Technical Analysis using Machine Learning

MLND Capstone Project

Samip Shah

2017

Definition

Project Overview

Basically in this project I would like machine to learn technical behavior analysis of a stock on its own. This is a fairly broad topic to be solved and as part of this capstone project I am going to solve a smaller problem. I got hold of https://www.kaggle.com/dgawlik/nyse dataset. Dataset contains daily trading price (open , close, low, high, volume) of S&P 500 stocks for last five years. The idea is that individual stocks have a daily trading pattern based on its technical features. And we could potentially predict stocks intraday activity beforehand. However, there are many technical features than this dataset provides (for e.g. 50 day moving average, 200 day moving average, etc ..). As part of this capstone I am going to be working with features available in current dataset.

Problem Statement

To elaborate more on the problem, basically using historical trading pattern of one of these companies and their stock prices for that day's close, we can train model to learn relationship between stock behavior and its technical features. Basically I am going to use following strategy to address this problem

1. Use kaggle datasource to retrieve cleaned up data of one company AAPL. The data source is cleaned up and it is adjusted for stock splits which is very valuable to have correct stock prices.
2. Scale data appropriately so that the data is proper for learning. One of the basic pre processing that can be applied is min max scaler (X - Xmin) / (Xmax - Xmin).
3. After proper data pre-processing , I would use features and target variable to train a machine learning model (SVR). Target variable is Close price in this case and features are Open, Low, High, Volume

Metrics

So our task at hand is a regression task and for regression task we have several metrics that we can use - mean squared error, mean absolute error, explained variance score and r2 score.

Main advantage of using r2 score is that it provides a measure of how well future samples are likely to be predicted by the model. So as per the definition of r2 score an insight can be had that the score is trying to capture variance in actual value in the denominator. So it penalizes difference from actual value differently based on variance of the underlying variable we are trying to predict which is good. Mean Squared Error does not capture such nuance it treats error directly without taking underlying variable's variance into calculation. Explained variance also takes variance of the underlying variable into consideration however R^2 score is the default scoring method of the SVR model and it has some difference with explained variance where explained variance deducts mean(error) from the numerator. So it gives higher score to an algorithm which errors only on either side which is not what we want and hence R^2 is perfect choice for us.

In this case we would use R^2 score to evaluate.

1. R^2 score - The coefficient R^2 is defined as (1 - u/v), where u is the regression sum of squares ((y_true - y_pred) ** 2).sum() and v is the residual sum of squares ((y_true - y_true.mean()) ** 2).sum(). Best possible score is 1.0, lower values are worse.

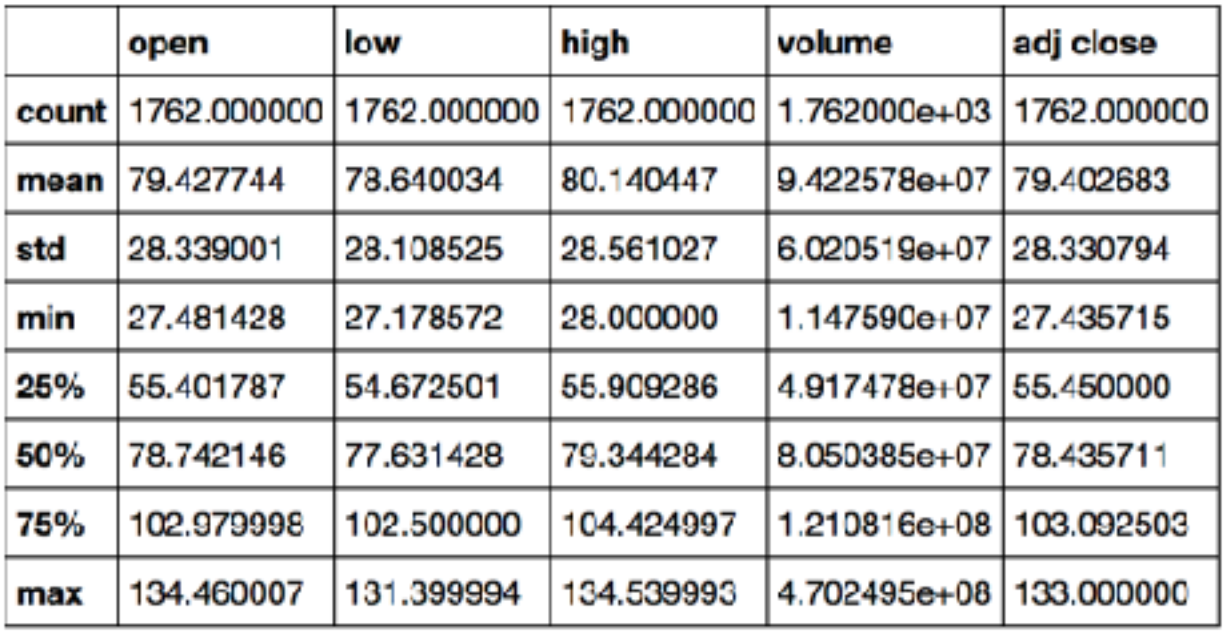
For our case , I expect R^2 score to be more than 0.5 , such model can be improved greatly by using more of technical features.

Analysis

Data Exploration

For this project I am using nyse/prices-split-adjusted.csv datasource. In total it is fairly large dataset with 851264 rows. However since this model has to be trained for every single security separately because technical behavior of each security (For eg. AAPL, GOOG, …) is different and hence we need to train model for each security differently

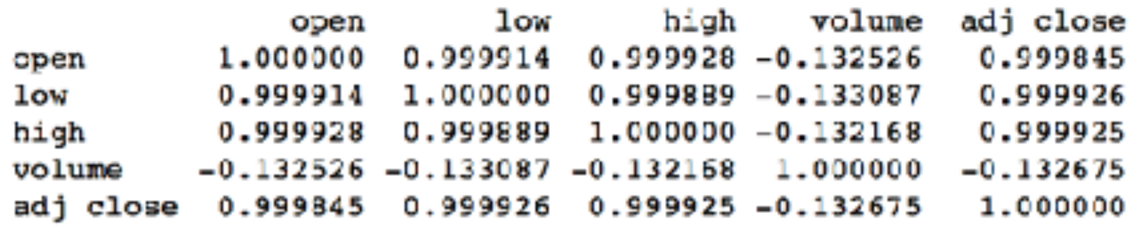
First I extract AAPL security from the data frame and described it using pandas data frame describe function.



	open	low	high	volume	adj close
count	1762.000000	1762.000000	1762.000000	1.762000e+03	1762.000000
mean	79.427744	78.640034	80.140447	9.422578e+07	79.402683
std	28.339001	28.108525	28.561027	6.020519e+07	28.330794
min	27.481428	27.178572	28.000000	1.147590e+07	27.435715
25%	55.401787	54.672501	55.909286	4.917478e+07	55.450000
50%	78.742146	77.631428	79.344284	8.050385e+07	78.435711
75%	102.979998	102.500000	104.424997	1.210816e+08	103.092503
max	134.460007	131.399994	134.539993	4.702495e+08	133.000000

Based on the data above mean and median have pretty much similar values for open, low, high and close (price related columns) and hence data is normally distributed without any skew . However volume median < mean and hence a right skewed distribution.

Moreover diving deeper into correlation matrix , we get more insight

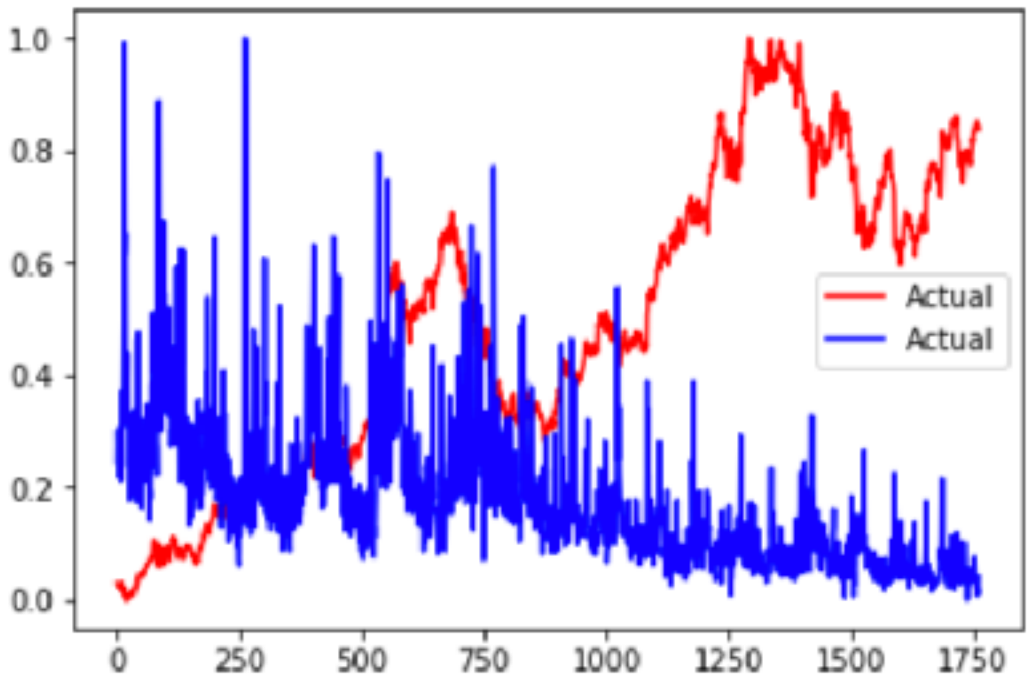


```
               open       low      high    volume  adj close
open       1.000000  0.999914  0.999928 -0.132526   0.999845
low        0.999914  1.000000  0.999889 -0.133087   0.999926
high       0.999928  0.999889  1.000000 -0.132168   0.999925
volume    -0.132526 -0.133087 -0.132168  1.000000  -0.132675
adj close  0.999845  0.999926  0.999925 -0.132675   1.000000
```

As from above we can see ofcourse, open, low, high are positively correlated to close price however volume is negatively correlated to close price. That affirms intuition when stock price is falling volumes are generally high.

Exploratory Visualization

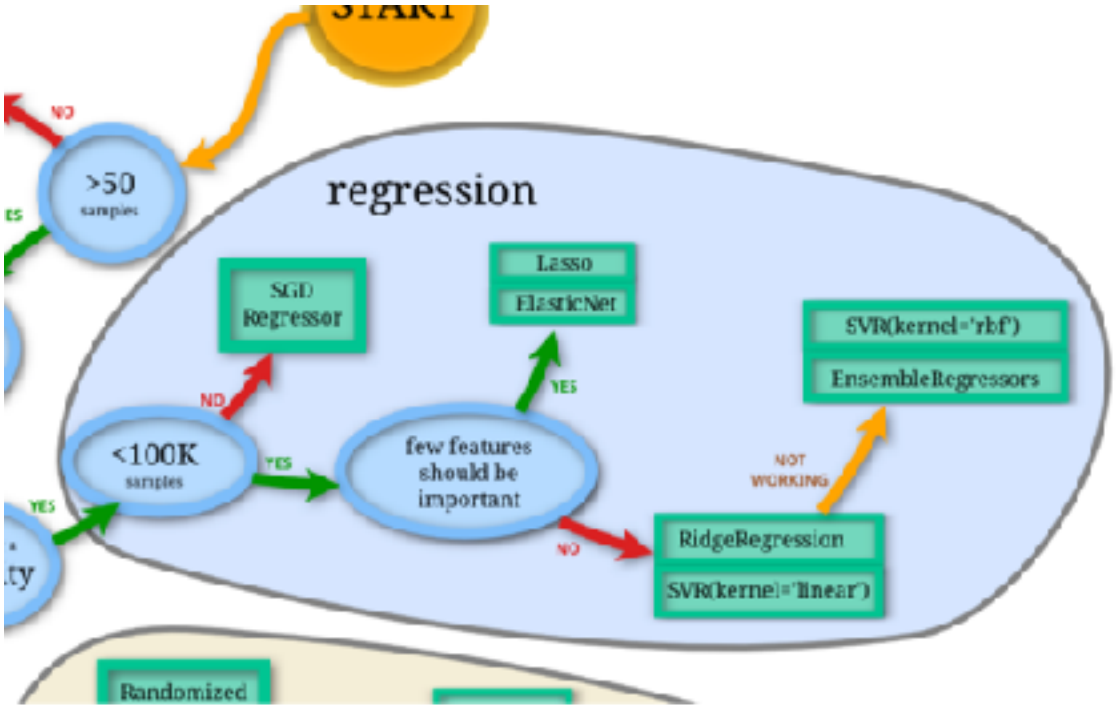
In this section we would try to understand a slightly negatively correlated two columns volume and adjusted close price. Moreover, intuition is such that on a day with higher market movement generally volume tends to be high.



As above image shows volume tended to be higher when the stock was trading at lower price and moreover there are spikes in volumes periodically which correlates with higher price movement in stock to some degree.

Algorithm

To solve this problem I intend on using some of the regression algorithms and one such algorithm I choose is SVR which uses support vector machine for regression task. I intend on using grid search algorithm to find out the best and most optimized parameter for SVR. Based on http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html



I see that in our use case we do not have more than 100k samples which makes SGD Regression to perform not so well as it has stochastic approach to finding regression and for good probability values training sample size needs to be large. Also in current use case we begin with few features and hence it is better to use an algorithm which tries to use all features available. SVR with 'rbf' kernel seems to work well for this particular model.

The way I would use SVR is that I will train it on 90% of the samples and keep 10% of the samples for testing to see whether it performs equally well on test samples or not.

SVR underneath uses Support Vector Machine algorithm. SVMs decision function depends on some subset of the training data, called the support vectors. The model produced by Support Vector Regression depends only on a subset of the training data, because the cost function for building the model ignores any training data close to the model prediction. SVR first of all maps input into a higher dimensional feature space based on the kernel being used. And then it performs a linear regression in the high-dimension feature space using a cost function. This when brought back to the lower dimensional space makes it a non linear. Basic working is map inputs to a higher dimensional space where you could run a linear regression and find the output value that matches the input vector.

Benchmark

Basically in our use case training data R^2 score and testing data R^2 score should be closer to each other not widely different. By focusing and training one security at a time , I would expect these scores to be higher than 0.5 for a given security.

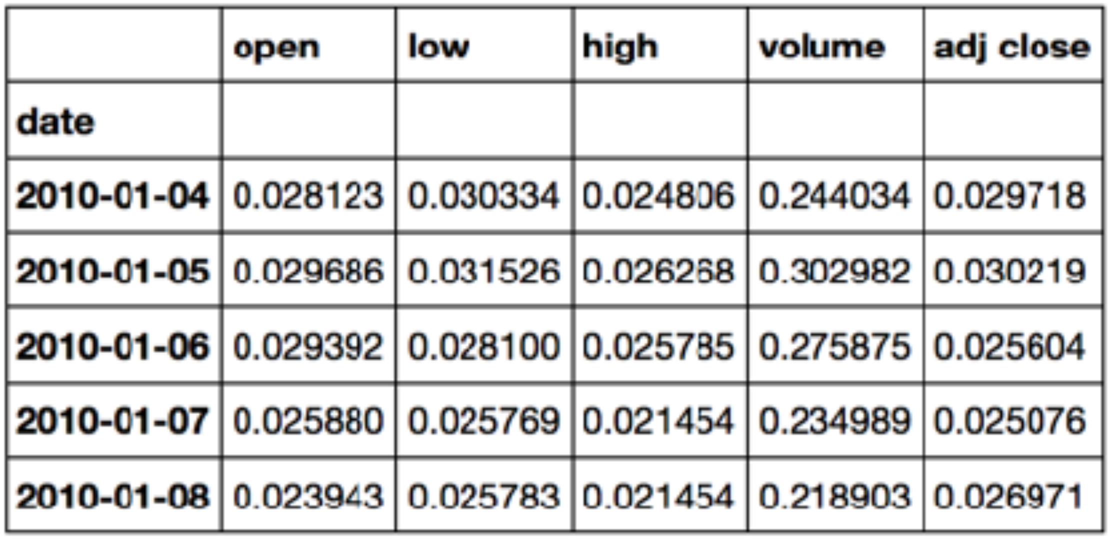
Methodology

Data Preprocessing

I have used min max scalar from sklearn to preprocess every column of the dataset. This way all column data gets converted to 0-1 range keeping earlier skew. However , since in data exploration we discovered our training/testing data does not have a very high skew or data which is not proper and should be discarded. And hence we do not need any more data processing beyond min_max_scalar shift to data tables.

As can be seen in the below table all columns have been converted using following formulas

X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))



	open	low	high	volume	adj close
date					
2010-01-04	0.028123	0.030334	0.024806	0.244034	0.029718
2010-01-05	0.029686	0.031526	0.026268	0.302982	0.030219
2010-01-06	0.029392	0.028100	0.025785	0.275875	0.025604
2010-01-07	0.025880	0.025769	0.021454	0.234989	0.025076
2010-01-08	0.023943	0.025783	0.021454	0.218903	0.026971

Implementation

First collected data 1 sequence at a time and then used training test split of 90/10 to load data. There is also a windowing logic where I wanted to train based on some history of its performance rather than 1 sequence at a time in isolation. However, windowing logic does not seem to working as well because for regression I would have to come up with some proper way of converting multiple y values to a single regression value. I tried using mean of two values however that was not working as well as I would have liked.

Current implementation uses window size of 1 and splits dataset into features, labels and also training and testing datasets.

Model I wanted to use some neural network however I faced issues using python 2.7 with Keras and Tensorflow. I had on my machine python3 setup to use Keras and Tensorflow but due to project requirement of python2.7 I went with models available in sklearn library. As per justification given in analysis section I landed on SVR as my algorithm of choice for tackling this problem.

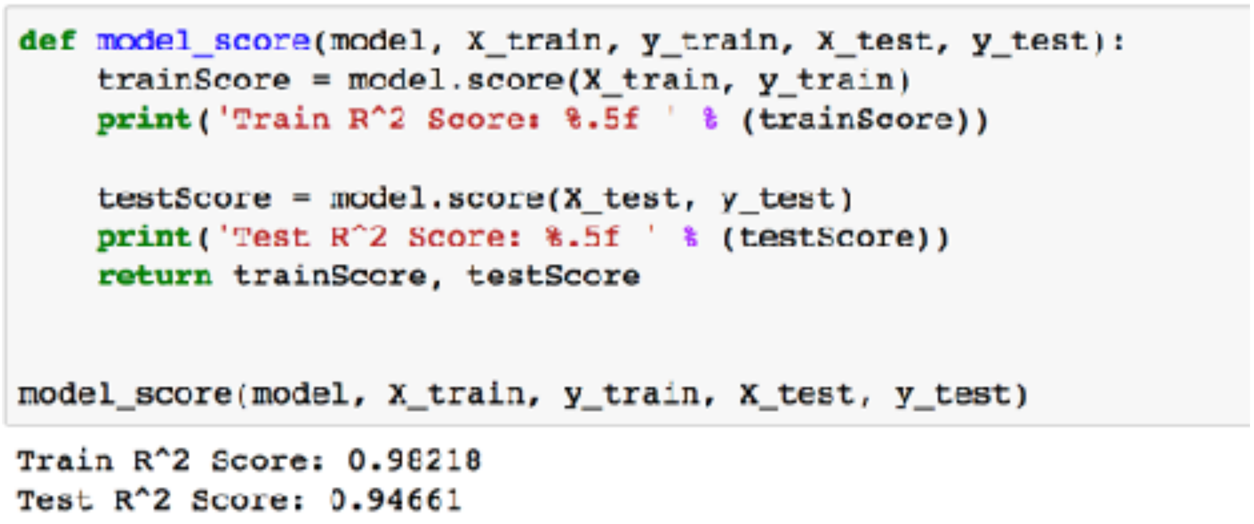
I used GridSearchCV to optimize on kernel among (‘rbf’, ‘poly’) and C (penalty parameter) value between [1,10]. GridSearch algorithm came up with kernel ‘rbf’ and C value of 1 to be best for the algorithm

Created model again with GridSearchCV’s best params and fit the model to training dataset.

Results

Model Evaluation

As noted earlier model’s performance was evaluated based on R^2 score of training and testing datasets.



```
def model_score(model, X_train, y_train, X_test, y_test):
    trainScore = model.score(X_train, y_train)
    print('Train R^2 Score: %.5f ' % (trainScore))

    testScore = model.score(X_test, y_test)
    print('Test R^2 Score: %.5f ' % (testScore))
    return trainScore, testScore


model_score(model, X_train, y_train, X_test, y_test)
```

```
Train R^2 Score: 0.98218
Test R^2 Score: 0.94661
```

R^2 score definition as defined by sklearn -
R^2 (coefficient of determination) regression score function.
Best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of y, disregarding the input features, would get a R^2 score of 0.0.

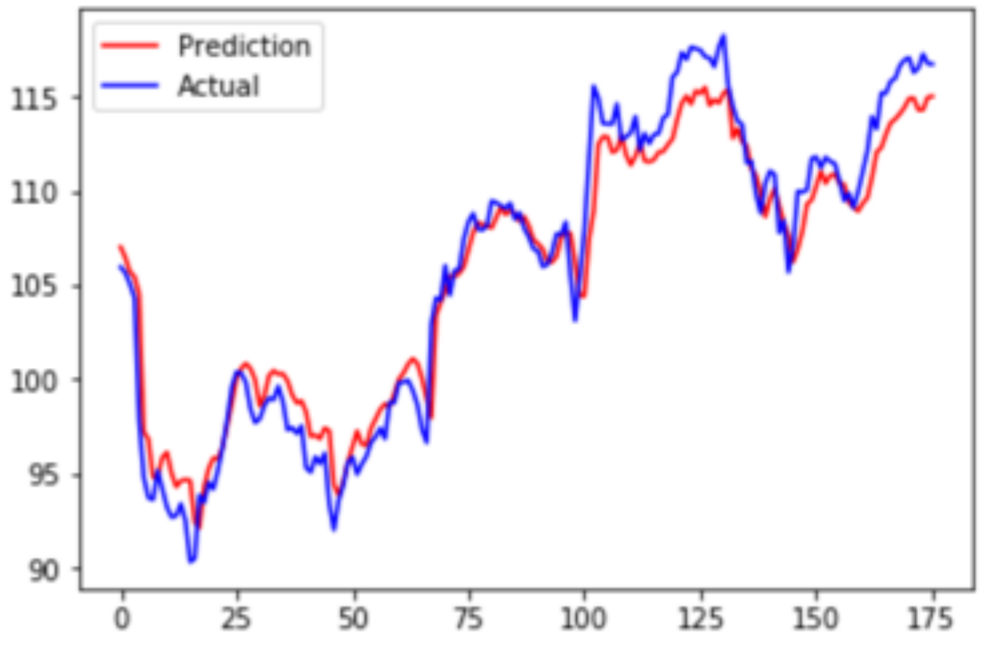
And as can be seen from the result Train and Test score both have value closer to 1.0 which is best possible score for R^2.

Although model’s performance on this AAPL security is fantastic however, I do not think model is robust enough for the problem and can be easily generalized to all other securities. Partly model is not taking historical performance of a security into account and currently it predicts based on a very limited feature set of intraday signals. I could foresee this features only to be incomplete set of features for a robust model.

Result found from this model on AAPL can be trusted as we are getting a really good score on testing as well. However, AAPL's future behavior can change and model needs to change accordingly. I feel in such scenario some sort of reinforcement learning model is very useful for this kind of task. Supervised learning is not totally appropriate for stock market behavior prediction. Model has to be nimble enough to change with changing behavior of a given security.

Conclusion

Visualization



Above is the figure suggesting how close algorithm predicts values to the actual close values.

Solution

So finally End-End solution for finding a relation between stock's close price to its technical analysis based on open,high,low,volume. Basically I ended up using historical daily trading data of AAPL security as features and everyday close price as target variable and trained a Support Vector Regressor model on that. Once trained model is received this model is supposed to encompass AAPL's intraday behavior and should be able to predict close price based on other values.

Improvement

This could be improved much more if the stock price prediction is correlated with its fundamentals. This dataset has fundamentals related information and I also attempted adding those values however, slight unfamiliarity with pandas frames and manipulating them it was getting really difficult to include fundamentals related information in the same data frame due to time constraints. I would like to further this by using fundamental information and also going forward would like to test to see how much can model be generalized and be applied to all the securities. I would like to try using some of the advances in Neural Network to be used with this. Also one major feature in technical analysis is past performance, and since this algorithm here uses only window size of 1 it is not capturing any history of stock movement. Later on history and also several other derived features like 50/200 day moving average should improve this performance.