Building Up Trading Strategies Using Machine Learning

Sheil Sarda, Yuezhong Chen

Abstract:

In this project, we aim to build up our trading strategies that can compute the financial data of certain stocks and find out which one of them are reasonable to be invested in. We introduce methods like yfinance to get stocks historical performance, such us closed stock price, open price and close price. Additionally, we include Linear Regression Model, Autoencoders Model and Neural Network Model as the regression methods. As a result, we should successfully achieve a portfolio that gives us higher returns comparing to S&P500 index.

Motivation:

Finance has been seeking to meet the needs of today's globalized modern economies as the stock markets become more and more well-developed after 21 centuries. In order to conduct activities in this complex and changing environment, introducing the Machine Learning Methods to deal with large number of datasets of historical stock prices start to be necessary in selecting the most profitable stocks. We are trying to figure out an effective way in picking the stocks with the comparative high rate of return under the analysis of their previous closed prices. The characters of a growing stock consist of high return, relative fast growth rate and low risk of investment. So, we will also introduce some technical indicators in measuring those corresponding characters.

Related Work:

"Predicting Stable Portfolios Using Machine Learning" by Nandita Dwivedi is aimed to generate stable portfolios on predicted stock prices for the following quarter. This paper focuses on the extent of the influences on the stock prices given by the quarterly and annual financial reports, how the sentiments of historical financial data changes the prediction of stock prices and the approaches to build up a stable portfolio based on the predicted values. Using the NLTK VADER as the sentiment analyzer helps extract sentiments on very large text data from SEC filings, which are significant to predict future stock trends.

"Intelligent Portfolio Construction: Machine-Learning enabled Mean-Variance Optimization" by Ghali Tadlaoui has the goal to forecast stock market level direction based on technical analysis, which is the use of statistical studies of trading data to forecast prices. The data is preprocessed to be used as input to the Random Forest Algorithm, which is an algorithm based on a set of N identical decision trees. Also, the

GARCH (Generalized Autoregressive Conditional Heteroskedasticity) model is used to capture the volatility of the returns.

“Portfolio Optimization-Based Stock Prediction Using Long-Short Term Memory Network in Quantitiative Trading” by Van-Dai Ta and CHUAN-MING Liu is aimed to use a long short-term memory (LSTM) network, which is a special kind of RNN, to predict stock movement based on historical data. And multiple portfolio optimization techniques, including equal-weighted modeling, simulation modeling Monte Carlo simulation, and optimization modeling mean variant optimization are used to construct an efficient portfolio. As a result, the LSTM prediction model could obtain high accuracy from stock prediction.

In “A Machine Learning Framework for Stock Selection” by XingYu Fu and his partners, they characterized stocks with respect to the return-to-volatility ratio by utilizing the algorithms Logistic Regression, Random Forest and Deep Neural Network.

Dataset:

The financial market is a complex, evolutionary and non-linear dynamical system. Stock market prediction is one of the most challenging time series forecasting problems because it is tough to isolate the signal from the noise. From a theoretical perspective, the efficient market hypothesis (EMH) suggests that financial time series data is best characterized as random, independent and Gaussian. Through our paper, we hope to prove that there exist long term patterns in the time series data of stock price data which can help us predict its future value.

Technical Analysis (TA) is an approach to investing that analyzes the statistics generated by market activity with an eye toward finding patterns useful for choosing future investments. TA attempts to find patterns in the supply and demand of a security, and uses these patterns to infer future price directions. Consequently, TA is indifferent to any data that is not derived from price movements in the stock. For this project, we will use TA to determine the dynamics of past behavior in stock prices, and give the model a view into how other market participants will react to price movements.

Features

In this project, we use data from the Yahoo Finance API (yfinance) to create a time series dataset of the following features:

Open	Daily opening price of the stock
High	Highest trading price of the day
Low	Lowest trading price of the day

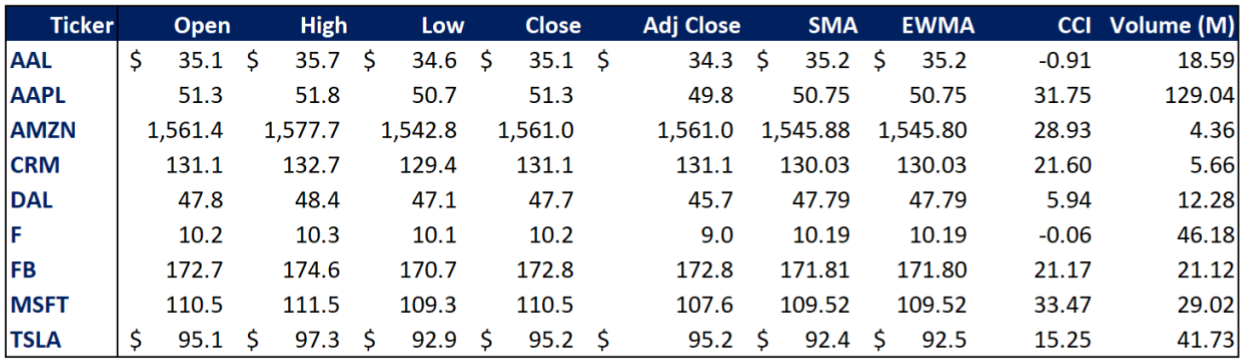
Adjusted Close	Amends the closing price to reflect any dividends, stock splits and follow-on equity offerings
Volume	The number of shares of the stock which changed hands daily
Name	Ticker of the stock (e.g. AAPL, MSFT, etc.)
Simple Moving Average (SMA)	15-day simple moving average of the unadjusted closing price
Exponentially Weighted Moving Average (EWMA)	15-day exponentially weighted moving average of the unadjusted closing price
Commodity Channel Index (CCI)	Measures current price level relative to an average price level based on an average of the high, low and closing price of the day

Basket of Stocks

We selected a bucket of stocks to analyze since we wanted to test our models on a equities in a variety of different industries (Auto, Tech Airlines) and business models (B2C, B2B)

AAL (American Airlines)	AAPL (Apple)	DAL (Delta Air Lines)
FB (Facebook)	AMZN (Amazon)	TSLA (Tesla)
MSFT (Microsoft)	CRM (Saleforce)	F (Ford Motor Company)

Our dataset consists of daily stock prices from 12/1/2016 till 12/1/2020. All in all, this represents 10,764 rows of data. A summary of the mean value of every feature we include for stocks is included below.



Ticker	Open	High	Low	Close	Adj Close	SMA	EWMA	CCI	Volume (M)
AAL	$ 35.1	$ 35.7	$ 34.6	$ 35.1	$ 34.3	$ 35.2	$ 35.2	-0.91	18.59
AAPL	51.3	51.8	50.7	51.3	49.8	50.75	50.75	31.75	129.04
AMZN	1,561.4	1,577.7	1,542.8	1,561.0	1,561.0	1,545.88	1,545.80	28.93	4.36
CRM	131.1	132.7	129.4	131.1	131.1	130.03	130.03	21.60	5.66
DAL	47.8	48.4	47.1	47.7	45.7	47.79	47.79	5.94	12.28
F	10.2	10.3	10.1	10.2	9.0	10.19	10.19	-0.06	46.18
FB	172.7	174.6	170.7	172.8	172.8	171.81	171.80	21.17	21.12
MSFT	110.5	111.5	109.3	110.5	107.6	109.52	109.52	33.47	29.02
TSLA	$ 95.1	$ 97.3	$ 92.9	$ 95.2	$ 95.2	$ 92.4	$ 92.5	15.25	41.73

Summary Plots of Price Movement

A few summary plots of what the adjusted closing prices of stocks look like during this period are included below for your reference:

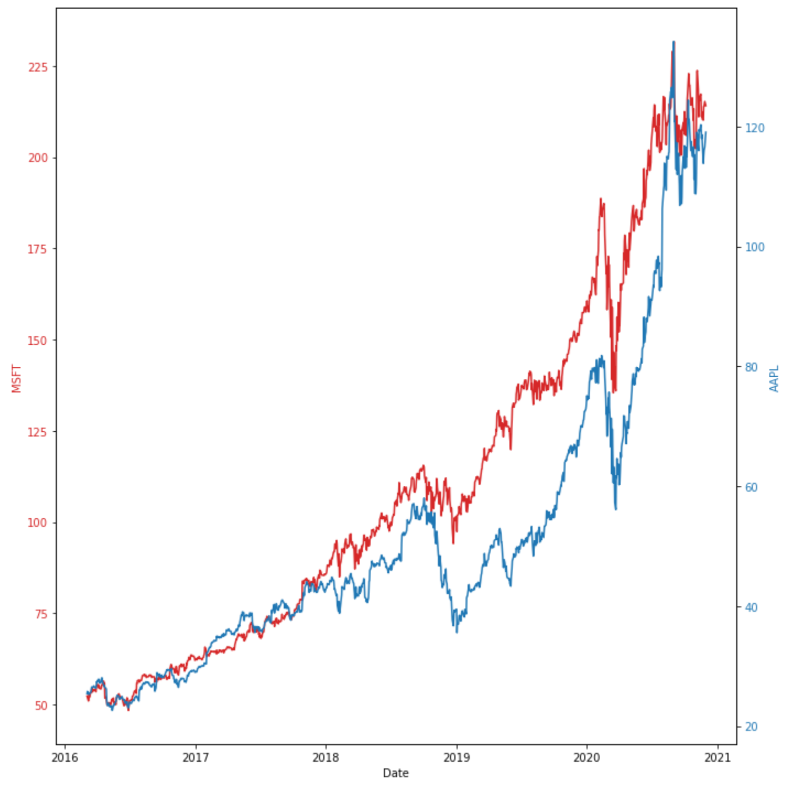


Figure 1. Microsoft (MSFT) vs Apple (AAPL) adjusted closing prices from 12/1/2016 to 12/1/2020

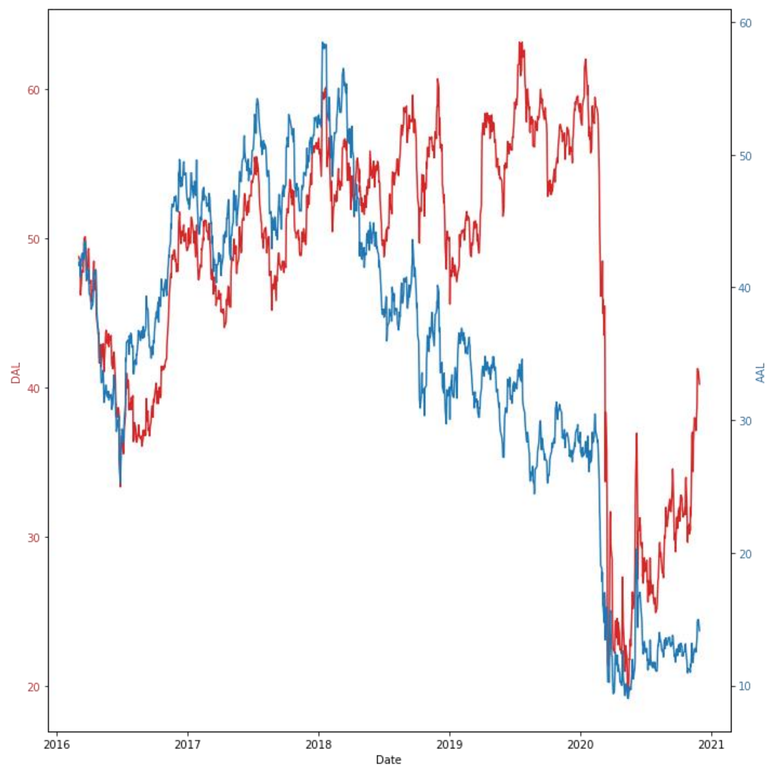


Figure 2. Delta Airlines (DAL) vs American Airlines (AAL) adjusted closing prices from 12/1/2016 to 12/1/2020

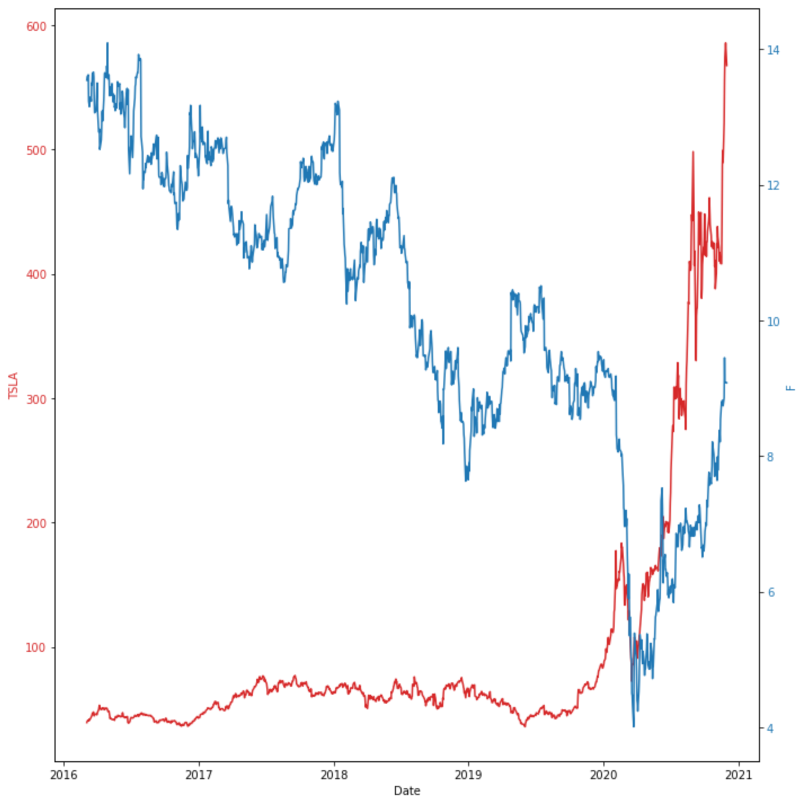


Figure 3. Tesla (TSLA) vs Ford Motor Company (F) adjusted closing prices from 12/1/2016 to 12/1/2020

Problem Formulation:

The process of selecting a portfolio is composed of two stages: the first to analyze the historical data and build an idea on the behavior of assets in the future, and the second one uses these insights to build the portfolio (Markowitz). Our work attempts to combine both Machine Learning and Investment Strategies by using ML to predict the stock direction in the first phase of the portfolio construction. We aim at comparing the performances of a portfolio constructed with the classic structure with one derived from a machine-learning enabled version.

We choose to work only on US Large Cap stocks for generalization purposes. We aim at forecasting expected returns for a set of stocks. This is done in two steps: we first use a supervised learning algorithm to forecast the direction of the stock, we then forecast the amplitude of the move to capture the volatility of the returns.

Performance Metrics:

The application of ML to forecast stock prices has been explored by a variety of researchers before us, primarily using statistical error measurements to highlight their proficiency in solving the task.

A common shortcoming of this typical error measurement is that great statistical error measurements might not be the investments which lead to the greatest profit. If there is very little uncertainty related to the security, the model might do a wonderful job of predicting future prices, but the lack of price movement also caps the upside of the investor. In the extreme example, consider a 1-year $100 Treasury Bill which pays 0% coupons until maturity. The price of this security can be forecasted using time value of money as 100/(1-r) where r is the risk free rate of investments. There is no upside or downside in this investment.

Another shortcoming of using ML for predicting stock prices is that the models don’t do such a good job of adapting to a shift in which factors are driving the stock price at any given point in time, since they lack the context of what investment criterion matters most to investors at any point in time. For instance, as a public company matures, the risk-return profile of the stock changes. At an early stage, companies tend to heavily reinvest their profits to fuel top-line growth, whereas mature companies pay-out a greater percentage of profits as quarterly dividends. Using time series data of stock prices fails to capture this important nuance of changing drivers.

Thus, in our analysis we wish to measure results primarily in terms of dollar valued returns in addition to statistical error measures.

Methods:

Baseline

Our baseline model is Linear Regression, chosen because of its simplicity and ease of implementation. The performance of a linear regression model on the above described dataset is as follows:

Tools for Implementation

We implemented our chosen models and loss functions in Python using Scikit-Learn, Keras, Matplotlib, Numpy and Pandas packages. To obtain our datasets, we used the Yahoo Finance yfinance library.

Experiments:

Neural Network

A neural network is a collection of algorithms that, through a mechanism that mimics the way the human brain works, aim to identify underlying relationships in a set of data. Neural networks, in this context, apply to neuron structures, either organic or artificial in nature. A single-layer neural network is also known as the perceptron model. This neural net comprises layers of input and output. No hidden layers exist in this form of neural network. It takes an input and for each node, calculates the weighted input. After that for classification purposes, it uses an activation function. An artificial neural network in which the nodes do not ever form a cycle is a feed-forward neural network. All of the perceptron are arranged in layers in this neural network where the input layer takes input, and output is created by the output layer.

Regression Methods

Neural Network Regression

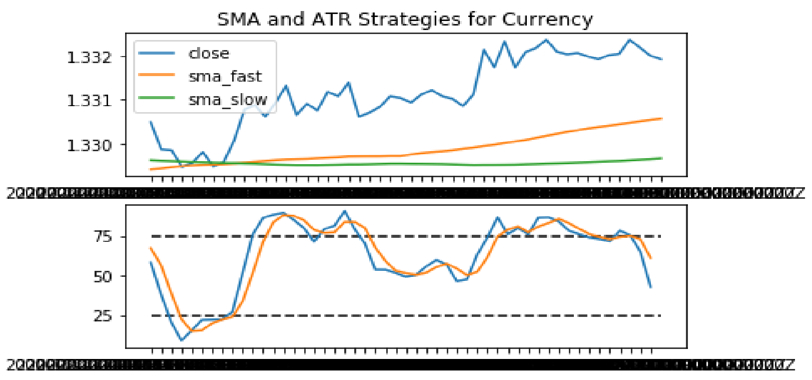
As a function of the inputs, Regression ANNs predicted an output variable. The input features may be categorical or quantitative types, but we need a numeric dependent variable for regression ANNs. Three layers of neurons exist in a shallow neural network: an input layer, a hidden layer and an output layer. A Deep Neural Network (DNN) has more than one hidden layer, which increases the model's complexity and can greatly enhance the capacity of prediction. Neural networks can be reduced to regression models; any form of regression model can "pretend" to be a neural network. For example, this very simple neural network is equivalent to logistic regression, with only one input neuron, one hidden neuron, and one output neuron. It takes several dependent variables = input parameters, multiplies them by their coefficients = weights, and runs them through a function of sigmoid activation and a function of unit step, closely resembling the function of logistic regression with its error term. Using Gradient Descent that employs the gradient of the cost function, neural networks are equipped. The simplest approach assumes that we use the entire dataset to compute the cost function gradient.

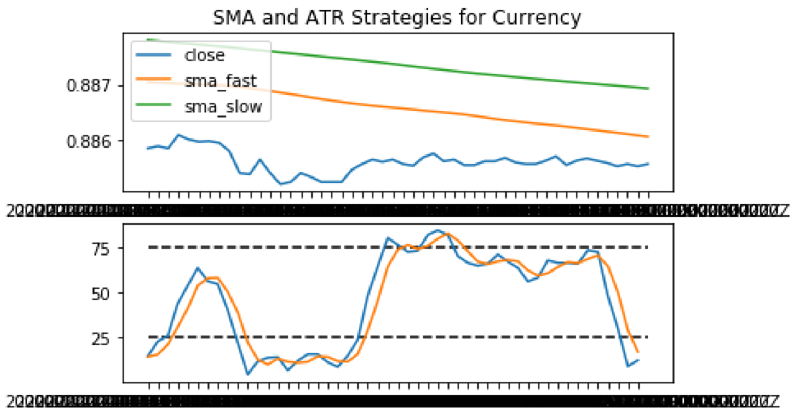
Random Forest Regression

The random forest regression which is a collection of decision trees produces different possible prices of the stock. We train a Random Forest Regressor on the stock price dataset using the SciKit learn library, and a 70-30 split for test and train. Other relevant parameters include a max tree depth of 10 and

Conclusion and Discussion:

We collected the data from NASDAQ websites, the datahub contains hundreds of stock ticker symbols, but we started with the symbols mainly targeted at big IT companies, such as Apple, Amazon.com, Facebook, Tesla, Inc. and etc. Our trading strategies consist of SMA, EWMA, ATR, CAGR, Volatility and etc. The stocks with satisfactory historical trends will be considered as the targeted stocks included in our portfolio. Also, to make our strategies more reasonable, the Sharpe Ratio calculation, Magic Formula, Piotroski F-Score and returns from dividends should be taken into consideration. To backtest our strategies, we compared our rate of returns with that of S&P 500 and found out that the portfolio is comparatively more profitable. For the Backtesting part, we use OANDA as an API and create a demo account, where users are restricted to buy or sell currencies. Below are the plots for GBP/USD and AUD/USD by using SMA Crossover and ATR as our trading strategies.





During the experiment, we used the Linear regression as the baseline model and introduced Autoencoders, Neural networks models for the analysis over higher levels of complexity. It should be more reasonable if we add Slope, Renko Chart and some other indicators to make the strategies more comprehensive, but the results come to be less practical since very little stocks will be picked for the investment under such demanding criteria. Furthermore, there remains a time gap between picking up the right stocks and trading, thus, the investor may not buy or sell a stock at the right time if he doesn't execute immediately. As a result, an API can be used to improve the performance, by which the users are allowed to trade through running the code on an IDE.