Department of Informatics,

University of Leicester

CO4015 Computer Science Project

Dissertation

for

Stock Market Prediction using python

Author: Osato Osagie (1690 11647)

# Introduction

The term ‘stock market’ does not refer to a single market. It refers to several stock exchanges dispersed around the world. In these stock exchanges, traders and investors can purchase and sell shares of companies that are publicly traded. The share prices of these companies constantly fluctuate in response to the law of supply and demand.

A share is a small possession stake in a publicly traded company. The price of a stock portrays the expectation of stock investors and market analysts on the company’s future earnings.

When traders believe a company will perform well, they bid the price up by creating demand for stocks in that company. On the other hand, those traders who don’t believe in a company’s future will bid the price down by selling their holdings, thus creating an excess supply. Sellers aim to receive a high price for each share hopefully more than their initial payment, whilst buyers aim to acquire shares at their lowest price possible so that they can make a profit when selling the stock.

Investing in stocks is deemed as a reliable method to achieve profits that beat inflation over time. The returns, on average do better than those of other investments, such as bonds and commodities. According to research, as of February 2021, the total market value of all companies trading on the London Stock Exchange stood at 3.67 trillion British pounds [1].

There are two ways to make a profit on the stock market. Investors can either trade stocks or decide to hold them. Trading requires investors to buy and sell stocks frequently, taking advantage of little ticks in cost. Investors who purchase and lean toward to let their stocks appreciate in esteem over time. In some cases, investors who hold shares get rewarded with regular payments of dividends.

Investors have previously found ways to obtain insight about the businesses listed on the market for as long as markets have existed in order to increase their investment returns. However, owning to the size market and the pace at which transactions are conducted, this is not possible today. Simple statistical analysis of financial data can reveal certain trends, but in recent years, investment firms have increasingly turned to Artificial Intelligence (AI) systems to search for patterns in vast quantities of real-time equity and economic data.

AI is the ability of machines to behave like humans and learn autonomously. For instance, a machine might display learning and problem-solving abilities without the use of hard-coded software containing detailed instructions [1].

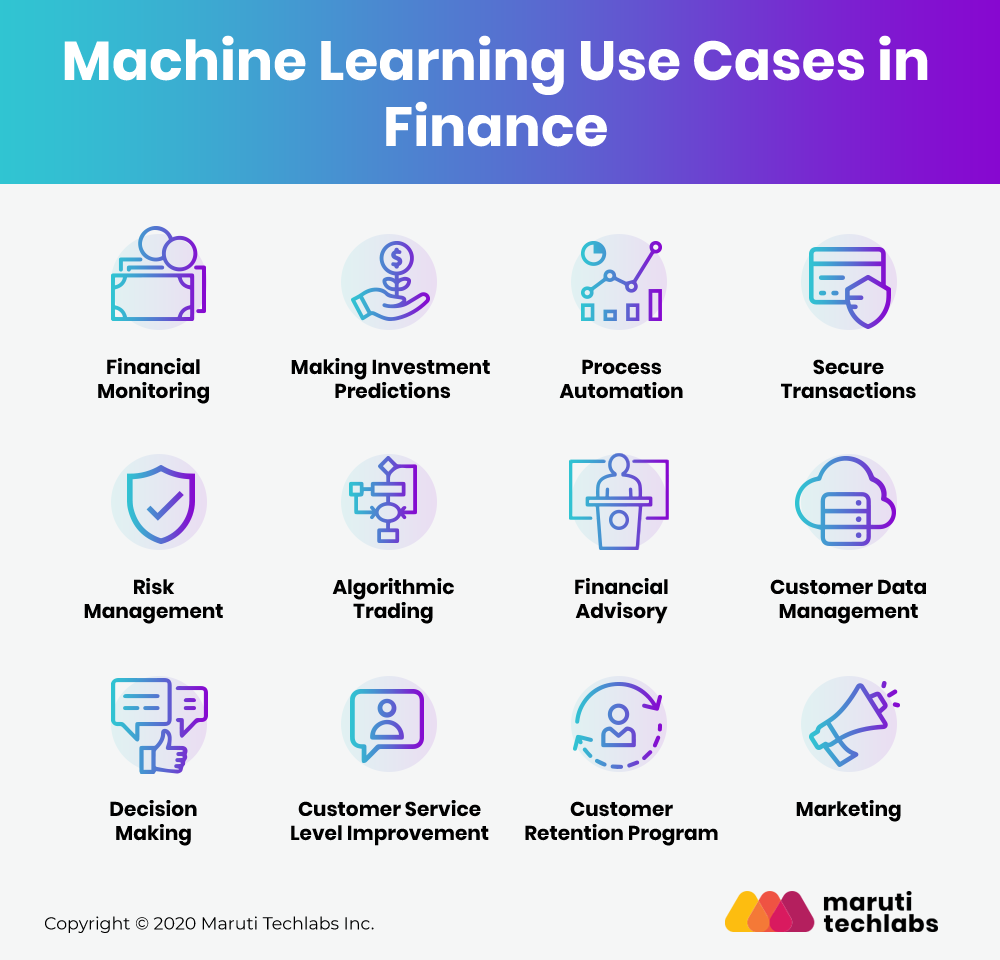


Figure 1- An image showing the main use cases of Machine Learning in Finance [2]

Machine Learning (ML) is a subfield of AI, that enables machines to learn from historic data or experiences without being explicitly programmed. Figure 1 shows the different use cases for Machine Learning in finance. The project I will be building focusing on using ML to make investment predictions. Using ML to make investment predictions is advantageous as it can lead to better predictions of stock prices, fewer errors, and greater efficiency for the investor. To do this, ML algorithms extract key insights from the dataset, learns from it then apply several techniques to accurately predict the result.

However, it is important to take into consideration the other factors that might affect the price of a company’s stock. The stock market is very volatile, thus meaning no system can accurately predict it.

Deep Learning (DL) is a subfield of ML. It teaches computes to learn by example in the same way that humans do. A prominent example of DL is self-driving vehicles, allowing them to identify a stop sign or differentiate between a pedestrian and a lamppost. DL models are able to achieve cutting-edge precision, often even outperforming humans. A wide collection of labelled data and neural network architectures with several layers are used to train models.

## 1.1 Aims

Predicting markets has become an increasing priority for investors. The primary goal of an investor is to buy a stock when its value is low and sell when the value of the stock is high. However, this can be daunting for financial investors as they are unaware of the stocks that will return maximum profits. Using Machine learning to predict the long-term value of a stock makes this process somewhat easier. This project aims to predict the stock market price of a company using supervised machine learning algorithms.

## 1.2 Objectives

To meet the aims this project sets out to achieve, I have broken it down into several tasks. These objectives are as follows:

* Obtain real-time and historical equity data from Alpha Vantage API.
* Clean data and form data sets with the obtained data.
* Build python functions to calculate technical indicators from the obtained dataset.
* Train a Decision Tree and SVM model to predict the S&P 500.
* Test various models to find which one works best for predicting Standard and Poor’s 500 (S&P 500).
* Fine tune model parameters to have as low bias as possible while also having low variance on the training data.
* Split datasets into training and test data, and train models with data.
* Tests the models with data and measure accuracy using metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared (R2) .
* Evaluate which features have the most impact on stock prices.
* Visualise results using a line chart showing the predicted prices versus the actual prices of the stock.
* Implement a user-friendly interface for the prediction tool.

## 1.3. Changed objectives

Following the feedback from my interim report to ‘add more up to date technologies, some objectives have been changed. These include:

* Train a Ridge Regression, Random Forest and Long Short-Term Memory models to predict the S&P 500.

## 1.4. Resources & Tools

This project is built using Python programming language. I chose to use Python because it is easy and flexible to use. Python is also versatile and has a robust collection of libraries that make machine learning tools easily available to use. Also, Python has a diverse pack of visualisation options available which makes it ideal for creating graphs and charts.

The Python code will be written using Jupiter notebook. It is a web application that allows users to create documents containing live code and visualizations. In addition, I have used various Python libraries to access tools that have enabled me to build my project. These include:

* SKlean
* Matplotlib
* Pandas
* Numpy

# Literature Review

This chapter will discuss current literature that will be used to set the stage for this project. Such literature are based on subjects surrounding the current technology development of AI in trading and investing,

## 2.1. Impact of AI on Trading and Investing

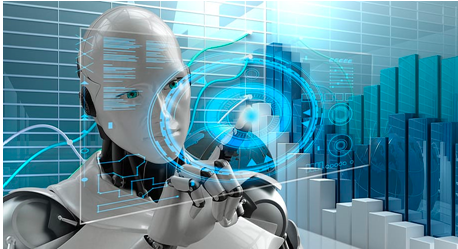


Figure 2-Photo credit: [3]

AI, ML, and DL have been transforming finance and investing. Although humans remain an important part of the trading equation, AI is becoming increasingly important. Electronic trades account for approximately 45 percent of cash equities trading revenues, according to a new report by Coalition [4], a U.K research company. Though hedge funds are wary of automation, many of them use AI-powered analysis to generate investment ideas and build portfolios.

“Machine Learning is progressing much faster, and financial institutions are among the early adopters.” Intelenet Global Service's vice president of global business growth, Anthony Antenucci, recently said [5]. Globally companies are developing new products and technology that use AI to make trading and investing more data-driven and effective. Bloomberg announced in September 2017 that Japan’s third largest lender will use AI in the equities sector through algorithm-based services for institutional clients [6]. In the United States, Merrill Lynch is currently testing a stock platform to find value in small-cap stocks that analysts would otherwise ignore [7].

Currently, there are companies already using AI for smarter trading. Aquan, a UK data science company [8] uses its platform to democratize trading by encouraging data scientists of all backgrounds to build algorithmic trading strategies that assist in the resolution of investment problems. As a result, investment clients will benefit from data science without having to invest in costly in-house expertise. Aquan has had significant industry impact, they graduated from Techstars in 2018 and was recently named the 2019 Europa Awards’ Hottest Fintech n Europe.

Similarly, EquBot’s[9] proprietary investment technology, which is affiliated with IBM, blends AI with an active exchange-traded fund (ETF). The business centralizes the investment process by gathering and processing data from different sources (news articles, social media postings, financial statements) from around the world to “build a cause-and-effect understanding of economies, businesses, and management”. The impact EquBot has had in the industry has also been significant, recently they launched the AI-powered Foreign Equity ETF, which aims to invest in established international markets outside of the United States.

## 2.2 Alpha Vantage Stock Application Programming Interface (API)

In this project, I will be using an API called Alpha Vantage. An API is where a website provides a set of structured Hypertext Transfer Protocol (HTTP) requests that return JavaScript Object Notation (JSON) or Extensible Markup language (XML) files.

In this project, the Alpha Vantage API provides access to historical and real-time data for a variety of markets. The API allows me to access the data directly in python, from there I can manipulate the data or store it for later use. Alpha Vantage provides its service at no fee. They permit 5 requests per minute and 500 requests per day.

If a higher rate cap is needed, there are many premium plans available. Premium plans vary in price from $29.99 per month for 30 requests per minute to $249.99 per month for 1200 requests per minute. For stocks, Alpha Vantage provides historical and real-time info. There are many time frames to choose from, ranging from 1-minute bars to weekly. The most important benefit is that it is absolutely free. Furthermore, the information is comprehensive. For stocks, we find price data going back 20 years.

## 2.3 Stock Market Prediction

There are three separate trading schools of thinking, all driven by the need to forecast market fluctuations and profit: fundamental, analytical and quantitative technical research.

### 2.3.1 Fundamental Analysis

The study of economic factors that affect the price of a stock is known as fundamental analysis. A balance sheet and an income statement are examples of such causes. Throughout the year, these reports are published quarterly. Fundamental analysis is commonly used to predict long-term market fluctuations because it depends on forecasts that are published on the basis of a longer time period.

### 2.3.2 Technical Analysis

Chart, histogram

Description automatically generated

Figure 3- Relative Strength Index

Technical research attempts to predict what other stakeholders are thinking based on available knowledge about stock prices and volume. To forecast future prices, technical analysts use a range of metrics derived from the past experience of stock price and volume. Overall, the main concept technical analysis is trend. According to technical analysts, stock price trends are triggered by a discrepancy between supply and demand for stocks, which is reflected in the bid and ask prices. Scientific analysts strive to derive trends from the chaotic data of stock prices. Since it depends on visual analysis, technical analysis is mostly qualitative. Figure 3 shows an example of such stock charts. It represents one year’s worth of historical stock price data for the IT sector in 2016.

### 2.3.3 Quantitative Analysis

Quantitative Analysis is a method of determining the value of a financial asset, such as a stock. It emphasises mathematical and statistical analysis. Investors use the data produced by these computer models to evaluate investment opportunities and create what they believe will be a profitable trading strategy.

In this proposed project, I will be implementing three prediction models: Linear Regression, Random Forest, and Long Short-Term Memory. Each will be evaluated using regression metrics.

There have been several other approaches to this problem. Using data from several global stock markets, Jasic and Wood (2004) created an artificial neural network to predict regular stock market index returns. The main goal was to encourage efficient trading. The neural network’s prediction efficiency is compared to a benchmark linear autoregressive model, and the prediction improvement was verified when applied to the S&P 500 and DAX indices.

Deep learning networks for stocks market research and prediction was investigated by Chong, Han, and Park (2017). Deep neural networks can extract additional information from the residuals of the autoregressive model and boost predictive efficiency, according to empirical findings.

# Design and Implementation

In this chapter I will be discussing the design and implementation which will be used to achieve accurate models to predict stock prices.

## 2.4 Technical Indicators

To get better results I have decided to use Technical Indicators. Technical indicators are mathematical calculations based on the price, volume, or open interest of a security [5]. Whilst these indicators are designed to analyze short-term price movements, they are also useful to long-term investors who want to identify entry and exit points.

In particular, I will be using momentum technical indicators. These indicators measure the rate at which a security’s price moves, and there are a number of metrics that can be used to do so including:

* The Relative Strength Index (RSI) is a common momentum indicator for evaluating whether a stock is overbought or oversold. When demand unjustifiably drives the price of a stock upwards, it is said to be overbought. This is commonly considered an indicator that the stock is overvalued and is likely to decline in price. When a stock’s price falls dramatically below its true worth, it is considered to be oversold. As a consequence of panic sale, this is the outcome. The RSI scale runs from 0 to 100, and a reading of more than 70 suggests that the stock is overbought, whereas a reading of less than 30 implies that the stock is oversold.

**Formula:**



Figure 4- RSI Formula

**Code:**

The code for RSI calculation is shown in appendix A. I started off by identifying the up days (days where the stock price went up) and the down days (days where the stock price went down). I created a column that identifies the change in price, then used a condition that set the value of up and down days based on the change in price. In addition, I ensured the values for down days were absolute by modifying the column and calculating the Exponential Moving Average of both the UP and Down columns. The final step was to calculate the Relative Strength metric and pass that through to the RSI calculation.

* Stochastic Oscillator tracks the price’s momentum. In certain cases, momentum shifts before the price shifts. It calculates the closing price in relation to the low-high range for a given time span.

**Formula:**

Graphical user interface, text, application

Description automatically generated

Figure 5- Stochastic Oscillator Formula

**Code:**

The code I used to calculate the Stochastic Oscillator is shown in appendix B. To calculate Stochastic Oscillator, I use the rolling lambda function. With this function, I specify a 14-period window, and what measurement to apply to each window. After the maximum and minimum values are obtained, I then pass it through my formula and apply the results to the main data frame.

* Williams %R varies from -100 to 0. It shows a sell signal when its value is greater than -20, and a buy signal when the value is less than -80.

**Formula:**

Text

Description automatically generated

Figure 6- Williams %R Formula []

**Code:**

The code I used to calculate the Stochastic Oscillator is shown in appendix C.

* Price Rate of Change (ROC) is a momentum-based technical indicator which compares the present price to the price from a certain number of periods ago. When price changes are to the upside, the ROC indicator moves upwards into positive territory, and when price changes are to the downside, the indicator moves downwards into negative territory.

**Formula:**

Graphical user interface, text, application

Description automatically generated

Figure 7-ROC Formula []

**Code:**

The code I used to calculate the Stochastic Oscillator is shown in appendix E.

* Moving Average Convergence Divergence (MACD) depicts the connection between two security price moving averages. By subtracting the 26-period exponential moving average (EMA) from the 12-period EMA the MACD is estimated. The "signal line," a nine-day EMA of the MACD, is then plotted on top of the MACD line. A sell signal is indicated when the MACD falls below the SingalLine. It indicates a buy signal as it rises above the SignalLine.

**Formula:**



Figure 8-MACD Formula []

**Code:**

The code I used to calculate the MACD is shown in appendix D. To calculate the MACD, I used the column in the data frame containing the closing price of the stock. After obtaining this column, I then applied the transform method along with the specified Lambda function. Then I proceeded to calculate the EMA by calling the exponential moving weight (emw) function and specifying how many periods to look back. I used the default definition provided by the formula and specified 26 & 12.

After calculating the 26-period EMA and the 12-period EMA, I took the difference between both to obtain the MACD. In addition to the MACD, I also wanted to calculate the EMA of the MACD, so I applied the same ewm function, but in this caace I specified a span of 9. Finally, both the MACD and MACD\_EMA were added to the main data frame.

# Models

## 4.1 Supervised Learning

There are two main types of learning in ML: supervised and unsupervised. In unsupervised learning the model isn’t supervised, instead it is allowed to explore knowledge on its own. We accomplish this by giving the model an “unlabelled” dataset that does not specify which category or value is the “right” response. When we use supervised learning, we give the model a “labelled” dataset that tells it what the “right” value is.

## 4.1.1 Regression

Regression is a supervised learning technique that aids in the discovery of variable correlations and allows one to forecast a continuous output variable using one or more feature variables. Prediction, forecasting, time series modelling, and evaluating the cause-effect relationship between variables are all common applications. In my project, a single output value is produced for the price of a stock using the training data provided. This a probabilistic interpretation that is determined by taking into account the strength of association between the input variables.

## 4.2 Random Forest

Random Forest is a ML ensemble approach that is common due to its versatility, simplicity, and often high-quality performance. In this project, I use Random Forest algorithm to build a regression model that will help me predict the closing price of a stock based on a range of technical indicators.

Random Forest is made up of Decision Trees, which are the basic building blocks. In essence, Decision Trees are a flowchart framework in which each node tests a different attribute of an entity. Consider the following scenario: I have an individual who will represent our object. We then put this person’s qualities to the test. One test would be to determine whether they are male or female. In our tree, the test will be a “Decision Node” and each of the potential outcomes, “Male” or Female” will be a leaf node. Out “Root Node” will be the first “Decision Node” in our Decision Tree.

**Key Words**

* **Root Node**- The entire population or sample is represented by this node, which is then divided into two or more homogeneous sets. This is where we begin out Decision Tree.
* **Splitting**- The process of breaking down a node into two or more sub-nodes, such as gender.
* **Decision Node**- A decision Node is Formed when a sub-node splits into more sub-nodes.
* **Leaf/Terminal Node**- This refers to nodes that do not split.
* **Pruning-** this is a method of removing sub-nodes from a decision node. The splitting process can be defined as the polar opposite of splitting.
* **Branch/Sub-Tree**- This is a part of a tree that is smaller than the entire tree.
* **Parent and Child Node**- A parent node of sub-nodes is a node that is divided into sub-nodes, while sub-nodes are the children of the parent node.

Ensemble Learning models are models that uses decisions from multiple models combined to increase the model’s overall efficiency. Ensemble Learning Is perfectly summed up by the old adage that two heads are better than one. To get a better understanding of what the true response is, we combine the effects of the several models. Bagging is used in our Random Forest algorithm to boost efficiency.

What is bagging?

# References

[[1] https://www.statista.com/statistics/324578/market-value-of-companies-on-the-london-stock-exchange/](%5b1%5d%20https://www.statista.com/statistics/324578/market-value-of-companies-on-the-london-stock-exchange/)

[2] Marutitech.com (-) 12 Use Cases of AI and Machine Learning In Finance [ONLINE]. Available at: <https://marutitech.com/ai-and-ml-in-finance/>

[3] <https://www.pxfuel.com/en/free-photo-qualj>

[4] <https://www.crisil.com/en/home/our-businesses/crisil-coalition.html>

[5] <https://www.itprotoday.com/machine-learning/how-ai-trading-systems-will-shake-wall-street>

[6] <https://www.bloomberg.com/news/articles/2017-09-20/mizuho-is-said-to-offer-ai-trading-service-before-mifid-overhaul>

[7] <https://www.cybersecobservatory.com/2017/06/06/artificial-intelligence-transforming-investment-strategies/>

[8] <https://www.auquan.com/>

[9] <https://equbot.com/>

# Appendix A

## Indicator Calculation: Relative Strength Index (RSI)

### Code Explanation:

1. Copy the desired columns and store them in new variables.
2. To calculate the indicator, group the columns by symbol, pick the column we want to transform, and use the transform method along with a lambda function.
3. Save the values in the main data frame

def calc\_RSI(data\_dated):

# Calculate the 14 day RSI

n = 14

# First make a copy of the data frame twice

up\_df, down\_df = data\_dated[['Ticker','change\_in\_price']].copy(), data\_dated[['Ticker','change\_in\_price']].copy()

# For up days, if the change is less than 0 set to 0.

up\_df.loc['change\_in\_price'] = up\_df.loc[(up\_df['change\_in\_price'] < 0), 'change\_in\_price'] = 0

# For down days, if the change is greater than 0 set to 0.

down\_df.loc['change\_in\_price'] = down\_df.loc[(down\_df['change\_in\_price'] > 0), 'change\_in\_price'] = 0

# We need change in price to be absolute.

down\_df['change\_in\_price'] = down\_df['change\_in\_price'].abs()

# Calculate the EWMA (Exponential Weighted Moving Average), meaning older values are given less weight compared to newer values.

ewma\_up = up\_df.groupby('Ticker')['change\_in\_price'].transform(lambda x: x.ewm(span = n).mean())

ewma\_down = down\_df.groupby('Ticker')['change\_in\_price'].transform(lambda x: x.ewm(span = n).mean())

# Calculate the Relative Strength

relative\_strength = ewma\_up / ewma\_down

# Calculate the Relative Strength Index

relative\_strength\_index = 100.0 - (100.0 / (1.0 + relative\_strength))

# Add the info to the data frame.

data\_dated['down\_days'] = down\_df['change\_in\_price']

data\_dated['up\_days'] = up\_df['change\_in\_price']

data\_dated['RSI'] = relative\_strength\_index

# Appendix B

## Indicator Calculation: Stochastic Oscillator

### Code Explanation:

1. Copy the desired columns and store them in new variables.
2. To calculate the indicator, group the columns by symbol, pick the column we want to transform, and use the transform method along with a lambda function.
3. Save the values in the main data frame

def stochastic\_Oscillator(data\_dated):

# Calculate the Stochastic Oscillator

n = 14

# Make a copy of the high and low column.

low\_14, high\_14 = data\_dated[['Ticker','3. low']].copy(), data\_dated[['Ticker','2. high']].copy()

# Group by symbol, then apply the rolling function and grab the Min and Max.

low\_14 = low\_14.groupby('Ticker')['3. low'].transform(lambda x: x.rolling(window = n).min())

high\_14 = high\_14.groupby('Ticker')['2. high'].transform(lambda x: x.rolling(window = n).max())

# Calculate the Stochastic Oscillator.

k\_percent = 100 \* ((data\_dated['4. close'] - low\_14) / (high\_14 - low\_14))

# Add the info to the data frame.

data\_dated['low\_14'] = low\_14

data\_dated['high\_14'] = high\_14

data\_dated['k\_percent'] = k\_percent

# Appendix C

## Indicator Calculation: Williams %R

### Code Explanation:

1. Copy the desired columns and store them in new variables.
2. To calculate the indicator, group the columns by symbol, pick the column we want to transform, and use the transform method along with a lambda function.
3. Save the values in the main data frame

def calc\_williams\_r(data\_dated):

# Calculate the Williams %R

n = 14

# Make a copy of the high and low column.

low\_14, high\_14 = data\_dated[['Ticker','3. low']].copy(), data\_dated[['Ticker','2. high']].copy()

# Group by symbol, then apply the rolling function and grab the Min and Max.

low\_14 = low\_14.groupby('Ticker')['3. low'].transform(lambda x: x.rolling(window = n).min())

high\_14 = high\_14.groupby('Ticker')['2. high'].transform(lambda x: x.rolling(window = n).max())

# Calculate William %R indicator.

r\_percent = ((high\_14 - data\_dated['4. close']) / (high\_14 - low\_14)) \* - 100

# Add the info to the data frame.

data\_dated['r\_percent'] = r\_percent

# Appendix D

## Indicator Calculation: Moving Average Convergence Divergence (MACD)

### Code Explanation:

1. Copy the desired columns and store them in new variables.
2. To calculate the indicator, group the columns by symbol, pick the column we want to transform, and use the transform method along with a lambda function.
3. Save the values in the main data frame

def calc\_macd(data\_dated):

# Calculate the MACD

ema\_26 = data\_dated.groupby('Ticker')['4. close'].transform(lambda x: x.ewm(span = 26).mean())

ema\_12 = data\_dated.groupby('Ticker')['4. close'].transform(lambda x: x.ewm(span = 12).mean())

macd = ema\_12 - ema\_26

# Calculate the EMA

ema\_9\_macd = macd.ewm(span = 9).mean()

# Store the data in the data frame.

data\_dated['MACD'] = macd

data\_dated['MACD\_EMA'] = ema\_9\_macd

# Appendix E

## Indicator Calculation: Price Rate Of Change

### Code Explanation:

1. Copy the desired columns and store them in new variables.
2. To calculate the indicator, group the columns by symbol, pick the column we want to transform, and use the transform method along with a lambda function.
3. Save the values in the main data frame

ef calc\_price\_rate\_of\_change(data\_dated):

# Calculate the Price Rate of Change

n = 9

# Calculate the Rate of Change in the Price, and store it in the Data Frame.

data\_dated['Price\_Rate\_Of\_Change'] = data\_dated.groupby('Ticker')['4. close'].transform(lambda x: x.pct\_change(periods = n))

calc\_price\_rate\_of\_change(data\_dated)