University of Amsterdam - ML and AI in Finance final project 2019

Predicting SPY price movements and classifying price patterns using methods in Machine Learning

Abhishek Guha

June 9, 2019

1 Introduction

1.1 What is an ETF ?

An Exchange traded fund (ETF) is a group of securities that you can buy or sell via a brokerage company on a stock exchange. ETFs can offer a range of asset classes – from traditional listed company investments to other asset streams like currencies or commodities. Furthermore, ETF structures enable investors to short markets, gain leverage, and avoid short-term capital gains taxes in some jurisdictions. Informally these instruments are basket of different stocks combined with certain weights and a certain cash component. Another interesting term related to ETFs are creation and redemption. You can create ETF by exchanging the basket component with an Issuer (Ishares, Blackrock etc) and redeem them by exchanging ETF for the basket with a certain fee. ETFs trade like stocks, are subject to investment risk, fluctuate in market value and may trade at prices above or below the ETFs net asset value. Brokerage commissions and ETF expenses will reduce returns.

1.2 SPY

In this project we will be mainly look at predicting future price movement of SPDR S&P 500trust, also known as SPY. This ETF trades in NYSE Arca under the ticker SPY. It tracks a market-cap-weighted index of US large cap stocks selected by the S&P Committee. The S&P 500 Index is a float-adjusted, capitalisation weighted index of the top 500 companies in the US market. The Index is designed to provide exposure to the large cap segment of the U.S. equities market and spans over 24 separate industry groups. It captures approximately 75% of the market capitalisation of US equities. In this project I will perform regression tasks (supervised learning [4]) on the data using the following techniques :-

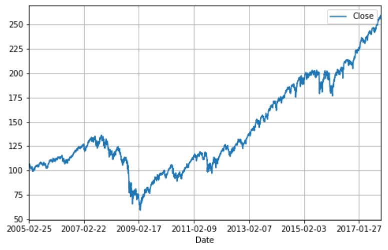

Figure 1: SPY closing price between 2005-2-28 and 2017-11-28.

- *Linear Regression*: Used to set the benchmark.
- *Lasso & Ridge Regression*: l_1 and l_2 norm based regression.
- *Support vector regressor & classifier*: For regressiona and classification task respectively.
- *Random forest*: Ensemble learning using decision trees for my classification task.
- *Gradient boost*: The popular LightGBM classifier would be used for my classification task.
- *LSTM & CNN*: Recurrent and Convolutional neural network for predicting closing price.

2 Data description

I collected daily trading data for SPY trading at NYSE Arca from 2005-02-25 to 2017-11-10. Total number of observations are 3201. This dataset includes each day's open price, close price, highest price, lowest price and trading volume. Additionally the following market indicator data have been included.

Symbol	Description
CME_GC1	CME Gold
CBOE_VX1	CBOE Volatility Index
ICE_DX1	ICE Dollar Index
CME_SP1	CME S&P 500 Index
CME_NG1	CME Natural gas
CME_CL1	CME Crude oil

All the above data are front months rolling future prices. Data is collected from a free online database named Quandl. The symbol in the above table is used to pull data from the website through an API[1]. Data for the specified range has been considered.

3 Feature and target variables

3.1 Target variable

For this project, I have considered the following two target variables.

$$Target_i^1 = (Close_i/Open_i) - 1 \quad (1)$$

where i is the day. This is a regression task.

The next target is the sign of the above target.

$$Target_i^2 = sgn((Close_i/Open_i) - 1) \quad (2)$$

where the definition of sgn is the following :-

$$sgn(x) = \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0 \end{cases} \quad (3)$$

3.2 Feature variables

I have considered two types of feature variables. One solely based on technical indicators and the other based on Market indicators (as in the above table). For technical indicators I have used a python library *ta*. Volume and volatility based technical indicators available in the library have been utilized. For the Market indicators, OHLC and trading volume data have been used.

3.3 Feature analysis and selection

Figure 2 represents the feature importance graph when using market indicators.

Figure 3 represents the feature importance graph when using technical indicators.

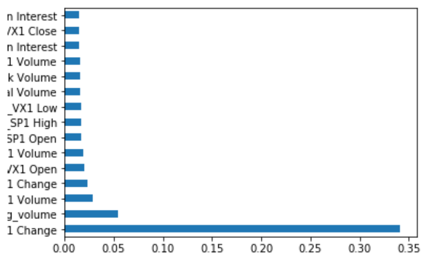

Figure 2: Feature importance when using market indicators.

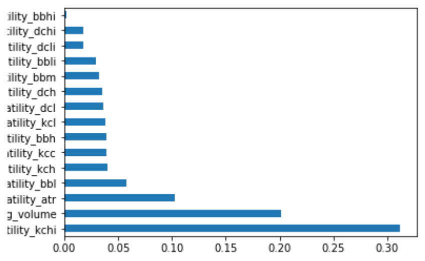

Figure 3: Feature importance when using technical indicators.

Feature importance has been calculated by fitting an `ExtraTreeRegressor` and then extracting the `model.feature_importances_` variable. The intuition of using technical indicator as features is to get the information encoded in the price data of each day. Corresponding to this, use of Market indicators like, Gold prices, is to see to what extent these market indicators are influencing the daily returns of the ETF and also to check the sensitivity. From the above feature importance graph we can see that, changes in VIX is far more important to the daily returns of SPY compared to the changes in S&P500 index itself.

For the technical indicators, the log volume and volatility volume KCHI dominate as the most important features.

4 Exploration of the dataset

I have first checked the daily returns or the $Target^1$ graph for SPY.

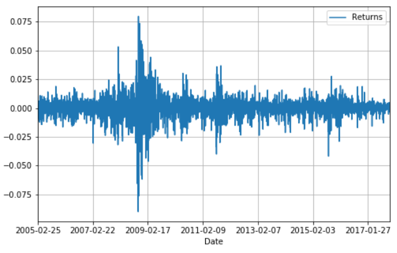

Figure 4: $Target^1$ values for the period.

As expected most variation is during the 2008 crisis.

The volume and log volume being traded for SPY during the period is represented in Figure 5 and 6 respectively.

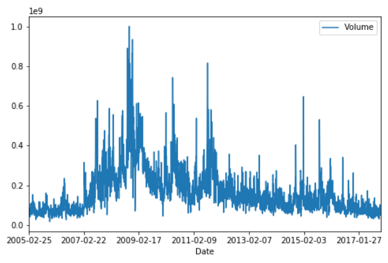

Figure 5: SPY volume for the period.

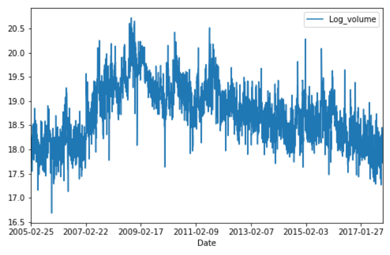

Figure 6: SPY log volume for the period.

5 Algorithm implementation

Now that I have defined my targets and described the features, I can now start with the basic linear models. For every model except the Neural nets, I will try to fit them both on technical indicators and Market indicators as features. My objective is to measure the R^2 score for each of the models and feature combination. I will not be using KFold CV since predicting values for the year 2015 using 2014 and 2016 data isn't logical. Further more, for my LSTM model, I will focus on price prediction rather than predicting the returns. This is because the returns data is extremely noisy and is not a good fit for applying LSTM, since my primary motive is to analyze and predict pricing movements either by using daily returns or directly predicting the price.

I have added two more indirect features outside technical indicators and market indicators. These are log volume (logarithm of the volume) and 1 day lagged $Target$[1]. I can see from the following heatmap no correlation between the target and volume or log volume.

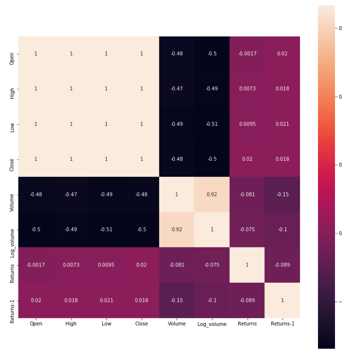

Figure 7: SPY Heatmap for log volume as feature.

80/20 rule has been followed for splitting the data for training and testing datasets.

In case of SVM, rbf kernel has been chosen which is as follows.

$$K(x_1, x_2) = \exp(-\gamma ||x_1 - x_2||^2) \quad (4)$$

where x_1 and x_2 are feature variables.

For the Neural network models (LSTM and CNN) I have implemented a wrapper class around the KERAS.LSTMMODEL and KERAS.CONV1D. It would be better to create a Model framework class using Python's Abstract meta class, which would then be inherited and implemented by child classes.

6 Analysis and discussion

The linear regression models do not behave well but classification (logistic regression) model does. The regression model's score lingers around the range of 0.19 to 0.30, including l_1 and l_2 regularization while classifier's score is around 0.82. Actually from the SPY return graphs I can see that linear classification model would work comparatively better than linear regression model. The regression approach and classification approach both can capture the underlying regularities. The supporting vector regressor does behave better than linear regression models while linear models perform better than support vector machines. One possible reason is that extremely noisy data linear classification models can behave better. Using more data to train each day's model may improve the performance of SVM, but the computational cost will increase significantly since the model is re-trained for each trading day. Another point of consideration is whether the models behave stably over time.

7 Result

Below are the tables of R^2 value for different Regression models using different feature sets.

7.1 Regression task

Here feature space are market and technical indicators.

Feature space	Model	R^2
Market	Linear	-0.03964
Market	Lasso	0.00391
Market	Ridge	-0.02296
Market	SVR	0.40074
Technical	Linear	0.19481
Technical	Lasso	0.33370
Technical	Ridge	0.20317
Technical	SVR	0.02725

It can be clearly seen that SVR is performing better with Technical indicators compared to SVR with Market indicators. This is intuitive as information related to $Target^1$ is encoded in the pricing data more efficiently than in Market indicators.

7.2 Classification task

Following is the table of R^2 with classification task result. $Target^2$.

Feature space	Model	R^2
Market	Logistic	0.69890
Market	SVM	0.58658
Market	RF	0.57878
Market	LightGBM	-0.22483
Technical	Logistic	0.82527
Technical	SVM	0.64898
Technical	RF	0.78939
Technical	LightGBM	0.26130

Here, RF stands for the ensemble learning method called Random Forest.

Classification task perform better compared to regression task. Logistic regression performs the best with Technical indicators and Market indicators. Market indicators also perform better here. This is indicative of the fact that alpha signals can be generated by combining different market indices and future prices.

One interesting observation is the popular LightGBM doesn't perform well enough for both feature space. Further study and model optimization might be required for the model.

7.3 Neural net models

I have performed learning using two models. One with Convolutional neural net (CNN)[2] and another is Long short term memory (LSTM)[3].

7.3.1 CNN

I first started with 50 epoch and 1 layer of 64 filters with 4 kernels. Following is the prediction vs actual graph.

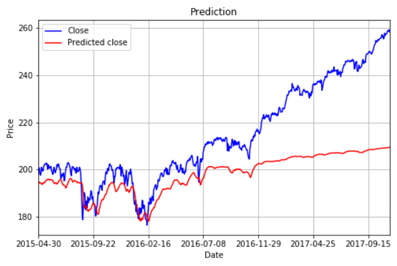

Figure 8: CNN epochs=50, 1 layer

RMSE for the above is 27.57364.

Increasing the epochs to 100 but keeping the same layer.

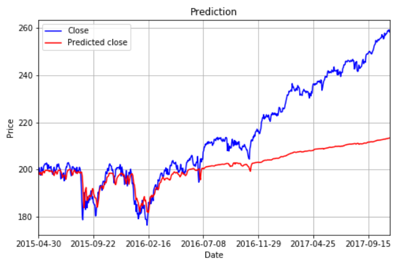

Figure 9: CNN epochs=100, 1 layer

RMS error for the above is 25.58650.

The model predicts the first few portion really well but starts deviating as time increases.

Keeping the same epoch but add another Convulation layer with same filters and kernel produces Figure 10.

RMS error for this is 24.38948.

A decrease is observed in RMSE but still satisfying predictions are not achieved with CNN.

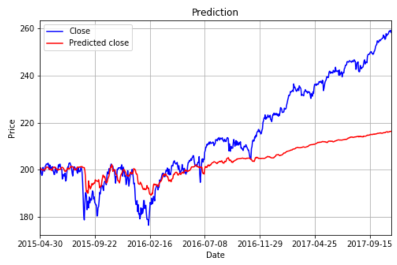

Figure 10: CNN epochs=100, 2 layers

7.3.2 LSTM

LSTM with 4 layers, 50 units in each layers and 3 epochs was created. Look back days was kept to 70 days. Following is the prediction.

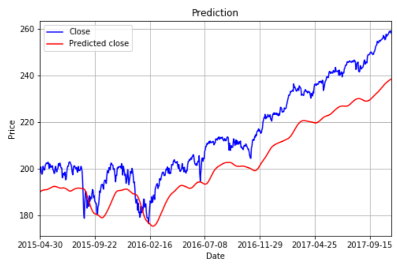

Figure 11: LSTM epochs=3, 3 layers 70 look back days

RMS error for this is 29.40540. The above computation took 39secs for epoch on average.

We can see that LSTM is able to predict the general direction even when time increases. The prices are not correct thus the high RMS error is observed as compared to CNN.

Decreasing the look back days to 60 but keeping every other parameters same I get the following graph.

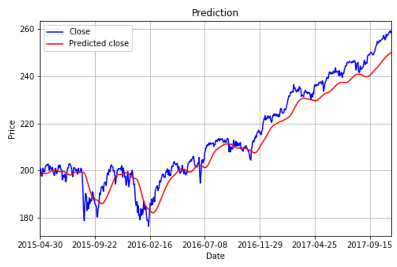

Figure 12: LSTM epochs=3, 3 layers, 60 look back days

RMS error for this is 27.97740.

This shows that increasing the look back days would not necessarily give better predictions.

Increasing the epochs to 20, look back days to 60 and 3 layers I get the following graph.

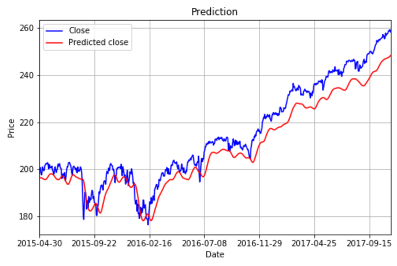

Figure 13: LSTM epochs=20, 3 layers, 60 look back days

RMS error for this is 28.58234.

Our RMSE increased and there isn't any significant betterment in the prediction.

Model loss graph for this setup.

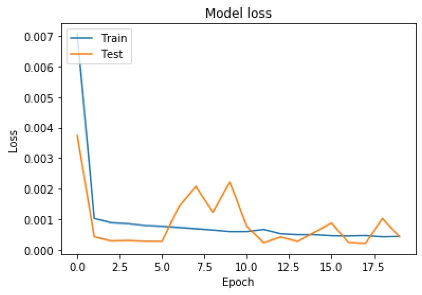

Figure 14: LSTM model loss for epochs=200, 4 layers, 60 look back days

8 Conclusion

Linear regression models for prediction of returns doesn't perform well. Classification problem should be preferred when we want to use Linear models. Predicting price movements with market indicators should be transformed into a classification problem. In case of Neural nets, CNN performs well for a short period but as time increases it deviates as seen in my tests. LSTM on the other hand performs consistently well for predicting the general direction and shape of the price movement. Also CNN is computationally faster than LSTM. Thus when we want to predict price movements for a short term in the future, CNN works better. While for longer running trend LSTM works better though computationally expensive.

9 Future work

- Gathering tick data for discovering intraday pricing patterns using CNN. Intraday spreads, and intraday volumes are great source of discovering hidden signals.
- Optimize LightGBM properly. With the current setup it performs poorly.
- SPY price movements are biased towards a long position holder. Further studies should be done with ETFs which are not baised towards long position holder.
- Use more technical indicators which can uncover hidden patterns. PCA should be performed to give the most important features.
- Macro data like 10 years Treasury bills and yields can be used to dig for pricing patterns for seemingly unrelated to SPY.
- For obtaining a high-performing and resistant to directional bias in time series data, sentiment analysis and data mining for news headlines it would be very beneficial.

All python code used in this project are in this repository.

References

[1] Quandl Python documentation. URL: https://docs.quandl.com/docs/python-time-series.

[2] Keras Convulation Neural Net Documentaton. URL: https://keras.io/layers/convolutional/#conv1d.

[3] Keras LSTM Documentaton. URL: https://keras.io/layers/recurrent/#lstm.

[4] SKLearn supervised learning. URL: https://scikit-learn.org/stable/supervised_learning.html.