Build A Simple Stock Movement

Classifier Using Machine Learning & Python

Rand

randerson112358

.

8 min read

·

Feb 21, 2021

Can stock indicators combined with machine learning predict the price movement of stocks ?

> **Disclaimer:** The material in this article is purely educational and should not be taken as professional investment advice. Invest at your own discretion.

Before we begin, if you enjoy my articles and content and would like more content on programming, stocks, machine learning, etc. , then please give this article a few claps, it definitely helps out and I truly appreciate it ! So let's begin !

In this article I will attempt to create a model that can determine if the price of an asset will go up or down the next day based on stock data using machine learning, technical indicators and python ! It is extremely hard to try and predict the stock market momentum direction, but let's give it a try.

What Are Stock Market Technical Indicators ?

Stock market technical indicators are signals used to interpret stock or financial data trends to attempt to predict future price movements within the market. Stock indicators help investors to make trading decisions.

Types of Technical Indicators

Simple Moving Average (SMA): A **simple moving average** is a technical trend **indicator** that can aid in determining if an asset price will continue or if it will reverse a bull or bear trend. A **simple moving average** can be enhanced as an exponential **moving average** (EMA) that is more heavily weighted on recent price action. -investopedia

Exponential Moving Average (EMA): The **EMA** is a **moving average** that places a greater weight and significance on the most recent data points. Like all **moving averages**, this technical trend **indicator** is used to produce buy and sell signals based on crossovers and divergences from the historical **average**. -investopedia

Moving Average Convergence Divergence (MACD) : Moving Average Convergence Divergence (MACD) is a trend-following momentum indicator that shows the relationship between two moving averages of a security's price. The MACD is calculated by subtracting the 26-period Exponential Moving Average (EMA) from the 12-period EMA. -investopedia

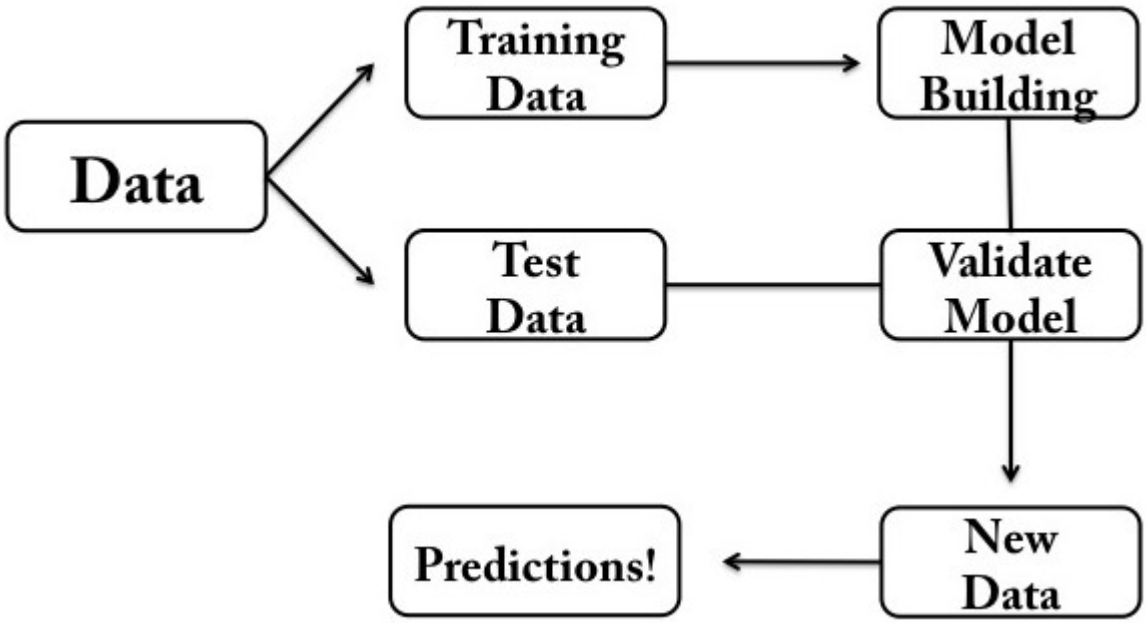
Relative Strength Index (**RSI**)**:** The **relative strength index** (**RSI**) is a momentum **indicator** used in technical analysis that measures the magnitude of recent price changes to evaluate overbought or oversold conditions in the price of a stock or other asset. -investopedia

These are the indicators that we will be programming in this article using python.

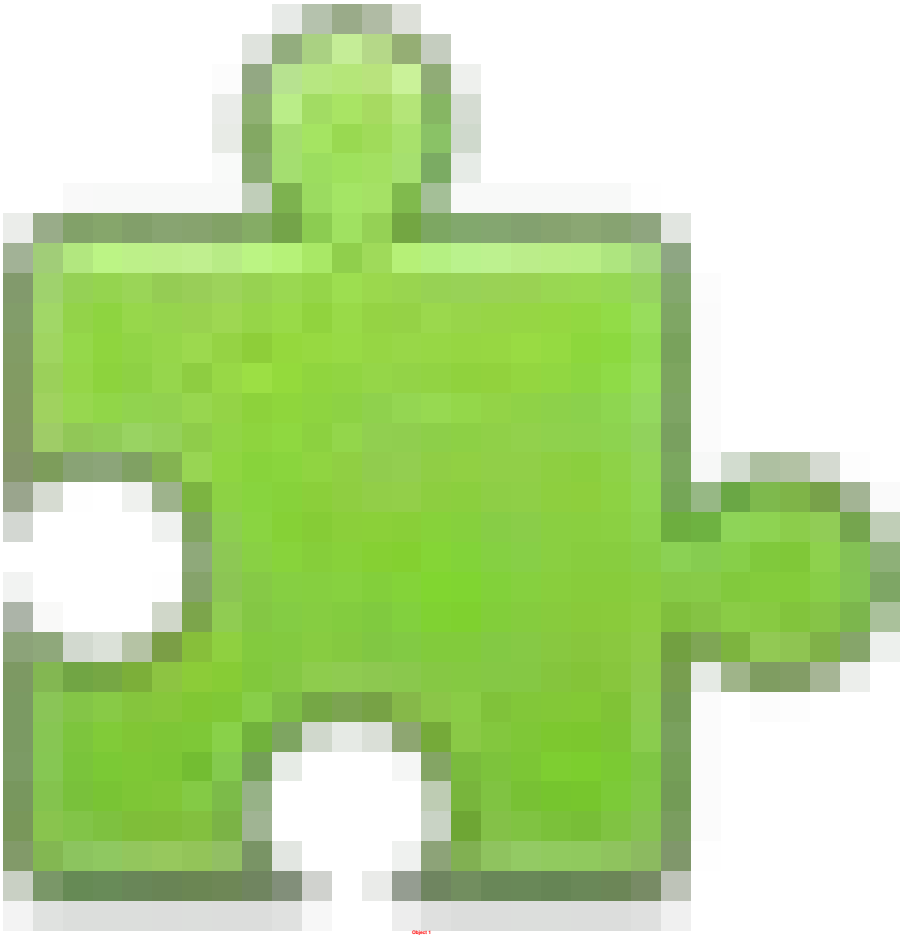
What is Machine Learning ?

Machine learning is a subset of artificial intelligence, it is the science of getting computers to act without being explicitly programmed, and is mostly just statistics. Machine learning is used to find patterns in data that you can then make predictions on. It can be subdivided into **supervised learning** and **unsupervised learning** or some mixture of both.

> *Machine learning is a computer program said to learn from experience 'E' with respect to some class of tasks 'T' and performance measure 'P', if its performance at tasks in 'T', as measured by 'P', improves with experience 'E'.*
> *— Tom Mitchell*



Before writing any code, if you prefer not to read this article and would like a video representation of it, you can check out the **YouTube Video** . It goes through everything in this article with a little more detail, and will help make it easy for you to start programming even if you don't have the programming language Python installed on your computer. Or you can use both as supplementary materials for learning !



Programming

First, I want to create a description about the program so that I can simply read the description and know what the program is supposed to do or what the program is about.

```
#Description: Use stock indicators with machine learning to try to predict the
direction of a stock price: #1 means the stock price goes up
#0 means the stock price goes down or stays the same
```

Import the libraries that we will need throughout the program.

```
#Import the libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

Load the data and store it into a variable. Note that I am using Google Collab to write this program, so I must use Googles library to upload the data set.

```
#Load the data set
from google.colab import files
files.upload()#Store the data into the data frame
df = pd.read_csv('GOOG_Stock.csv')#show the data frame
df
```

Date	High	Low	Open	Close	Volume	Adj Close
2019-06-03	1065.500000	1025.000000	1065.500000	1036.229980	5130600	1036.229980
2019-06-04	1056.050049	1033.689941	1042.900024	1053.050049	2833500	1053.050049
2019-06-05	1053.550049	1030.489990	1051.540039	1042.219971	2168400	1042.219971
2019-06-06	1047.489990	1033.699951	1044.989990	1044.339966	1703200	1044.339966
2019-06-07	1070.920044	1048.400024	1050.630005	1066.040039	1802400	1066.040039
...	...	...	...	...	...	...
2019-12-11	1351.199951	1342.670044	1350.839966	1345.020020	850400	1345.020020
2019-12-12	1355.775024	1340.500000	1345.939941	1350.270020	1281000	1350.270020
2019-12-13	1353.093018	1343.869995	1347.949951	1347.829956	1549600	1347.829956
2019-12-16	1364.680054	1352.670044	1356.500000	1361.170044	1397300	1361.170044
2019-12-17	1365.000000	1351.322998	1362.890015	1355.119995	1854000	1355.119995

139 rows × 6 columns

Create and Calculate the Indicators

Create functions to calculate the Simple Moving Average (SMA) and the Exponential Moving Average (EMA).

```
#Create functions to calculate the SMA, & the EMA
#Create the Simple Moving Average Indicator
#Typical time periods for moving averages are 15, 20,& 30
#Create the Simple Moving Average Indicator
def SMA(data, period=30, column='Close'):
  return data[column].rolling(window=period).mean()
#Create the Exponential Moving Average Indicator
def EMA(data, period=20, column='Close'):
  return data[column].ewm(span=period, adjust=False).mean()
```

Next, create a function to calculate the Moving Average Convergence Divergence (MACD).

```
#Create a function to calculate the Moving Average Convergence/Divergence (MACD)
def MACD(data, period_long=26, period_short=12, period_signal=9,
column='Close'):
    #Calculate the Short Term Exponential Moving Average
    ShortEMA = EMA(data, period_short, column=column) #AKA Fast moving average
    #Calculate the Long Term Exponential Moving Average
    LongEMA = EMA(data, period_long, column=column) #AKA Slow moving average
    #Calculate the Moving Average Convergence/Divergence (MACD)
    data['MACD'] = ShortEMA - LongEMA
    #Calcualte the signal line
    data['Signal_Line'] = EMA(data, period_signal,
column='MACD')#data['MACD'].ewm(span=period_signal, adjust=False).mean()

    return data
```

Last, but not least create a function to calculate the Relative Strength Index (RSI).

```
#Create a function to calculate the Relative Strength Index (RSI)
def RSI(data, period = 14, column = 'Close'):
  delta = data[column].diff(1) #Use diff() function to find the discrete
difference over the column axis with period value equal to 1
  delta = delta.dropna() # or delta[1:]
  up =  delta.copy() #Make a copy of this object’s indices and data
  down = delta.copy() #Make a copy of this object’s indices and data
  up[up < 0] = 0
  down[down > 0] = 0
  data['up'] = up
  data['down'] = down
  AVG_Gain = SMA(data, period, column='up')#up.rolling(window=period).mean()
  AVG_Loss = abs(SMA(data, period,
column='down'))#abs(down.rolling(window=period).mean())
  RS = AVG_Gain / AVG_Loss
  RSI = 100.0 - (100.0/ (1.0 + RS))

  data['RSI'] = RSI
  return data
```

Prepare the Data Set for Machine Learning

Add the indicators to the data set and show the data.

```
#Add the indicators to the data set
#Creating the data set
```

```
MACD(df)
RSI(df)
df['SMA'] = SMA(df)
df['EMA'] = EMA(df)
#Show the data
df
```

Date	High	Low	Open	Close	Volume	Adj Close	MACD	Signal_Line	up	down	RSI	SMA	EMA
2019-06-03	1065.500000	1025.000000	1065.500000	1036.229980	5130600	1036.229980	0.000000	0.000000	NaN	NaN	NaN	NaN	1036.229980
2019-06-04	1056.050049	1033.689941	1042.900024	1053.050049	2833500	1053.050049	1.341772	0.268354	16.820068	0.000000	NaN	NaN	1037.831892
2019-06-05	1053.550049	1030.489990	1051.540039	1042.219971	2168400	1042.219971	1.513789	0.517441	0.000000	-10.830078	NaN	NaN	1038.249804
2019-06-06	1047.489990	1033.699951	1044.989990	1044.339966	1703200	1044.339966	1.800425	0.774038	2.119995	0.000000	NaN	NaN	1038.829819
2019-06-07	1070.920044	1048.400024	1050.630005	1066.040039	1802400	1066.040039	3.735540	1.366339	21.700073	0.000000	NaN	NaN	1041.421269
...	...	...	...	...	...	...	...	...	...	...	...	...	...
2019-12-11	1351.199951	1342.670044	1350.839966	1345.020020	850400	1345.020020	16.384554	14.407740	0.359985	0.000000	70.052514	1307.157003	1316.052424
2019-12-12	1355.775024	1340.500000	1345.939941	1350.270020	1281000	1350.270020	17.132277	14.952647	5.250000	0.000000	72.606290	1310.123002	1319.311243
2019-12-13	1353.093018	1343.869995	1347.949951	1347.829956	1549600	1347.829956	17.328210	15.427760	0.000000	-2.440063	75.083593	1313.047001	1322.027311
2019-12-16	1364.680054	1352.670044	1356.500000	1361.170044	1397300	1361.170044	18.348414	16.011891	13.340088	0.000000	75.548668	1315.961336	1325.755190
2019-12-17	1365.000000	1351.322998	1362.890015	1355.119995	1854000	1355.119995	18.455994	16.500711	0.000000	-6.050049	69.643642	1318.086336	1328.551838

139 rows × 13 columns

Create the target column.

```
#Create the target column
df['Target'] = np.where(df['Close'].shift(-1) > df['Close'], 1, 0) # if
tomorrows price is greater than todays price put 1 else put 0
#Remove the date column
#remove_list = ['Date']
#df = df.drop(columns=remove_list)
#Show the data
df
```

Date	High	Low	Open	Close	Volume	Adj Close	MACD	Signal_Line	up	down	RSI	SMA	EMA	Target
2019-06-03	1065.500000	1025.000000	1065.500000	1036.229980	5130600	1036.229980	0.000000	0.000000	NaN	NaN	NaN	NaN	1036.229980	1
2019-06-04	1056.050049	1033.689941	1042.900024	1053.050049	2833500	1053.050049	1.341772	0.268354	16.820068	0.000000	NaN	NaN	1037.831892	0
2019-06-05	1053.550049	1030.489990	1051.540039	1042.219971	2168400	1042.219971	1.513789	0.517441	0.000000	-10.830078	NaN	NaN	1038.249804	1
2019-06-06	1047.489990	1033.699951	1044.989990	1044.339966	1703200	1044.339966	1.800425	0.774038	2.119995	0.000000	NaN	NaN	1038.829819	1
2019-06-07	1070.920044	1048.400024	1050.630005	1066.040039	1802400	1066.040039	3.735540	1.366339	21.700073	0.000000	NaN	NaN	1041.421269	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2019-12-11	1351.199951	1342.670044	1350.839966	1345.020020	850400	1345.020020	16.384554	14.407740	0.359985	0.000000	70.052514	1307.157003	1316.052424	1
2019-12-12	1355.775024	1340.500000	1345.939941	1350.270020	1281000	1350.270020	17.132277	14.952647	5.250000	0.000000	72.606290	1310.123002	1319.311243	0
2019-12-13	1353.093018	1343.869995	1347.949951	1347.829956	1549600	1347.829956	17.328210	15.427760	0.000000	-2.440063	75.083593	1313.047001	1322.027311	1
2019-12-16	1364.680054	1352.670044	1356.500000	1361.170044	1397300	1361.170044	18.348414	16.011891	13.340088	0.000000	75.548668	1315.961336	1325.755190	0
2019-12-17	1365.000000	1351.322998	1362.890015	1355.119995	1854000	1355.119995	18.455994	16.500711	0.000000	-6.050049	69.643642	1318.086336	1328.551838	0

139 rows × 14 columns

Remove the first 29 rows of data or days.

```
#Remove the first 29 days of data
df = df[29:]
#Show the data set
df
```

Date	High	Low	Open	Close	Volume	Adj Close	MACD	Signal_Line	up	down	RSI	SMA	EMA	Target
2019-07-15	1150.819946	1139.400024	1146.859985	1150.339966	903800	1150.339966	20.316557	17.334901	5.439941	0.000000	62.063456	1096.864327	1113.511647	1
2019-07-16	1158.579956	1145.000000	1146.000000	1153.579956	1238800	1153.579956	21.003375	18.068596	3.239990	0.000000	78.393463	1100.775993	1117.327677	0
2019-07-17	1158.359985	1145.770020	1150.969971	1146.349976	1170000	1146.349976	20.725375	18.599952	0.000000	-7.229980	77.945728	1103.885990	1120.091705	0
2019-07-18	1147.604980	1132.729980	1141.739990	1146.329956	1290700	1146.329956	20.269786	18.933919	0.000000	-0.020020	80.494359	1107.356323	1122.590586	0
2019-07-19	1151.140015	1129.619995	1148.189941	1130.099976	1647200	1130.099976	18.387148	18.824565	0.000000	-16.229980	69.422722	1110.214990	1123.305766	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2019-12-11	1351.199951	1342.670044	1350.839966	1345.020020	850400	1345.020020	16.384554	14.407740	0.359985	0.000000	70.052514	1307.157003	1316.052424	1
2019-12-12	1355.775024	1340.500000	1345.939941	1350.270020	1281000	1350.270020	17.132277	14.952647	5.250000	0.000000	72.606290	1310.123002	1319.311243	0
2019-12-13	1353.093018	1343.869995	1347.949951	1347.829956	1549600	1347.829956	17.328210	15.427760	0.000000	-2.440063	75.083593	1313.047001	1322.027311	1
2019-12-16	1364.680054	1352.670044	1356.500000	1361.170044	1397300	1361.170044	18.348414	16.011891	13.340088	0.000000	75.548668	1315.961336	1325.755190	0
2019-12-17	1365.000000	1351.322998	1362.890015	1355.119995	1854000	1355.119995	18.455994	16.500711	0.000000	-6.050049	69.643642	1318.086336	1328.551838	0

110 rows × 14 columns

Split the data set into a feature/independent data set (X) and a target/dependent data set (Y).

```
#Split the data set into a feature or independent data set (X) and a target or dependent data set (Y)
keep_columns = ['Close', 'MACD', 'Signal_Line', 'RSI', 'SMA', 'EMA']
X = df[keep_columns].values
Y = df['Target'].values
```

Split the data again, but this time into 80% training and 20% testing data sets.

```
#Split the data again but this time into 80% training and 20% testing data sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2)
```

Create and Train the Machine Learning Model

Create and train the model.

```
#Create and train the model
tree = DecisionTreeClassifier().fit(X_train, Y_train)
```

Check how well the model did on the training data.

```
#Check how well the SVC Model on training data
print(tree.score(X_train, Y_train))
```

```
1.0
```

Check how well the model did on the testing data.

```
#Check the SVC Model on the test data set
print(tree.score(X_test, Y_test))
```

```
0.6818181818181818
```

Get the classification report to see how well the model performed.

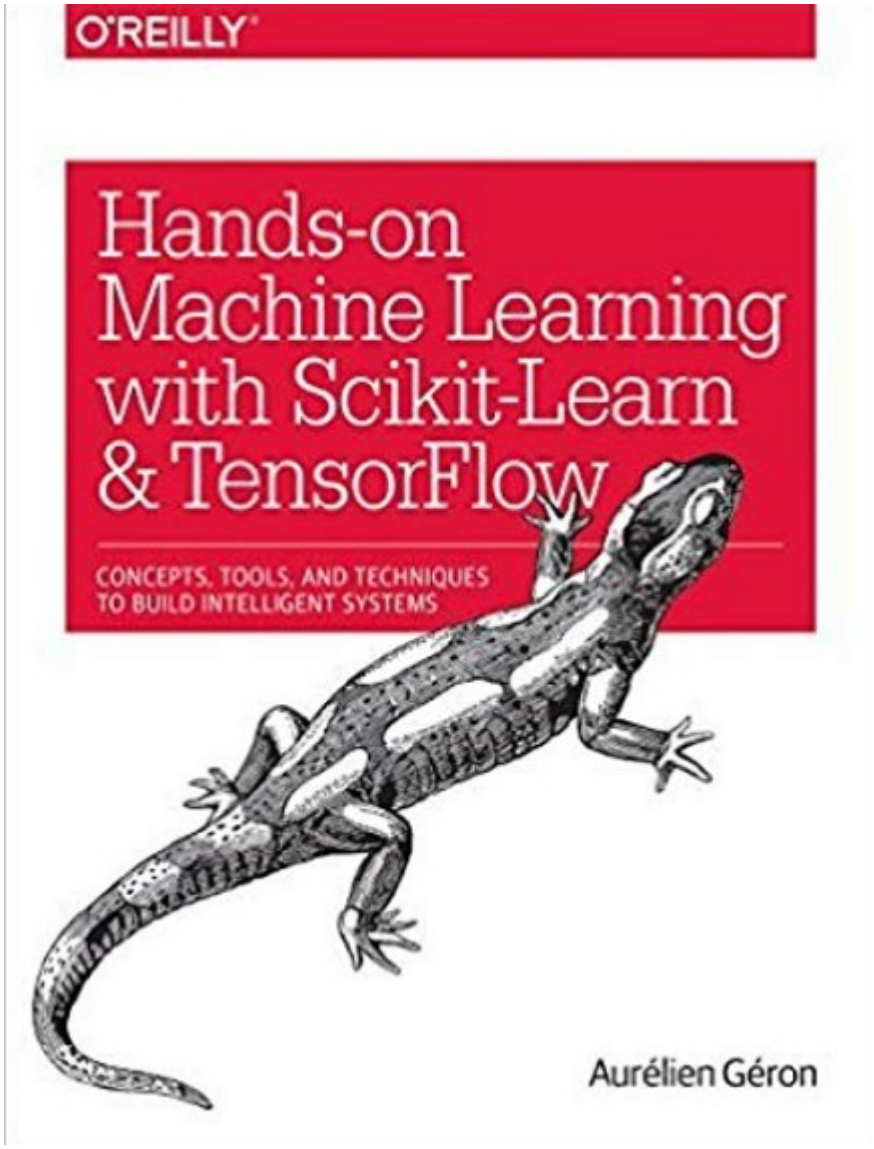
```
from sklearn.metrics import classification_report
print(classification_report(Y_test, rbf_svc_prediction))
```

	precision	recall	f1-score	support
0	0.62	0.80	0.70	10
1	0.78	0.58	0.67	12
accuracy			0.68	22
macro avg	0.70	0.69	0.68	22
weighted avg	0.70	0.68	0.68	22

It looks like this model gave an accuracy score of about 68.18%. This model did better than guessing or flipping a coin which is encouraging, but with an accuracy level at 68.18% on this small set of data, it most certainly is not ready for real world trading, but this model is promising for exploring more on Machine Learning Classifiers for stock price movements. Maybe the model can be improved upon with the use of other indicators, more data, parameter tuning and more analysis.

If you want to start an investment portfolio, then sign up with WeBull using this link and get **FREE stocks** just for opening an account and funding it with an initial deposit of $100 or more! It's free stocks that you can either sell, play with or create your own trading strategy with. For a free stock trading app, I highly recommend it.

If you are interested in reading more on machine learning to immediately get started with problems and examples then I strongly recommend you check out **Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems**. It is a great book for helping beginners learn how to write machine learning programs, and understanding machine learning concepts.



Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems

Thanks for reading this article I hope it was entertaining to you all! If you enjoyed this article and found it helpful please leave some claps to show your appreciation. If you aren't a member of Medium already, then consider becoming a member if not for my articles then for all of the other amazing articles & authors on this site. You can easily become a member of Medium by using the link here. Keep up the learning, and if you like finance, computer science, or programming please visit and subscribe to my YouTube channels (randerson112358 & computer science).

Machine Learning
Python
Stock Market
Finance
Money

Written by randerson112358

2.6K Followers

A simple programmer: https://www.youtube.com/user/randerson112358
https://www.youtube.com/channel/UCbmb5IoBtHZTpYZC