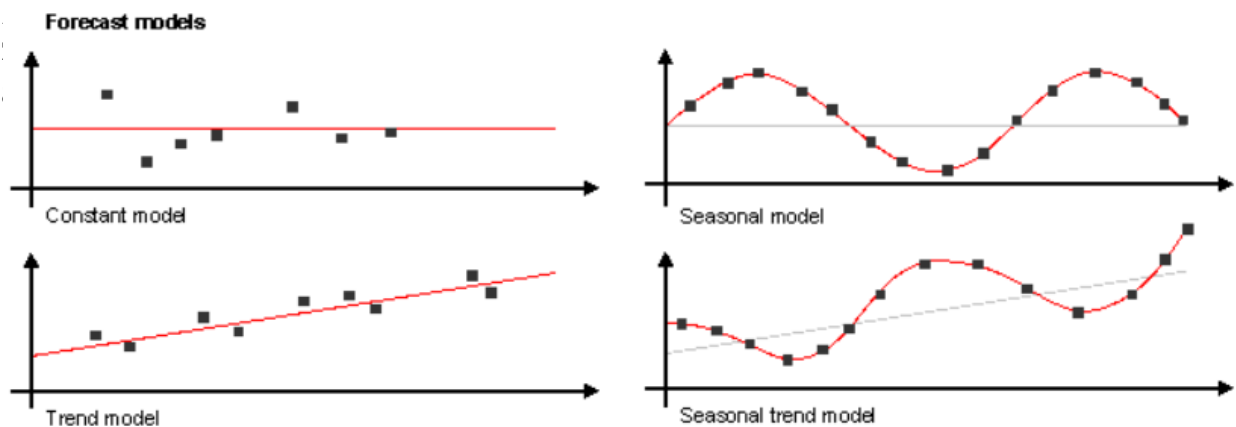
- Autoregressive Models (AR)
- Moving Average Models (MA)
- Seasonal Regression Models
- Distributed Lags Models



ARIMA stands for Autoregressive Integrated Moving Average. ARIMA is also known as Box-Jenkins approach. Box and Jenkins claimed that non-stationary data can be made stationary by differencing the series, Y_t . The general model for Y_t is written as

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} \dots \phi_p Y_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \dots \theta_q \epsilon_{t-q}$$

Where, Y_t is the differenced time series value, ϕ and θ are unknown



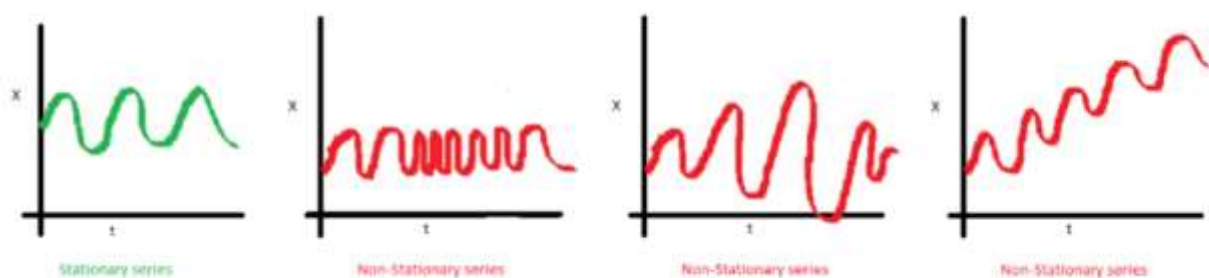
The ARIMA model combines three basic methods:

- **Auto Regression (AR)** – In auto-regression the values of a given time series data are regressed on their own lagged values, which is indicated by the “p” value in the model.
- **Differencing (I-for Integrated)** – This involves differencing the time series data to remove the trend and convert a non-stationary time series to a stationary one. This is indicated by the “d” value in the model. If $d = 1$, it looks at the difference between two time series entries, if $d = 2$ it looks at the differences of the differences obtained at $d = 1$, and so forth.
- **Moving Average (MA)** – The moving average nature of the model is represented by the “q” value which is the number of lagged values of the error term.

A non seasonal ARIMA model is classified as an "ARIMA(p,d,q)" model, where:

- p is the number of autoregressive terms,
- d is the number of non seasonal differences needed for stationarity, and
- q is the number of lagged forecast errors in the prediction equation.

• **Stationary Series:** A stationary series has no trend, its variations around its mean have a constant amplitude. A non stationary series is made stationary by differencing



	ACF	PACF
AR(p)	Die out	Cut off after the order p of the process
MA(q)	Cut off after the order q of the process	Die out
ARMA(p,q)	Die out	Die out

Note that:

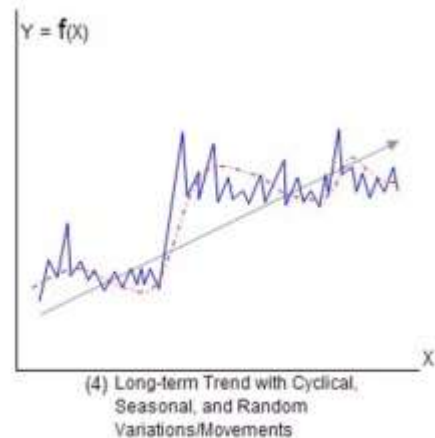
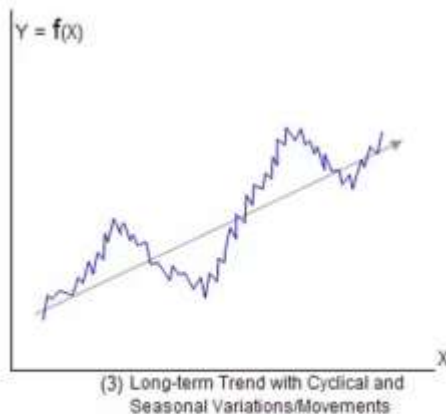
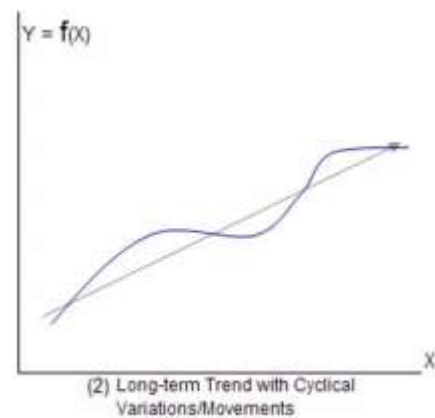
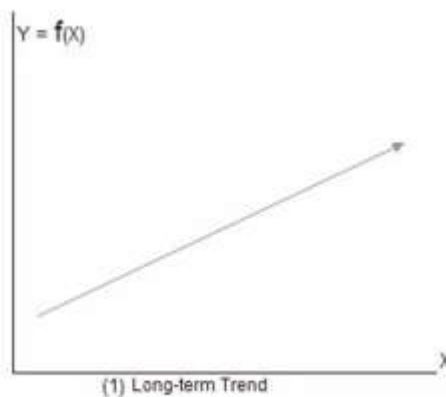
- ARMA(p,0) = AR(p)
- ARMA(0,q) = MA(q)

In practice, the values of p and q each **rarely exceed 2**.

In this context...

- **“Die out”** means “tend to zero gradually”
- **“Cut off”** means “disappear” or “is zero”

$$Y_t = \phi_0 + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \dots + \phi_p Y_{t-p} + \varepsilon_t - \omega_1 \varepsilon_{t-1} - \omega_2 \varepsilon_{t-2} - \dots - \omega_q \varepsilon_{t-q}$$



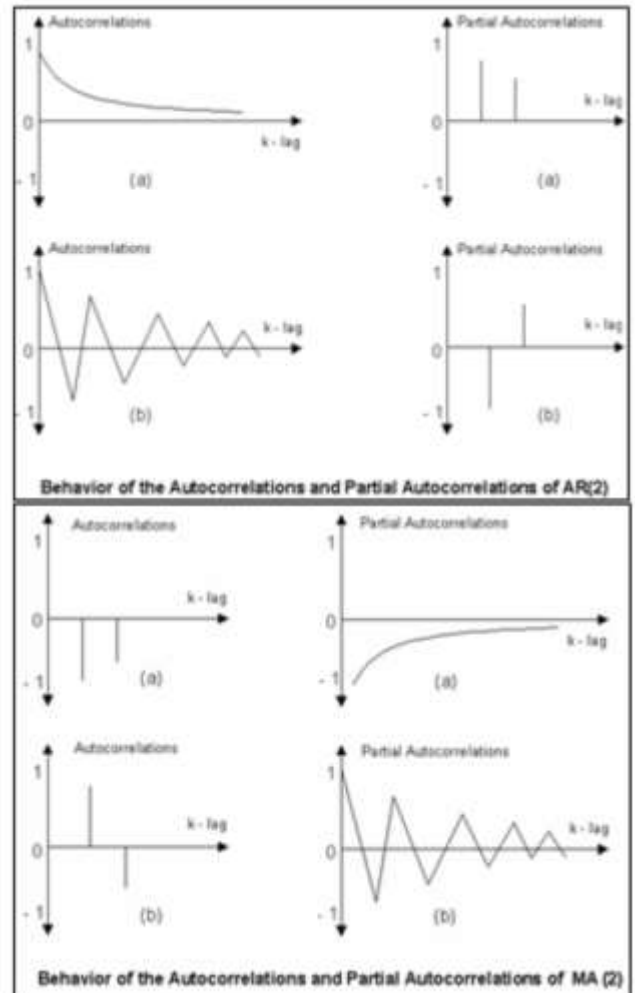
Summary on AR, MA and ARMA

	ACF	PACF
AR(p)	Die out	Cut off after the order p of the process
MA(q)	Cut off after the order q of the process	Die out
ARMA(p,q)	Die out	Die out

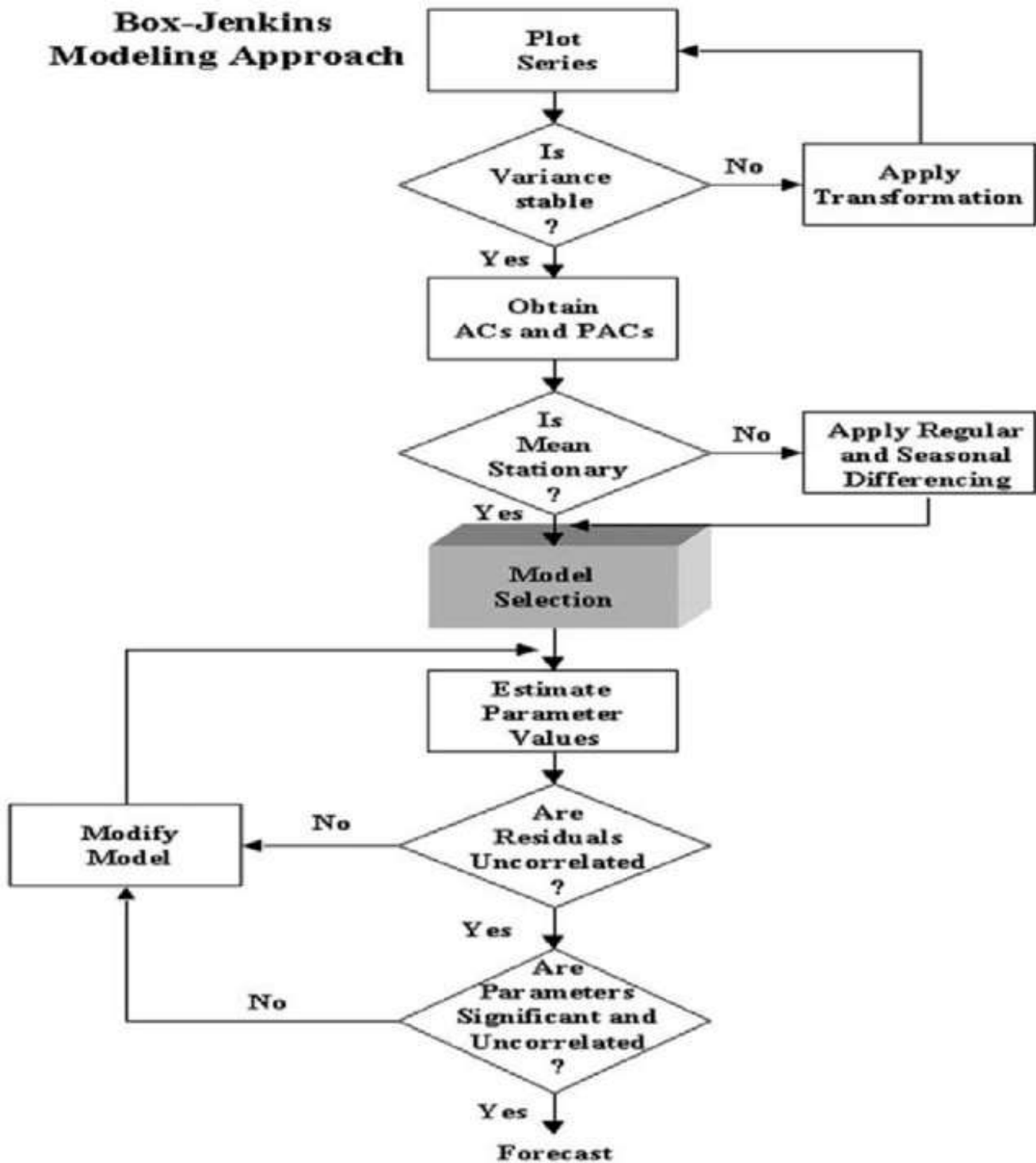
In this context...

"Die out" means "tend to zero gradually"

"Cut off" means "disappear" or "is zero"

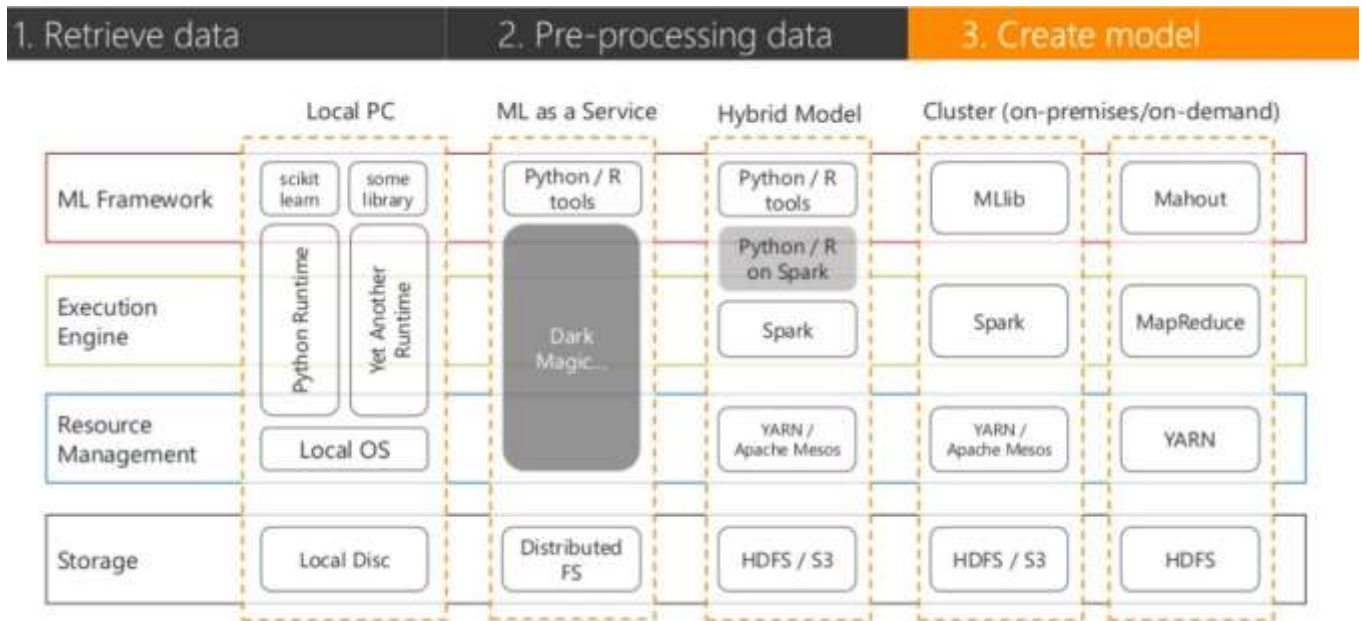


Box-Jenkins Modeling Approach



Chapter 3 Proposed Project Implementation

3.1 Framework architecture



3.2 Analysis of stock price data by Python

I will be using two packages, [quandl](#) and [pandas_datareader](#), which are not installed with Anaconda if you are using it. To install these packages, run the following at the appropriate command prompt:

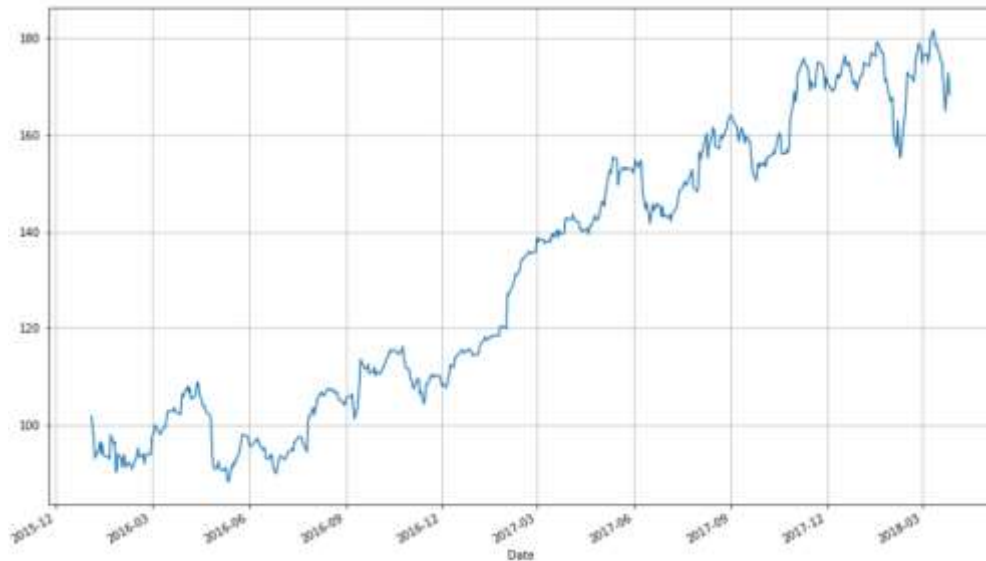
```
1 | conda install quandl
2 | conda install pandas-datareader
```

Getting Data from Quandl

Before we analyze stock data, we need to get it into some workable format. Stock data can be obtained from [Yahoo! Finance](#), [Google Finance](#), or a number of other sources. These days I recommend getting data from [Quandl](#), a provider of community-maintained financial and economic data. ([Yahoo! Finance](#) used to be the go-to source for good quality stock data)

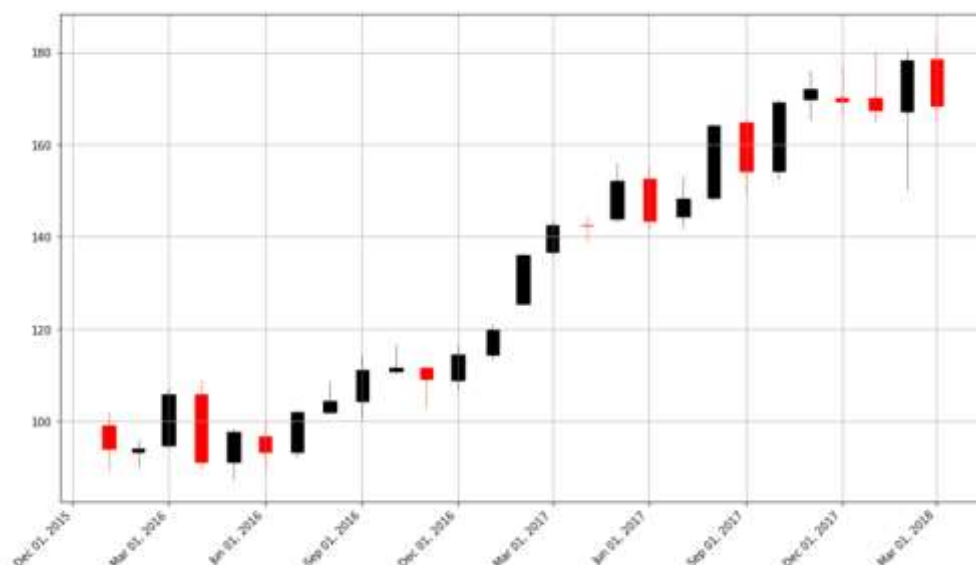
Visualizing Stock Data

Now that we have stock data we would like to visualize it. I first demonstrate how to do so using the matplotlib package. Notice that the apple DataFrame object has a convenience method, `plot()`, which makes creating plots easier.



Candlestick Plot

A line chart is fine, but there are at least four variables involved for each date (open, high, low, and close), and we would like to have some visual way to see all four variables that does not require plotting four separate lines. Financial data is often plotted with a Japanese candlestick plot, so named because it was first created by 18th century Japanese rice traders. Such a chart can be created with matplotlib, though it requires considerable effort.

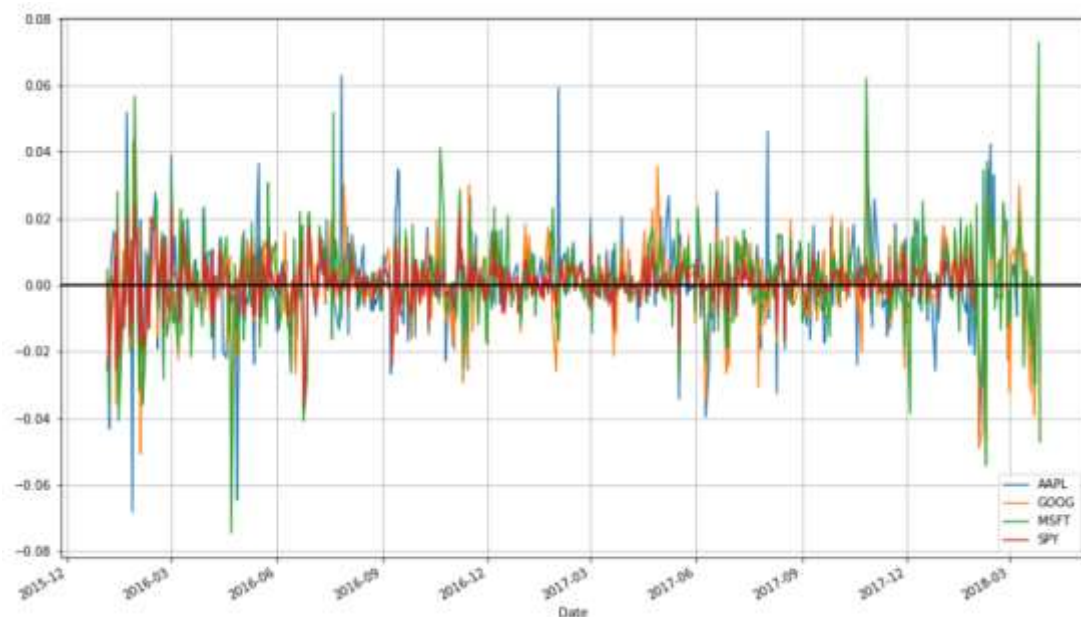


Compare stocks with other companies and marking (Spyder):

We may wish to plot multiple financial instruments together; we may want to compare stocks, compare them to the market, or look at other securities such as exchange-traded funds (ETFs). Later, we will also want to see how to plot a financial instrument against some indicator, like a moving average. For this you would rather use a line chart than a candlestick chart



Non-Station and not Trend



Moving Averages

Charts are very useful. In fact, some traders base their strategies almost entirely off charts (these are the "technicians", since trading strategies based off finding patterns in charts is a part of the trading doctrine known as technical analysis). Let's now consider how we can find trends in stocks.



3.3 Prediction of stock price data by R

There are a number of packages available for time series analysis and forecasting. We load the relevant R package for time series analysis and pull the stock data from yahoo finance.

Load Data and Plot

```
install.packages("quantmod")
install.packages("tseries")
install.packages("timeSeries")
install.packages("forecast")
install.packages("xts")

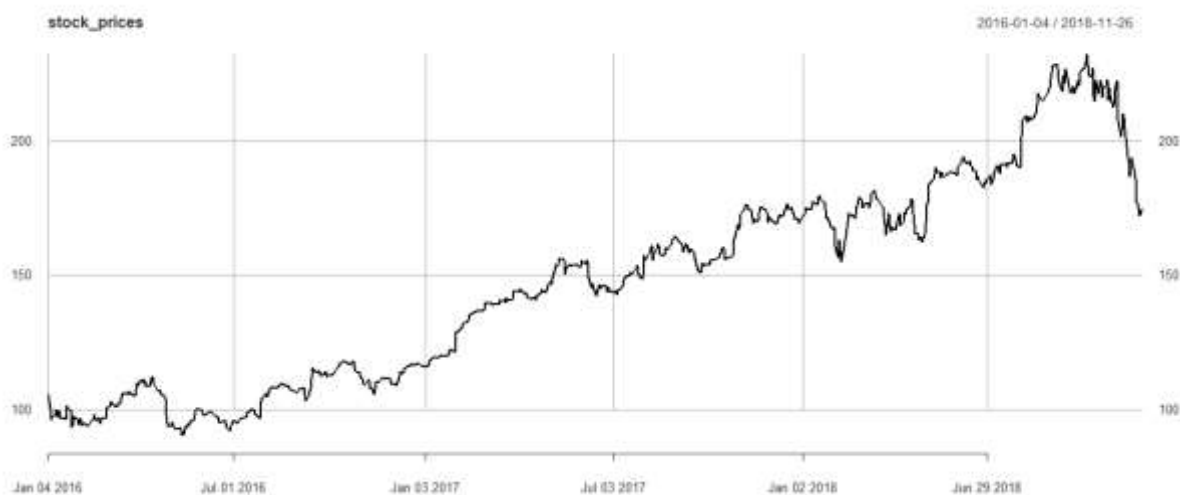
library(quantmod)
library(tseries)
library(timeSeries)
library(forecast)
library(xts)

# Get data from Yahoo finance
getSymbols('AAPL',src='yahoo',from='2012-01-01')
plot(AAPL)

# select the adj. close price
stock_prices = AAPL[,4]

# Compute Differencing (I) - to remove(trend) and convert a (non-stationary) to (stationary)
stock = diff(log(stock_prices),lag=1)
stock = stock[!is.na(stock)]

# test check stationary or not by (ADF test) p-value = 0.01 < 0.05 then stationary
print(adf.test(stock))
```



Create Object series

```
#-----  
  
# Split the dataset in two parts - training and testing  
splitpoint = floor(nrow(stock)*(0.90))  
  
#-----  
  
# Create Actual series  
Actual_series = xts(0,as.Date("2018-11-21", "%Y-%m-%d"))  
  
# Create Forecast series  
forecasted_series = data.frame(Forecasted = numeric())  
forecasted_series = xts(forecasted_series, index(Actual_series))  
  
#-----  
  
# xts: create object continuous time series
```

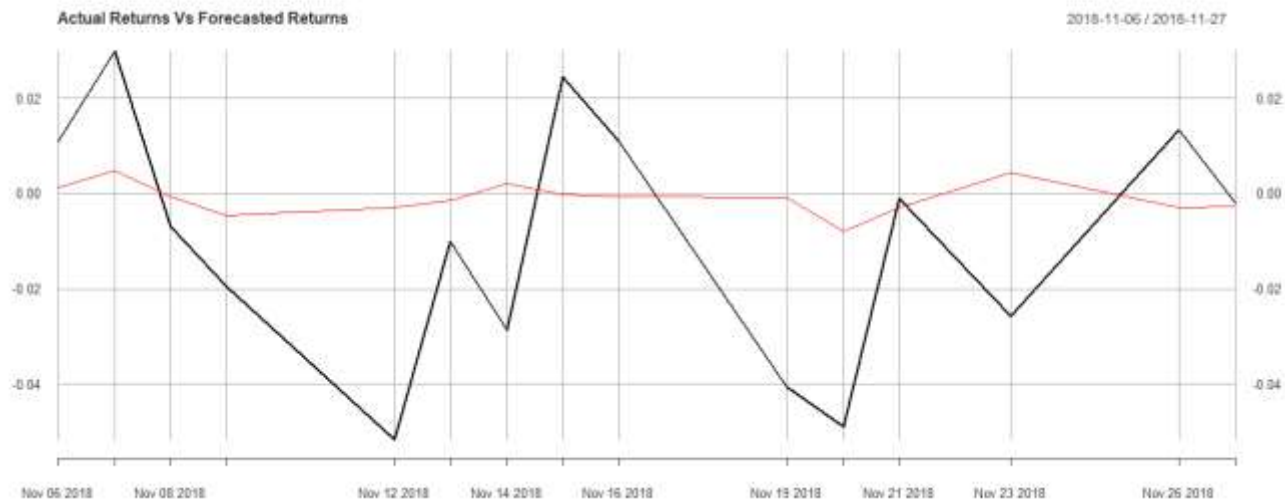
ARIMA Model (Split, Fit and Forecast)

```
# Check all adj. close by for loop  
for (b in splitpoint:(nrow(stock)-1)) {  
  
  # --- Split Data -----  
  
  stock_train = stock[1:b, ] # start from first row to 677  
  stock_test = stock[(b+1):nrow(stock), ] # start from 677 to 752  
  
  # --- Fit ARIMA (p,d,q) -----  
  
  fit = arima(stock_train, order = c(2, 0, 2), include.mean=FALSE)  
  
  # --- Forecasting -----  
  
  arima.forecast = forecast(fit, h = 1, level=99)  
  
}
```

Create Table

```
# --- Create Table accuracy of forecast -----  
  
compare = merge(Actual_series, forecasted_series)  
compare$Accuracy = sign(compare$Actual_series) == sign(compare$Forecasted)  
print(compare)  
# sign == sign > 1 true | sign != sign > 0 false compare  
  
# Compute the accuracy percentage metric  
Accuracy_percentage = sum(compare$Accuracy == 1)*100/length(compare$Accuracy)  
print(Accuracy_percentage)
```

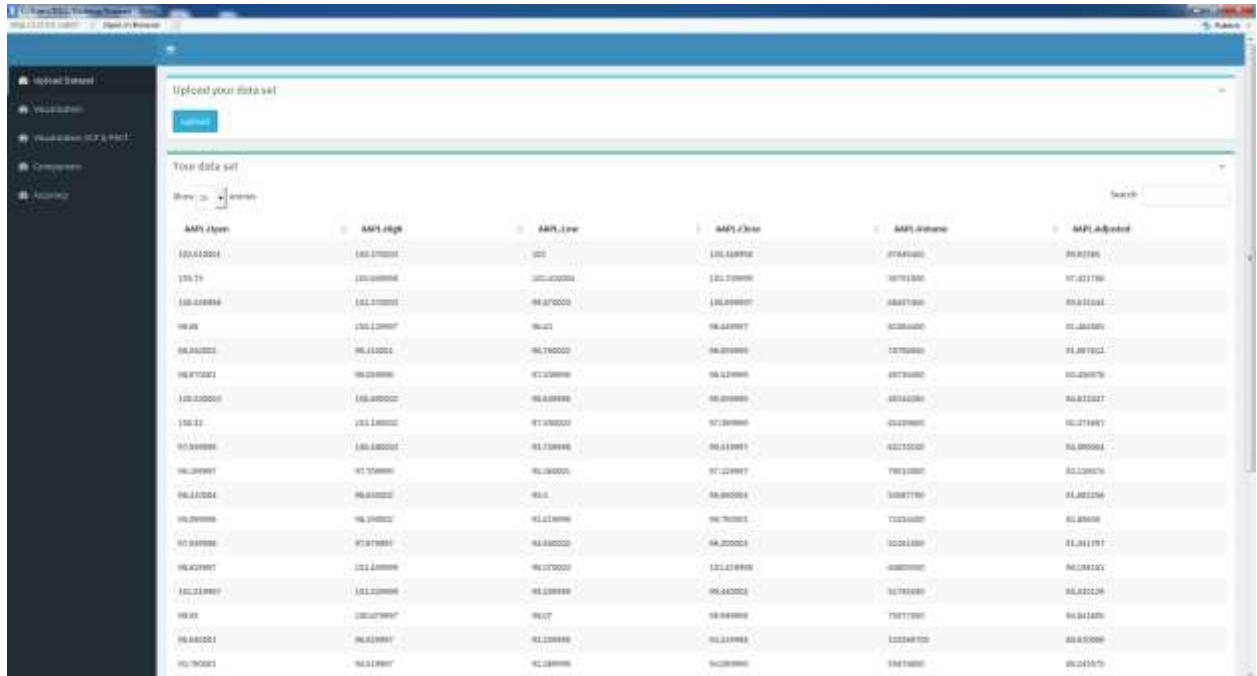
Result



```
> print(compare)
      Actual_series  Forecasted Accuracy
2018-11-07    0.029877466    0.0047873639      1
2018-11-08   -0.006978290   -0.0007427484      1
2018-11-09   -0.019469832   -0.0046477572      1
2018-11-12   -0.051687217   -0.0029073390      1
2018-11-13   -0.010041503   -0.0014819823      1
2018-11-14   -0.028654009    0.0020738424      0
2018-11-15    0.024379203   -0.0003112249      0
2018-11-16    0.011014789   -0.0006007427      0
2018-11-19   -0.040438827   -0.0008811561      1
2018-11-20   -0.048956998   -0.0078965686      1
2018-11-21   -0.001130693   -0.0030117929      1
2018-11-23   -0.025726953    0.0042695753      0
2018-11-26    0.013433093   -0.0029912314      0
2018-11-27   -0.002178468   -0.0027149454      1
> # Compute the accuracy percentage metric
> Accuracy_percentage = sum(comparsion$Accuracy == 1)*100/length(comparsion$Accuracy)
> print(Accuracy_percentage)
[1] 64.28571
```

3.4 Stock Price Prediction by Shiny R API

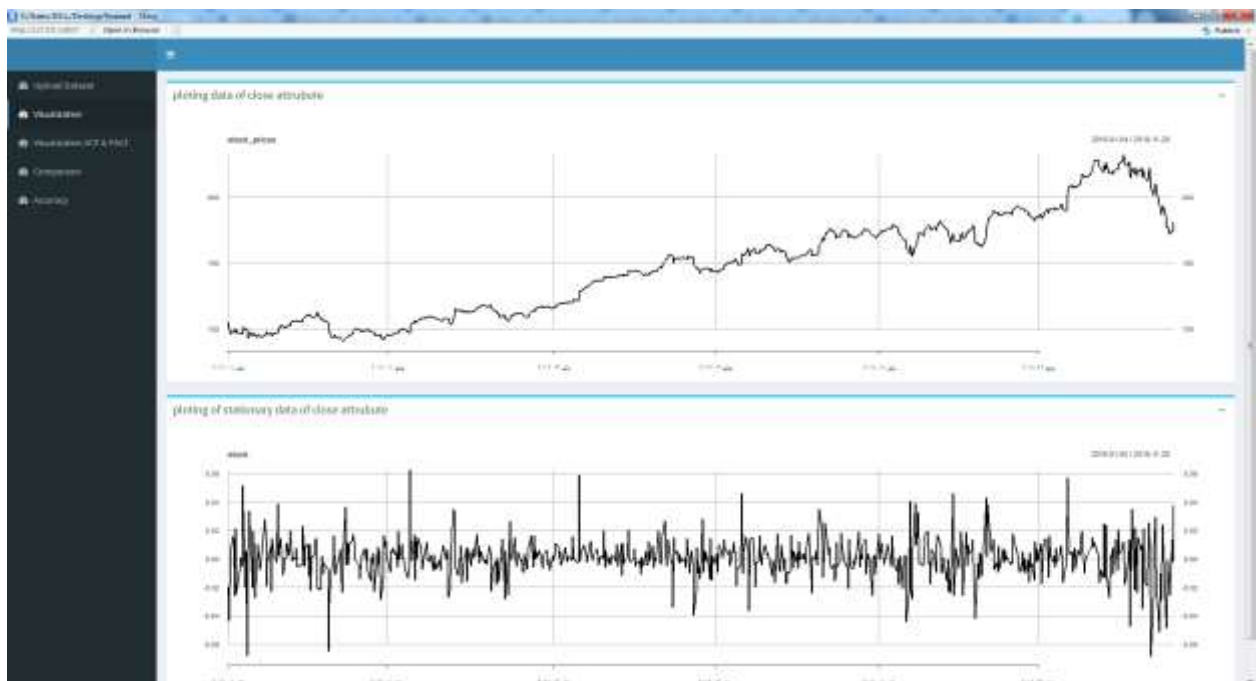
Load Data



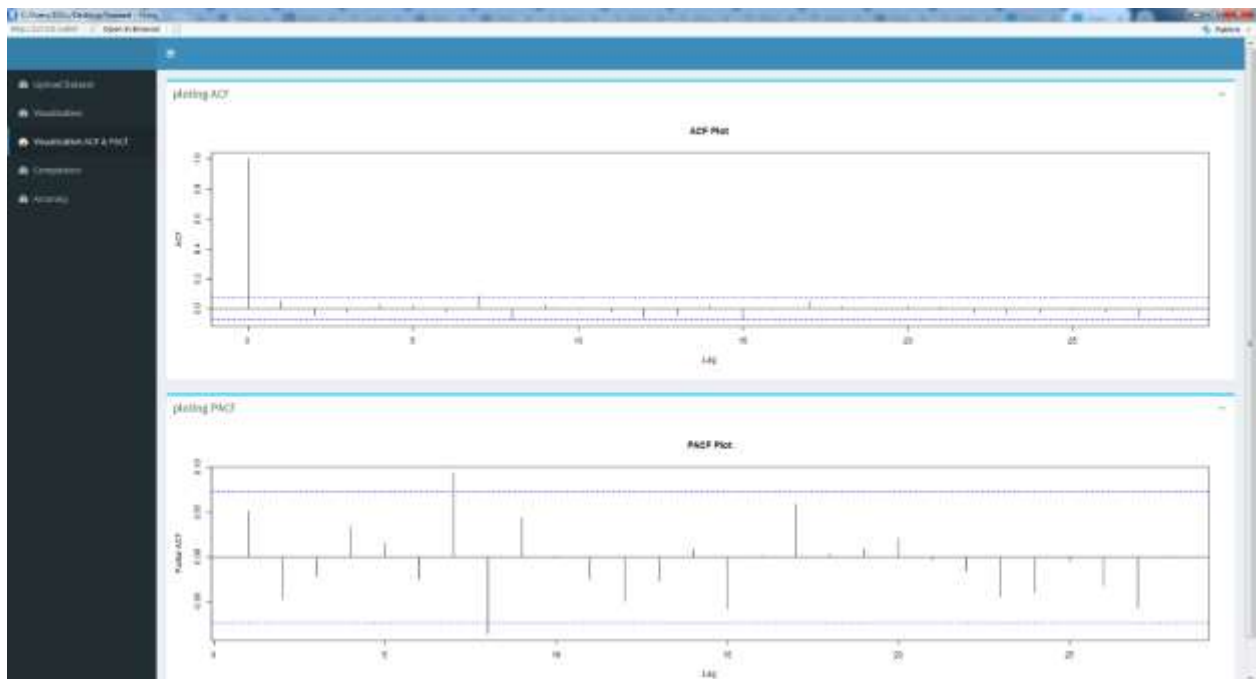
The screenshot shows a Shiny R application interface. On the left is a sidebar with navigation links: "Upload Dataset", "Visualization", "Visualization: ACF & PACF", "Computation", and "Accuracy". The main panel is titled "Upload your data set" and contains a table of stock data. The table has columns for AAPL Open, AAPL High, AAPL Low, AAPL Close, AAPL Volume, and AAPL Adjusted. The data spans from 2015 to 2019.

AAPL Open	AAPL High	AAPL Low	AAPL Close	AAPL Volume	AAPL Adjusted
120.810003	120.870003	120	120.850003	37943400	99.83286
120.75	120.840004	120.420004	120.730003	16713500	97.421786
120.880004	121.070005	120.870005	120.880005	26607300	99.833264
99.95	100.120007	99.83	99.920007	82354000	97.383359
99.860003	99.910004	99.760003	99.850003	73703000	97.387852
99.870003	99.920004	99.780003	99.870003	49734000	97.394979
100.320005	100.480006	100.380006	100.380006	30543200	99.833267
100.32	100.480006	100.380006	100.380006	42224000	99.8374957
97.990006	98.080007	97.980006	97.990006	42270200	98.990064
98.290007	98.350008	98.280007	98.290007	79073000	98.128979
98.310004	98.360005	98.3	98.350005	30687700	98.882256
98.390006	98.450007	98.380006	98.420006	22254000	98.88406
97.990006	98.080007	97.980006	97.990006	32261300	98.881197
98.420007	98.480008	98.410007	98.420007	40801000	98.148343
100.320005	100.480006	100.380006	100.380006	30786000	99.833269
99.95	100.120007	99.83	99.920007	75077300	99.843859
99.860003	99.910004	99.760003	99.850003	122648700	99.833966
99.760003	99.810004	99.680003	99.780003	39470000	99.845579

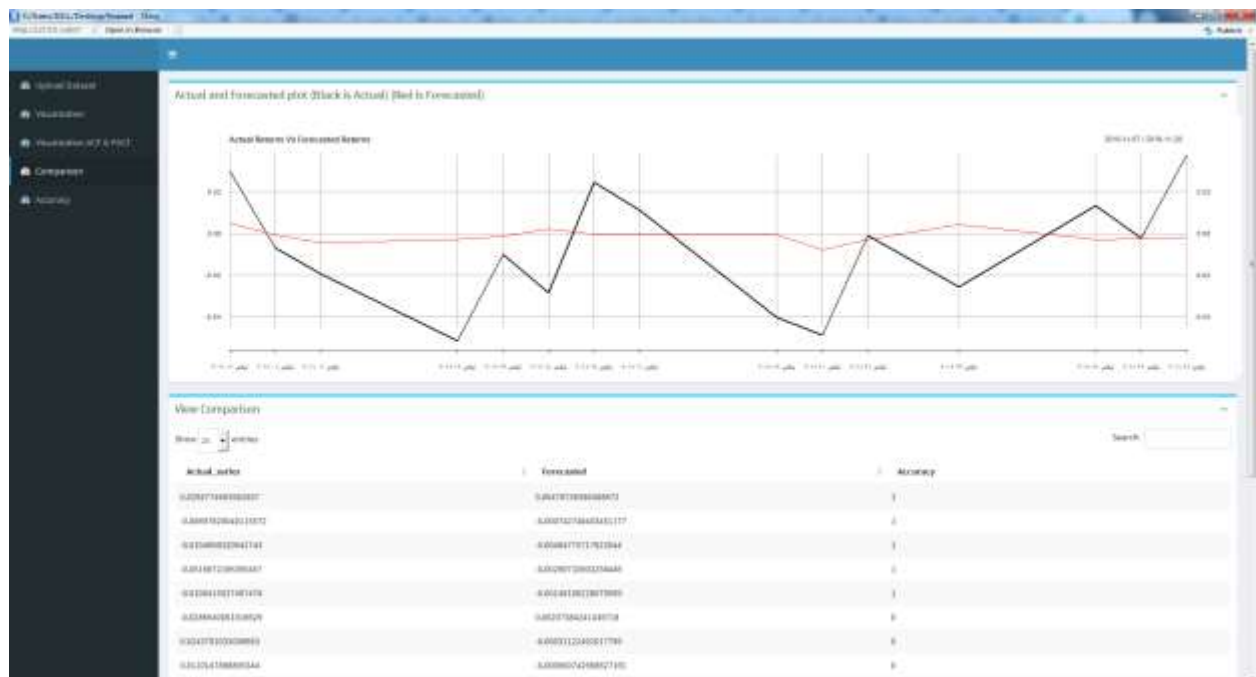
Visualization before stationary and after it



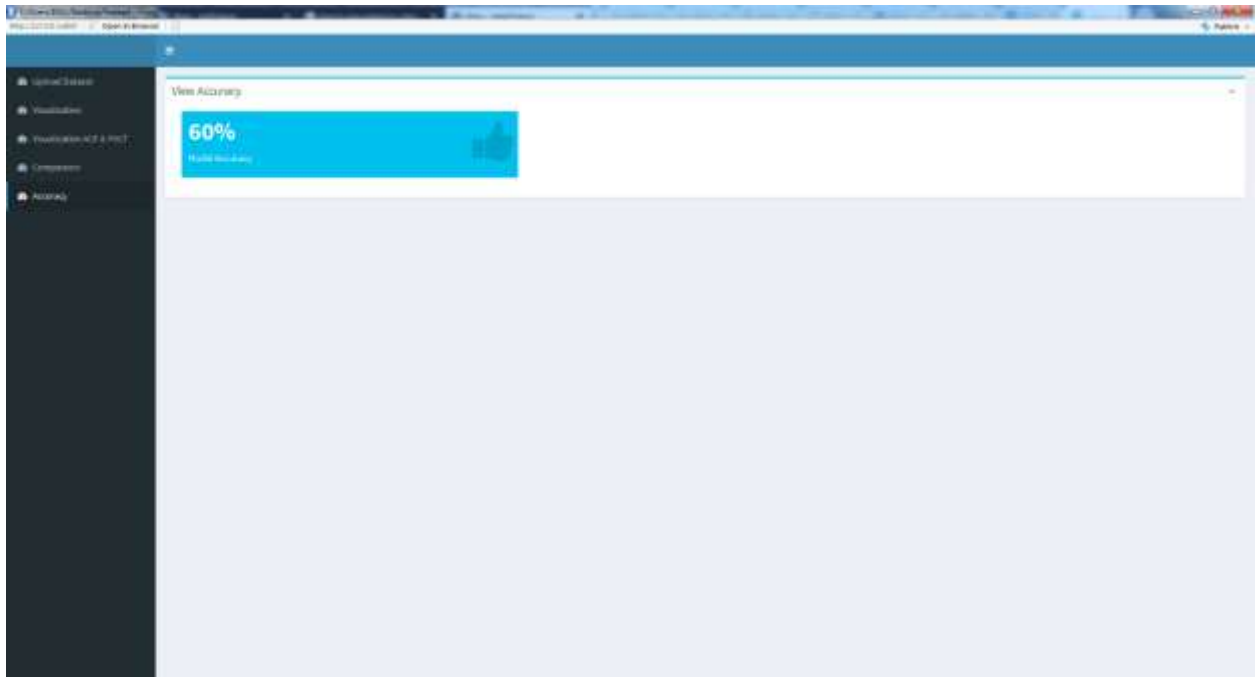
Visualization (ACF-PACF)



Comparison (Actual vs Forecast)



Accuracy



3.5 Algorithmic Trading (Quant Develop)

With automation coming to finance in a big way, the industry is in the middle of an adoption phase and it is estimated that **around 40 percent** of the trades made on the **London Stock Exchange** (LSE) now originate from algorithmic trading systems. Both Bank of America and Morgan Stanley are developing automated robotic advisers. As per Goldman Sachs, even in investment banking, where the human element is central to deal-making, technology will have a huge impact. According to some reports, some desks in Goldman Sachs had **600 traders** in **NYC 15 years ago**. Today, they have been downsized to a little **less than 2!**

Yes, big shot Wall Street jobs have been automated away. It might be a bad thing for those traders who could not cope up with this revolutionary shift in financial technology, but it has opened a plethora of lucrative opportunities for those who are well groomed in algorithmic trading. Deal-making jobs have been in decline since 2007 and it sunk by 14 percent last year at the world's largest investment banks. Up to **30 percent** of the current employees in the banking industry may lose their jobs to new technologies in the next 10 years, according to new projections from Citigroup.

By saying that an industry is contracting doesn't mean that people won't be earning a living in finance industry down the road. Finance industry will need computer engineers and data scientists, algo and quant analysts. Those days when one could just learn Excel and do some basic analysis are over. Quant analysts are going to be working with big data of transactions and volumes and market behavior, sophisticated algorithms will be trading the markets and price/quote inefficiencies will be reduced.

In order to storm this world, we will use tools from Facebook and **Google**

Facebook tool (**Prophet**) is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is robust to missing data and shifts in the trend, and typically handles outliers well.

Tools using:

Pandas and numpy: for data manipulation

Quandl: for loading data financial analysis

Pytrends: for Google trend data

Fbprophet: for Facebook tool for producing high quality forecasts for additive model time series data that has multiple seasonality with linear or non-linear growth.

PyStan is a state-of-the-art platform for statistical modeling and high-performance statistical computation. Thousands of users rely on Stan for statistical modeling, data analysis, and prediction in the social, biological, and physical sciences, engineering, and business.

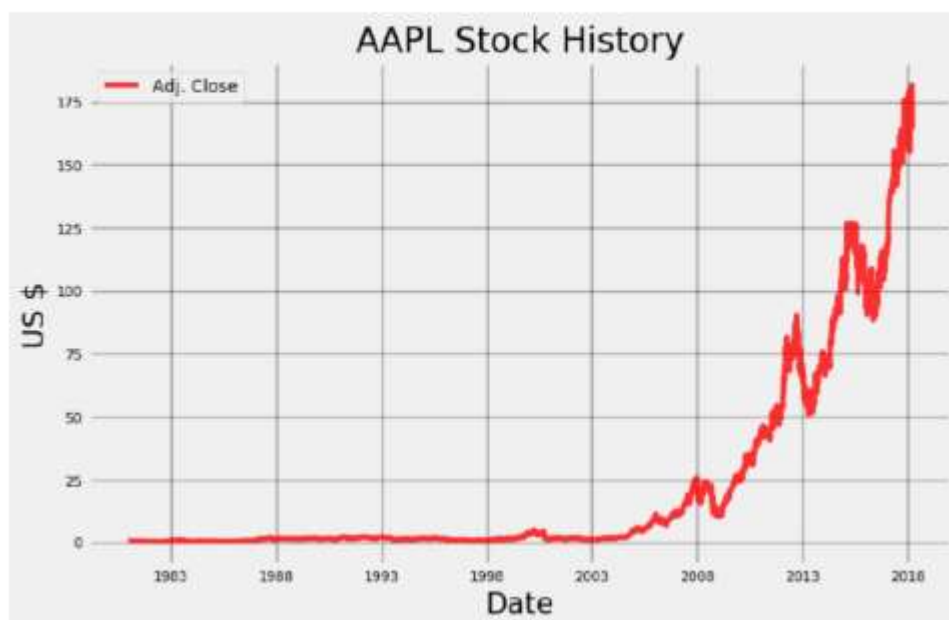
Users specify log density functions in Stan's probabilistic programming language and get:

- full **Bayesian statistical inference** with MCMC sampling (NUTS, HMC)
- approximate **Bayesian inference** with variational inference (ADVI)
- penalized **maximum likelihood estimation** with optimization (L-BFGS)

Basic Plot of Stock History

```
apple.plot_stock()
```

Maximum Adj. Close = 181.72 on 2018-03-12.
Minimum Adj. Close = 0.16 on 1982-07-08.
Current Adj. Close = 168.34 on 2018-03-27.



Examine Trends/Patterns

We can create a basic model (with no predictions) trained on the past 3 years of data to inspect any trends and patterns in the data.

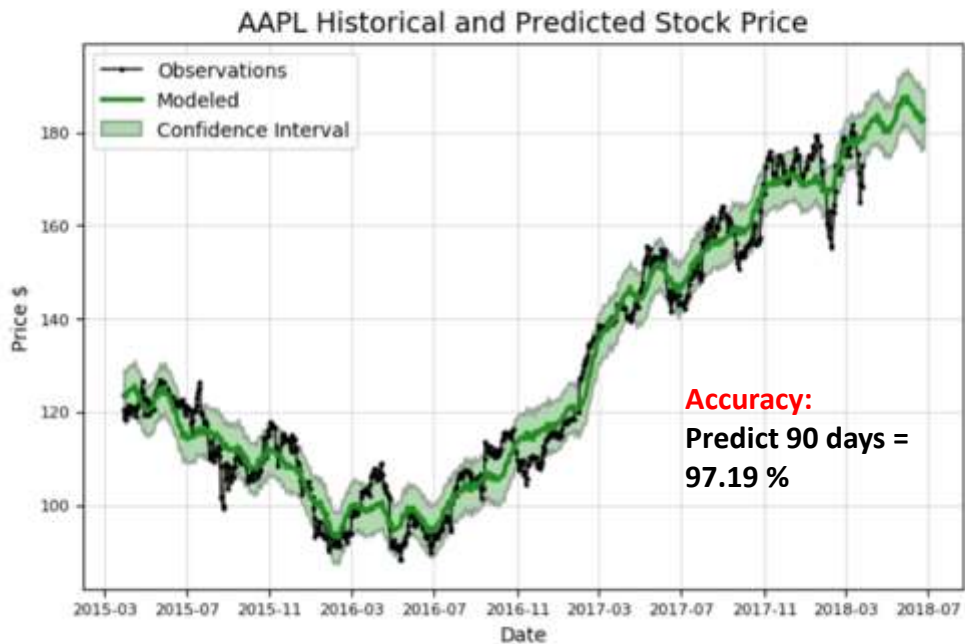
```
model, model_data = apple.create_prophet_model()
```



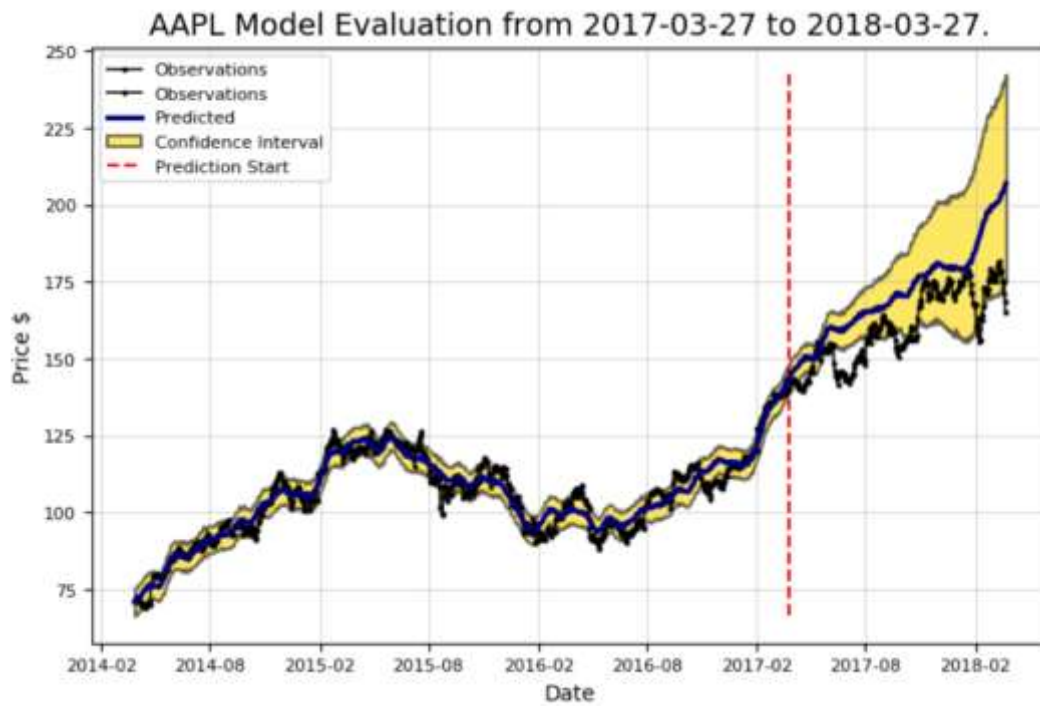
First Predictions

```
model, model_data = apple.create_prophet_model(days=90)
```

Predicted Price on 2018-06-25 = \$182.64



Evaluate Model



Reference:

- https://www.wikiwand.com/en/Stock_market_prediction
- <https://emerj.com/ai-sector-overviews/machine-learning-in-finance/>
- <https://emerj.com/ai-podcast-interviews/machine-learning-not-a-crystal-ball-but-it-brings-clarity-to-investment-decisions/>
- <https://www.investopedia.com/articles/active-trading/101014/basics-algorithmic-trading-concepts-and-examples.asp>
- <https://www.r-bloggers.com/forecasting-stock-returns-using-arma-model/>
- <http://itfeature.com/time-series-analysis-and-forecasting/component-of-time-series-data>
- <https://www.sciencedirect.com/science/article/pii/S1532046415002282>
- <https://facebook.github.io/prophet>
- https://facebook.github.io/prophet/docs/quick_start.html

Source Code:

- <https://github.com/MSaber9/Stock-Price-Prediction-Project>