Machine Learning Engineer Nanodegree

Capstone Project: Stock Price Indicator

Ömer Faruk BÜLBÜL
June 22nd, 2019

I. Definition

Project Overview

Investment firms, hedge funds and even individuals have been using financial models to better understand market behaviour and make profitable investments and trades. Since investor behaviour often deviates from logic and reason, and investors display many behaviour biases that influence their investment decision-making processes[1], it is important to construct an investment strategy in terms of strict algorithmic rules.

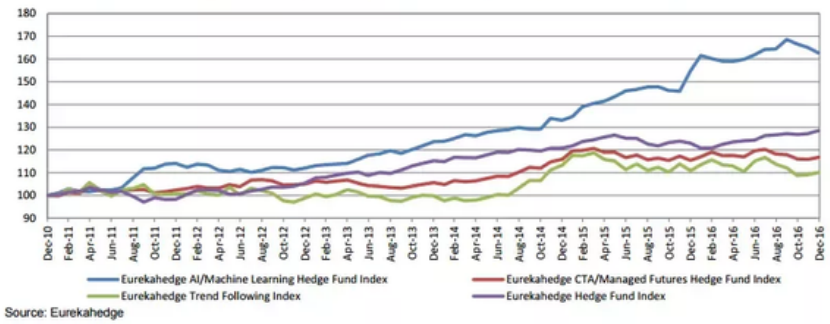
Financial modeling is the task of building an abstract representation (a model) of a real world financial situation.[2] This is a mathematical model designed to represent (a simplified version of) the performance of a financial asset or portfolio of a business, project, or any other investment. Realistic financial models which require great effort are than used by financial analysts to anticipate the impact of an economic policy change or any other event on a company's stock to predict the future price of the stock. In quantitative finance, financial modeling entails the development of a sophisticated mathematical model. Models here deal with asset prices, market movements, portfolio returns and the like.

There are so many factors involved in the prediction – physical factors vs. physhological, rational and irrational behaviour, etc. All these aspects combine to make share prices volatile and very difficult to predict with a high degree of accuracy. Since finance area dynamics are very complex and stock prices depend on so many factors, machine learning can be a good solution if a suitable subset of these factors is selected and a reasonable performance is targeted. A wealth of information is available in the form of historical stock prices and company performance data, suitable for machine learning algorithms to process for constructing a financial model predicting the stock price of companies. There are lots of websites and web services for gathering historical stock price and financial performance data including Yahoo Finance, Google Finance etc.

There are many successful implementations to make an estimator for stock prices using machine learning methods like linear regression, svm, lstm, arima etc. one of which creates a framework with neural networks and decision forests.[3] In their "A machine learning based stock trading framework using technical and economic analysis" work they have managed to beat S&P500 Index by far according to charts provided.

Hedge fund research firm Eurekahedge has published some informative data. The chart below displays the performance of the Eurekahedge AI/Machine Learning Hedge Fund Index vs. traditional quant and hedge funds from 2010 to 2016. The Index tracks 23 funds in total, of which 12 continue to be live.

Figure 1: AI/Machine Learning Hedge Fund Index vs. quants and traditional hedge funds



Source: Eurekahedge

With following table provided, Eurekahedge notes that:

“AI/machine learning hedge funds have outperformed both traditional quants and the average hedge fund since 2010, delivering annualized returns of 8.44% over this period compared with 2.62%, 1.62% and 4.27% for CTA’s, trend-followers

and the average global hedge fund respectively." Performance in numbers – AI/Machine Learning Hedge Fund Index vs. quants and traditional hedge funds

	Eurekahedge AI/Machine Learning Hedge Fund Index	Eurekahedge CTA/Managed Futures Hedge Fund Index	Eurekahedge Trend Following Index	Eurekahedge Hedge Fund Index
2011	14.10%	2.33%	0.71%	(1.75%)
2012	(1.80%)	2.66%	(1.86%)	7.34%
2013	10.34%	0.55%	1.02%	9.24%
2014	7.64%	9.66%	13.44%	4.89%
2015	16.40%	(0.31%)	(2.18%)	1.78%
2016	5.01%	1.15%	(0.62%)	4.48%
5 year annualised returns	7.35%	2.68%	1.80%	5.51%
5 year annualised volatility	4.95%	4.18%	7.13%	3.20%
5 year Sharpe Ratio (RFR=1%)	1.28	0.40	0.11	1.41
3 year annualised returns	9.57%	3.41%	3.31%	3.71%
3 year annualised volatility	5.61%	4.63%	7.78%	3.03%
3 year Sharpe Ratio (RFR=1%)	1.53	0.52	0.30	0.89
2 year annualised returns	10.56%	0.42%	(1.40%)	3.12%
2 year annualised volatility	6.31%	4.90%	8.07%	3.31%
2 year Sharpe Ratio (RFR=1%)	1.51	(0.12)	(0.30)	0.64

With the information provided by Eurekahedge it is clear that machine learning algorithms are very usefull for constructing successfull investment strategies. Similarly I would like to implement a stock price indicator with machine learning methodologies aiming to predict future stock price for a given stock.

Problem Statement

The problem is predicting the actual value of adjusted close price of a requested stock for a requested day with 5-10 days of previous data in the form of average values provided.

For this project, I will try to build a stock price predictor that takes daily trading data over 5 or 10 days as input of requested stock, and outputs projected estimates for given query dates. The predictor given open price(open), highest price(highest), volume, adjusted close price(adjusted close) will only predict Adjusted Close price for a given stock. The predicted value can easily be compared and a benchmark can easily be set up with actual adjusted price of the requested date for various dates upon request.

Since we are trying to predict a value this problem is a regression problem. We take historical stock market values and try to predict adjusted close price of the stock. We will be constructing several predictive models which investigates the relationship between a dependent (target) and independent variable (s) (predictor). We will try to forecast, time series modeling and finding the causal effect relationship between the variables.

Metrics

I would like to use R2 and RMSE(root mean squared error) for evaluation metrics which are frequently used for estimation error calculations. With the following formulas:

Mean squared error	$\mathrm{MSE} = \frac{1}{n}\sum_{t=1}^{n} e_t^2$
Root mean squared error	$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$
Mean absolute error	$\mathrm{MAE} = \frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$
Mean absolute percentage error	$\mathrm{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n} \left\lvert \frac{e_t}{y_t} \right\rvert$

$$\hat{R}^2 = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2} = 1 - \frac{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \bar{Y})^2}$$

We will look at the deviation by calculating the RMSE of the model. If the deviation is big RMSE score will be far away from 0. Simply put, the lower the value the better and 0 means the model is perfect. Since we do not want high deviation for our predictions we would like to have RMSE score near to 0.

For a regression with an intercept, $R2$ is between 0 and 1, and from its definition $R2=1-SSE/TSS$ we can find an interpretation: SSE/TSS is the sum of squared errors divided by the total sum of squares, so it is the fraction ot the total sum of squares that is contained in the error term. So one minus this is the fraction of the total sum of squares that is not in the error, or $R2$ is the fraction of the total sum of squares that is 'explained by' the regression.

II. Analysis

(approx. 2-4 pages)

Data Exploration

I used 13 different stocks' historical stock prices saved to "data" folder of this project. Each file is in the csv format and containing approximately 9 years of historical stock market data. Data saved are from Yahoo Finance website with python code and open to public use. The symbols of the stocks in alphabetical order is: AAPL, AMZN, AVGO, CSCO, MA, MSFT, NVDA, NVS, PFE, QCOM, TXN, V, WNT. Historical Data of stock prices can be easily gathered using https://pypi.org/project/yahoo-finance/ project or pandas data reader commented in first code cell of the project.

In the csv files columns are: 1- Date: The date which market was open and stock is exchanged. 2- Open: The price which first transaction of the day was occured. 3- High: Highest price of the day 4- Low: Lowest price of the day 5- Close: The price which last transaction occured. 6- Adjusted Close: The adjusted closing price is a useful tool when examining historical returns because it gives analysts an accurate representation of the firm's equity value beyond the simple market price. It accounts for all corporate actions such as stock splits, dividends/distributions and rights offerings. 7- Volume: The number of shares that changed hands during a given day.

In order to have good predictions we need to get trend of the stock market. If close, adjusted close price of a stock is greater than open price it clearly shows a bullish market rather than a bearish one. So relations between open price and close price will be very important for predicting the requested days adjusted close price. Also volume is very important since volume reflects the intensity (strength) of a stock. Volume also provides an indication of the quality of a price trend and the liquidity of a stock. Highest price and lowest price values when compared to open price and close price are used to interpret trend of the stock by financers. A trend analysis can be done with volume and price like in the below table [4]. General Rules in Volume Analysis:

Volume	Price	Interpretation
Increasing	Rising	bullish
Decreasing	Falling	bullish
Increasing	Falling	bearish
Decreasing	Rising	bearish

Here is an example of data structure:

Date	Open	High	Low	Close	Adj Close	Volume
2019-01-28	155.789993	156.330002	153.660004	156.300003	155.632523	26192100
2019-01-29	156.250000	158.130005	154.110001	154.679993	154.019440	41587200
2019-01-30	163.250000	166.149994	160.229996	165.250000	164.544296	61109800

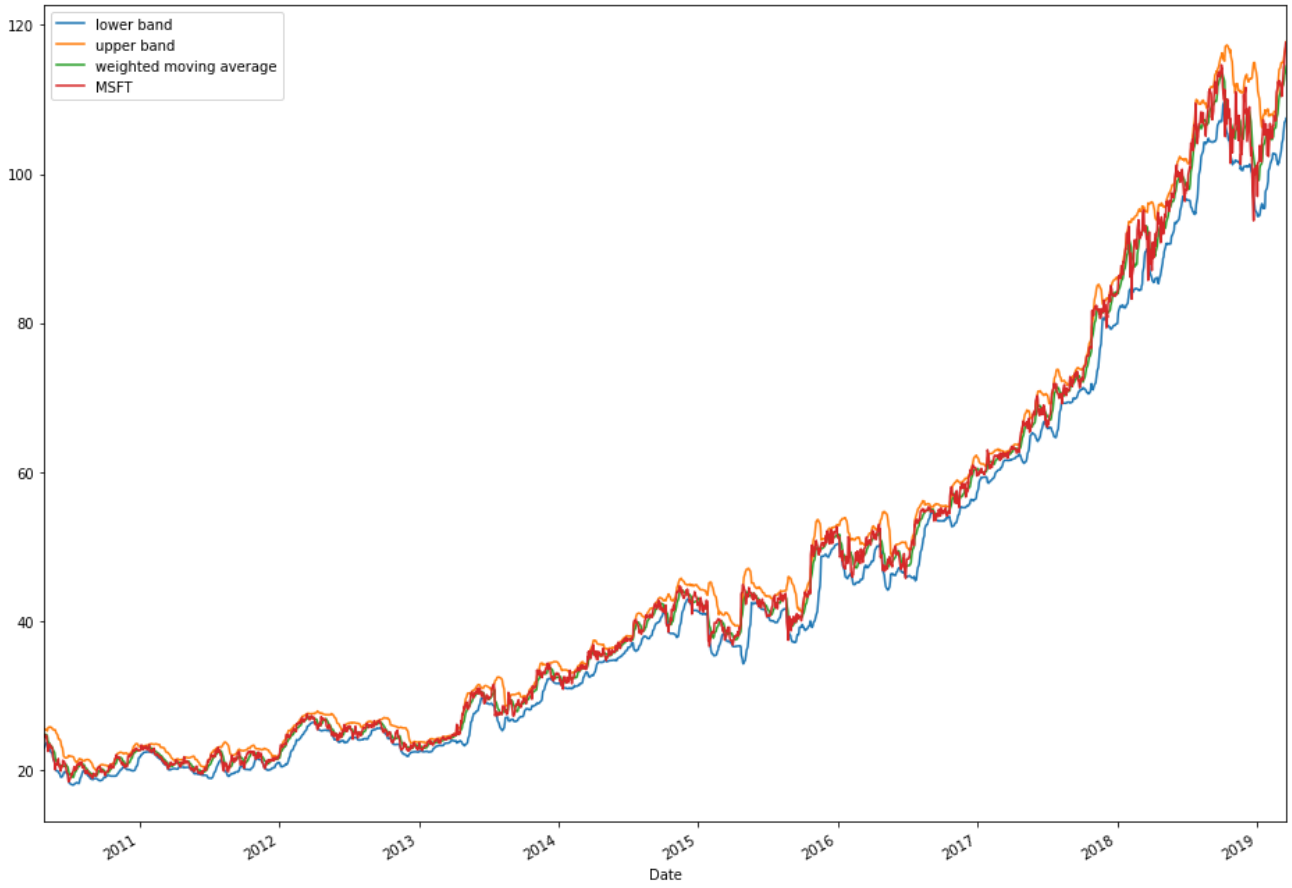
Typically every dataframe(stock historical file) have more than 2000 rows. In cell 3 I checked if a null value was existing in the data but i could not found any, so I did not need to do anything to cover null values.

```
AMZN.describe()
```

	High	Low	Open	Close	Volume	Adj Close
count	2264.000000	2264.000000	2264.000000	2264.000000	2.264000e+03	2264.000000
mean	594.079943	581.213975	588.092138	587.949607	4.482567e+06	587.949607
std	497.885293	486.777432	492.993055	492.473424	2.676545e+06	492.473424
min	111.290001	105.800003	105.930000	108.610001	9.844000e+05	108.610001
25%	229.035000	222.980000	225.862499	226.129997	2.781600e+06	226.129997
50%	359.330002	349.425003	355.264999	355.610001	3.837850e+06	355.610001
75%	815.782516	803.580002	809.322495	809.839981	5.310550e+06	809.839981
max	2050.500000	2013.000000	2038.109985	2039.510010	4.242110e+07	2039.510010

Exploratory Visualization

In cell 4 I added frequetly used trend analysis formulas to the data as additional columns in order to have better results and in cell 5 I plotted original adjusted close price, bollinger bands and weighted moving average calculated. We can consider bollinder bands as standard deviation margins from current price of the stock. Also moving averages provide trend information for the data.



Algorithms and Techniques

So we have timeseries data that we would like to have a regression and predict adjusted close price of the next market day. So we have lots of methodologies we could apply to the problem. Namely, we can apply simple linear regression, support vector machines regression algorithms, logistic regression, multiple regression algortihm, long short term memory(LSTM) regression etc.

For the project I chose following techniques to tackle the problem:

1- SVM Linear/Polynomial Regression: This supervised machine learning algorithm has strong regularization and can be leveraged both for classification or regression challenges. They are characterized by usage of kernels, the sparseness of the solution and the capacity control gained by acting on the margin, or on number of support vectors, etc. The capacity of the system is controlled by parameters that do not depend on the dimensionality of feature space. Since the SVM algorithm operates natively on numeric attributes, it uses a z-score normalization on numeric attributes. In regression, Support Vector

Machines algorithms use epsilon-insensitivity (margin of tolerance) loss function to solve regression problems. Support vector machines regression algorithms has found several applications in the oil and gas industry, classification of images and text and hypertext categorization. In the oilfields, it is specifically leveraged for exploration to understand the position of layers of rocks and create 2D and 3D models as a representation of the subsoil. I chose error penalty coefficient C as 100 for both linear and polynomial regressors. For the polynomial regressor I chose gamma as auto mode, polynomial degree as 3 and epsilon as 0.1.

2- LSTM: LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the exploding and vanishing gradient problems that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications Since The Long Short-Term Memory recurrent neural network has the promise of learning long sequences of observations I chose it for using it as a regressor for a time period of approximately 9 years. I chose 100 nodes with dropout rate of 20% with activation linear and optimizer as rmrsprop.

Benchmark

As a benchmark for the problem, simple linear regression is a statistical method that enables users to summarise and study relationships between two continuous (quantitative) variables. Linear regression is a linear model wherein a model that assumes a linear relationship between the input variables (x) and the single output variable (y). Some of the most popular applications of Linear regression algorithm are in financial portfolio prediction, salary forecasting, real estate predictions and in traffic in arriving at ETAs. I have chosen simple linear regression method for benchmarking because it is the simplest method for regression. This benchmark is frequently used for regression problems. We know that R2 score will be very near to 1 when we have a good predictor for the problem. From the results very near to 1 at the end of the report we can say that a simple linear regression method is a challenging benchmark to be selected for the methods implemented. For the linear regression I chose learning rate as 0.2 having 10 nodes and 10 epochs.

III. Methodology

Data Preprocessing

For preprocessing data, I needed to eliminate NaN or null values, normalize, put additional frequently used trend analysis information methodologies in order to have a successfull learning. I have followed the following steps:

1- Download data from Yahoo Finance servers with pandas datareader into 13 different pandas dataframes. 2- Check for null values if there is any. I encountered no null values so I made no action for null values. 3- Add MACD (Moving Average Convergence Divergence) Data for column Adjusted Close for every data frame for 26, 12 day periods. This is a very frequently used trend information. 4- Add RSI(Relative Strength Index) for period 14 days. 5- Add Bollinger Bands namely standard deviation margins informations in column up and low. 6- Delete first 25 rows of every dataframe since we have added NaN values to newly added columns. 7- Plot data with newly added information in order to check for any mistakes not confirming with data 8- Normalize data with sklearn preprocessing minmaxscaler and multipy adjusted close with 100 to scale up to 100 (I also tried without normalizing the adjusted column and saved results for that case) 9- seperate y values column namely adjusted close price column from dataframes 10- Shift backward 1 days y values dataframes since we are trying to predict (n+1)th day with nth day input 11- Also for lstm network I needed to turn data into three dimensional array

Implementation

I have used Jupyter Notebooks as development environment backed up with python 3.6 having numpy, scipy, pandas, pandas-datareader, matplotlib, scikit-learn, tensorflow, keras.

After preprocessing the data I have done the following steps:

1- Plan: I have planned implentation in the following order as stated between code cell 6 and 7: Benchmark Model, RMSE and R2 Validation, Solution Model, Create Test Flow

2- Create Benchmark Model: I have selected simple linear regression for benchmarking. With Keras: The Python Deep Learning library it is very easy to implement a simple linear regression by adding a dense layer consisting of only 1 node with activation type of linear. With Stochastic gradient descent as optimizer with learning rate 0.2.

3- Create Metrics I have chosen to use two metrics for comparing the results of learning techniques R2 and RMSE. These metrics are already provided by sklearn and I added a little informative lines to print stating method and score.

4- Create Solution Models: In order to use in test flow I have implemented 3 methods returning models created. First method of support vector machine regressor is very straight forward with sklearn libraries having the option linear and penalty coefficient 100. todo: take screenshots

Second method of SVM regressor has polynomial kernel and gamma is in auto mode with third degree polynomial fitting.

Thirdly, I have created an lstm model having 100 nodes with dropout rate of 0.2 followed by a flattening layer and a dense layer of one node for final prediction.

5- For test flow I wondered what will be the effect of using single model for 13 stocks and 13 diffreent models for each stock. So for the test flow I set a parameter for calling different models and single model. I simply created models, split data fit models, get predictions and than keep the r2, rmse scores after calling the check score.

Refinement

Here is an initial solution result for the problem. It is clear that RMSE namely deviation of the error depends on the value of the price so I decided to normalize the adj. close value and than run the solution again. It is clear that stocks with heigher prices rmse scores reduced and rmse values approached to each other.

linear_rmse	linear_r2	svm_linear_rmse	svm_linear_r	svm_polynor	svm_polynor	lstm_rmse	lstm_r2
2.192528	0.998134	2.004182	0.99844	0.839159	0.999727	6.9289	0.981359
7.805897	0.999726	5.369823	0.999871	6.121477	0.999832	25.304551	0.997126
2.037792	0.999393	1.878048	0.999484	1.024915	0.999846	4.122919	0.997514
0.373997	0.998415	0.318431	0.998851	0.180695	0.99963	0.946034	0.989856
0.922202	0.999698	0.537153	0.999898	0.381595	0.999948	1.961162	0.998634
0.501376	0.999621	0.354377	0.999811	0.268849	0.999891	1.408609	0.997009
1.319128	0.999684	0.274242	0.999986	0.400191	0.999971	3.076882	0.998283
0.70635	0.9982	0.659137	0.998432	0.438073	0.999308	6.199673	0.861317
0.261775	0.999008	0.242498	0.999149	0.174377	0.99956	2.330596	0.921378
0.781117	0.992585	0.72915	0.993539	0.483466	0.997159	4.308404	0.774404
0.757133	0.999201	0.565553	0.999554	0.402025	0.999775	1.444661	0.997091
0.594605	0.99975	0.413177	0.999879	0.258257	0.999953	1.486428	0.99844
0.846397	0.996935	0.77153	0.997453	0.383184	0.999372	2.377591	0.975812

linear_rmse	linear_r2	svm_linear_rmse	svm_linear_r	svm_polynor	svm_polynor	lstm_rmse	lstm_r2
2.120015	0.998255	2.004182	0.99844	0.839159	0.999727	7.67189	0.977147
8.308104	0.99969	5.369823	0.999871	6.121477	0.999832	22.541021	0.997719
2.027183	0.999399	1.878048	0.999484	1.024915	0.999846	4.654158	0.996832
0.368127	0.998464	0.318431	0.998851	0.180695	0.99963	1.20297	0.983598
0.963566	0.99967	0.537153	0.999898	0.381595	0.999948	3.019329	0.996763
0.502139	0.99962	0.354377	0.999811	0.268849	0.999891	1.064994	0.99829
1.31664	0.999686	0.274242	0.999986	0.400191	0.999971	2.965975	0.998404
0.652179	0.998465	0.659137	0.998432	0.438073	0.999308	6.336198	0.855142
0.263022	0.998999	0.242498	0.999149	0.174377	0.99956	3.31033	0.841382
0.841118	0.991402	0.72915	0.993539	0.483466	0.997159	3.840367	0.820756
0.703717	0.99931	0.565553	0.999554	0.402025	0.999775	1.228029	0.997898
0.59493	0.99975	0.413177	0.999879	0.258257	0.999953	1.60534	0.99818
0.848118	0.996922	0.77153	0.997453	0.383184	0.999372	1.98252	0.983183

	linear_rmse	linear_r2	svm_linear_rmse	svm_linear_r2	svm_polynomial_rmse	svm_polynomial_r2	lstm_rmse	lstm_r2
0	1.338723	0.996995	0.935777	0.998532	0.355210	0.999788	1.762615	0.994791
1	0.481175	0.999612	0.051701	0.999996	0.080450	0.999989	1.374554	0.996838
2	0.740714	0.999356	0.642850	0.999515	0.349160	0.999857	2.164429	0.994503
3	0.958284	0.998120	0.760326	0.998816	0.452071	0.999582	1.689076	0.994158
4	0.451842	0.999666	0.239102	0.999907	0.153518	0.999961	1.115738	0.997966
5	0.571461	0.999516	0.357395	0.999811	0.271301	0.999891	1.192665	0.997891
6	0.648095	0.999400	0.089160	0.999989	0.092692	0.999988	1.234466	0.997824
7	1.299821	0.997686	1.075784	0.998415	0.734936	0.999260	3.424758	0.983938
8	1.444055	0.996181	0.691003	0.999126	0.520144	0.999505	2.060950	0.992222
9	1.947591	0.989490	1.539399	0.993434	1.069459	0.996831	4.339100	0.947834
10	0.734320	0.999286	0.580172	0.999554	0.413732	0.999773	1.535708	0.996875
11	0.450135	0.999713	0.290247	0.999881	0.175407	0.999956	0.997092	0.998591
12	1.508813	0.995496	1.152131	0.997374	0.589213	0.999313	2.325818	0.989299

	linear_rmse	linear_r2	svm_linear_rmse	svm_linear_r2	svm_polynomial_rmse	svm_polynomial_r2	lstm_rmse	lstm_r2
0	1.340637	0.996986	0.935777	0.998532	0.355210	0.999788	1.924135	0.993792
1	0.474834	0.999623	0.051701	0.999996	0.080450	0.999989	1.412788	0.996659
2	0.757063	0.999327	0.642850	0.999515	0.349160	0.999857	1.927947	0.995638
3	0.996786	0.997965	0.760326	0.998816	0.452071	0.999582	1.553540	0.995058
4	0.470756	0.999638	0.239102	0.999907	0.153518	0.999961	1.201012	0.997643
5	0.544455	0.999560	0.357395	0.999811	0.271301	0.999891	1.267394	0.997618
6	0.660917	0.999376	0.089160	0.999989	0.092692	0.999988	1.505960	0.996762
7	1.298960	0.997689	1.075784	0.998415	0.734936	0.999260	3.862164	0.979573
8	1.277722	0.997010	0.691003	0.999126	0.520144	0.999505	2.113977	0.991816
9	1.857843	0.990437	1.539399	0.993434	1.069459	0.996831	4.549580	0.942650
10	0.732865	0.999288	0.580172	0.999554	0.413732	0.999773	1.554728	0.996797
11	0.441273	0.999724	0.290247	0.999881	0.175407	0.999956	1.687712	0.995964
12	1.554954	0.995217	1.152131	0.997374	0.589213	0.999313	2.572681	0.986906

I than tried having two layers of lstm of 10 nodes to have better reults but it ended in a slightly worse result. Also trying tanh and softmax functions ended in local minimas having poor results. Since benchmark and lstm results are very near I stopped trying to make lstm results better.
As far as the methods I tried best performing method is SVR polynomial. This method has very good R2 mean for stocks which is 0.9995363077 and very near to 1. This indicates that this model can be definetly used as an estimator for stock prices given the supportive information extracted from the stock market prices.

I was expecting better results in LSTM technique but SVR polynomial was better in results. In the next few weeks I will try configure a better LSTM network which can outperform SVR polynomial.

For the SVR definetly polynomial kernel performs better than linear one and also it is better than the benchmark. I have added a grid search similar section at the end of bar charts for SVM polynomial solution configuration generating the following results. For the C Value C = 50, 100, 200 vales are tried and following mean values are gathered. C=200 can be selected

	RMSE	R2
default	0.404407	0.999515
c200	0.382911	0.999563
c50	0.436590	0.999446

For the gamma value gammma = 0.1, 0.2, 0.4, 0.8 values are tried and following results are gathered. gamma = 0.8 can be selected

	RMSE	R2
default	0.404407	0.999515

	RMSE	R2
gamma_dot2	0.364122	0.999563
rmse_gamma_dot4	0.315261	0.999716

For polynomial if we use degree 4 it gives better results.

	RMSE	R2
default	0.404407	0.999515
c200	0.382911	0.999563
c50	0.436590	0.999446
gamma_dot2	0.364122	0.999605
rmse_gamma_dot8	0.315261	0.999716
degree4	0.280099	0.999778

IV. Results

Model Evaluation and Validation

The final model as optimized in a manual grid search is SVR with parameters gamma=0.8, C=200, degree=3 since it gives the best results overall. In the refinement section first two images of summary including all the methodologies both for R2 and RMSE scores the best option was the SVR method. For checking the robustness of the solution when we apply gaussian noise to Open, High, Close columns of the test data we get the following tables. When we apply gaussian noise we see that gathered information is still enough to predict better than most of the 0.75 R2 most of the time. It looks like small perturbations made to input data does not affect performance drastically.

Original Data		Perturbed Data	
R2	RMSE	R2	RMSE
0.9348	0.9300	0.8780	1.2720
0.9996	0.0653	0.9129	0.9963
0.7605	2.6058	0.7573	2.6230
0.8539	1.4107	0.7500	1.8454
0.9979	0.1005	0.7402	1.1089
0.8718	0.9552	0.7720	1.2736
0.9952	0.2580	0.9192	1.0579
0.9043	1.2920	0.9013	1.3120
0.9614	0.4736	0.9601	0.4816
0.9862	1.3174	0.9865	1.3026
0.9141	1.1139	0.8930	1.2434
0.9825	0.2795	0.6286	1.2873
0.9646	0.5077	0.9570	0.5594

Since ideal R2 score is 1 and our results are very near to 1 most of the time and also RMSE namely the deviation is small enough we can trust this solution to build an investment strategy on top of it.

Justification

When we compare benchmark and optimized SVR results it is clear that the performance of the SVR method is better than the benchmark in terms of both R2 and RMSE metrics as seen below. There is a significant improvement at the RMSE metric which show the deviation of the solution is declined very much. This predictor can be used for predictions since it has good scores of both R2 and RMSE.

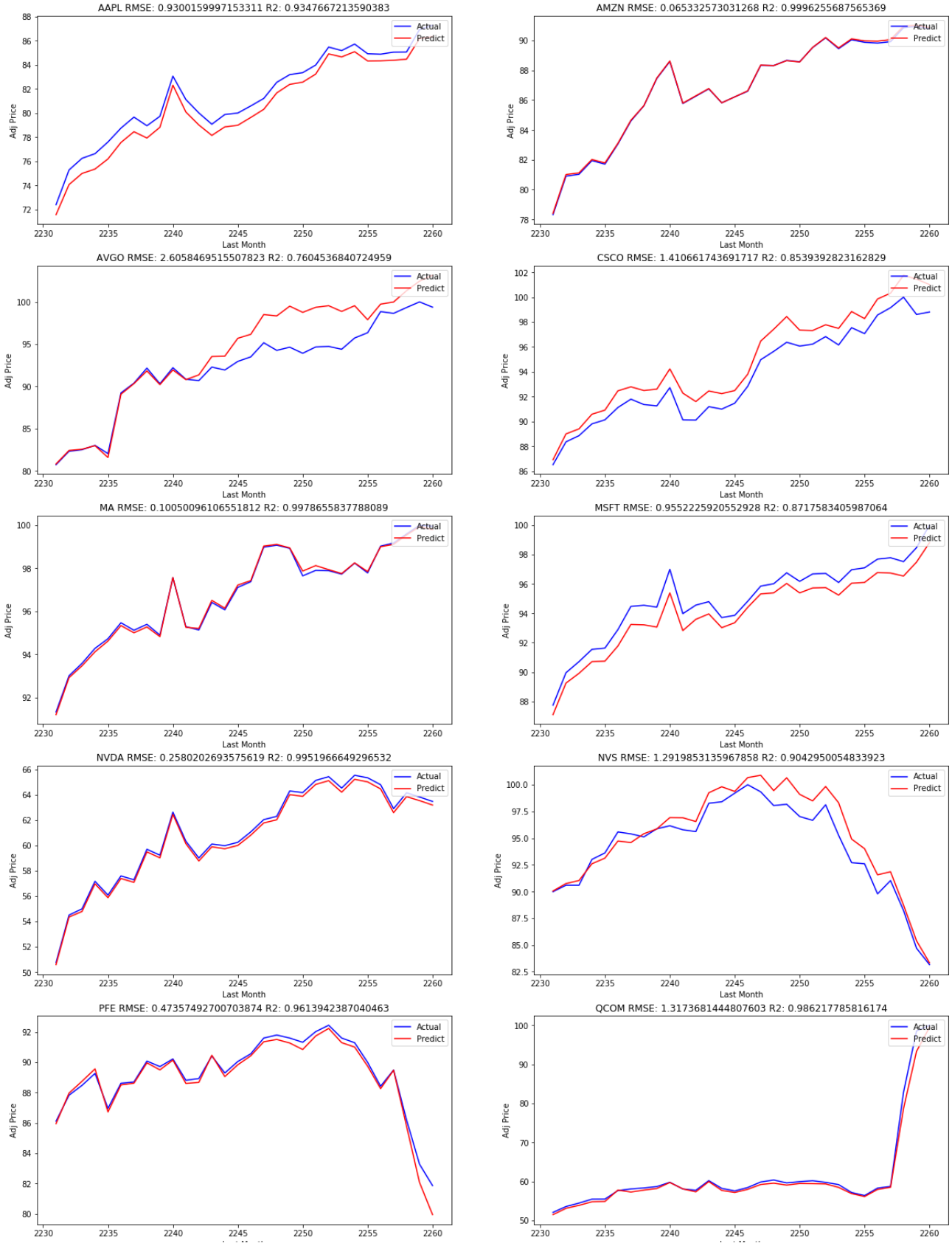
	RMSE	R2
best	0.280099	0.999778

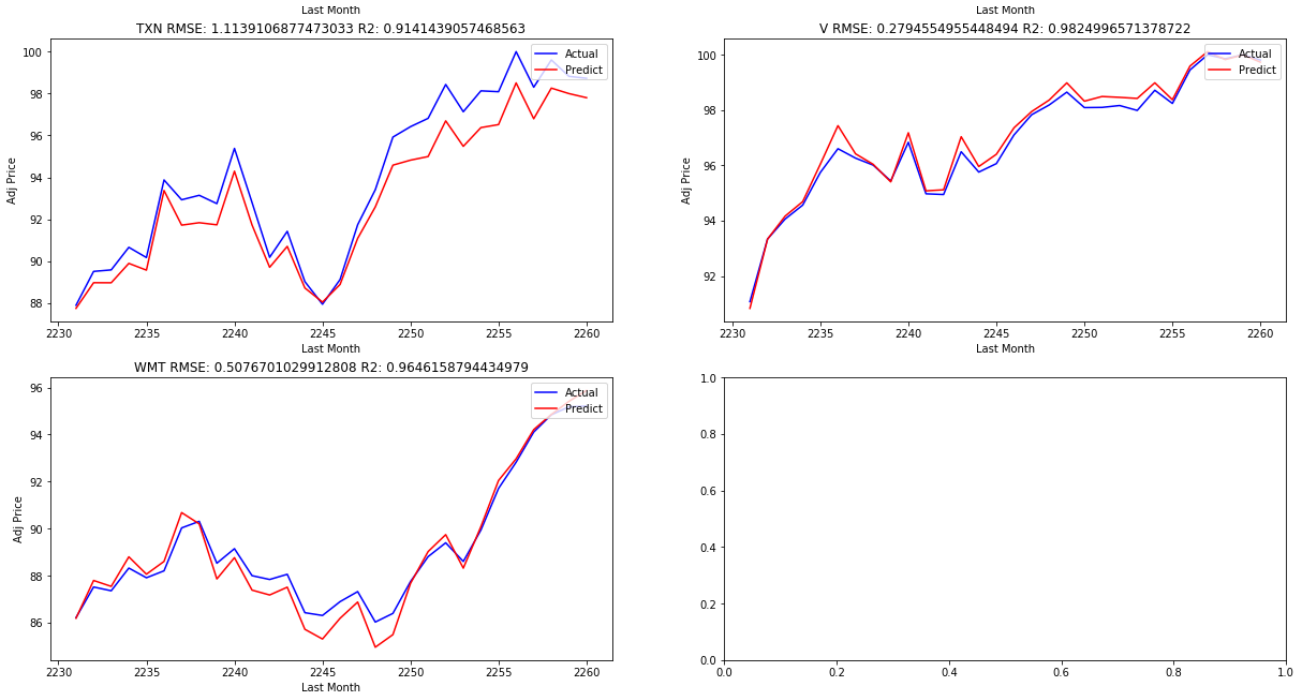
	RMSE	R2
benchmark	0.810680	0.998444

V. Conclusion

Free-Form Visualization

For the visualization I have downloaded last months stock market prices and tried the predictor after training with 9 years of data. The following figure for 12 stocks are plotted for predicted and actual prices. It looks like the predictor is successfull at predicting the price with R2 and RMSE scores. We have R2 score over 0.9 for most of the stocks.





Reflection

To sum up in this project I tried to build a stock price predictor having input of stock market data. I followed the steps below. 1- Gather data 2- Explore and normalize data 3- Create metrics for comparison R2 and RMSE 4- Create benchmark 5- Create solution alternatives 6- Compare solution alternatives 7- Visualize result 8- perturb data to check robustness When compared with the results of similar works, my metric scores was beyond my expectations. I will try to serve this work in the after developing an investment strategy in the future. Intertesting thing to notice was as far as I tried LSTM was poorly performing than SVR. I suppose this is because of non linear information added as column to data like MACD, Bollinger Bands and RSI. In future work I will investigate the effect of these columns.

Improvement

Since I have a full time job I did not have enough time to investigate other methods before the deadline. After having such good results I will try to investigate the performance of other methodologies used frequently to predict timeseries data.

Definitely a stock price depends on not only previous exchange data but also company's financial performance and macro-economical conditions. So in order to have a better long term predictions we need to quantify those information and put into our information space. I suppose after having that done it would really be a good investment engine.

[1] The Effects of Psychology on Individual Investors' Behaviors: Evidence from the Vietnam Stock Exchange http://www.ccsenet.org/journal/index.php/jms/article/download/39897/22142 [2] http://www.investopedia.com/terms/f/financialmodeling.asp [3] A machine learning based stock trading framework using technical and economic analysis http://cs229.stanford.edu/proj2017/final-reports/5234854.pdf [4] Trading volume: What it reveals about the market from https://www.rediff.com/money/special/trading-volume-what-it-reveals-about-the-market/20090703.htm