Investment Strategies using Support Vector Machines and Support Vector Regression

Ben Johnson, Jesse Smith, Hunter Johnson

***Abstract*- In this project we use SVR and SVM to create indicators for different investment strategies. Overall, we find our models outperform simple stock purchasing strategies and our SVM model predicts price better than the timeseries VAR model for simple stock prices.**

I. Introduction

For our project, our group is evaluating the effectiveness of using Support Vector Machine (SVM), and Support Vector Regression (SVR) to create investment strategies. Our investment strategies are limited to using SVM and SVR model signals to decide optimal times to buy and sell securities, in particular blue-chip stocks such as Amazon (AMZN) and Microsoft (MSFT). Our project is a great example of applied machine learning applied to a field commonly associated with time series modeling. In addition to using SVM and SVR to create investment strategies, we also compare these models to common time series modeling strategies such as Vector Auto Regression (VAR). The two main papers we used for inspiration around the SVM implementation were "Predicting Stock Price Direction using Support Vector Machines" by Saahil Madge [4] and "Stock intelligent investment strategy based on support vector machine parameter optimization algorithm" by Xuetao Li and Yi Sun [5]. In both articles, the authors found that their models were able to outperform the market in certain scenarios, and we went into the project hoping to also find similar results.

III. Methodology

A. *Data Creation*

We pulled data from the S&P500 from February 8th, 2013 to February 7th, 2018. Data is pulled in Python using the YFinance package with features historical and up to date stock data for the United States stock market. To structure the data in section one of documentation, stock prices are used to create the feature variables we will use for training and define the output classification variable. The two feature variables used are the day change from open price to close price and the day change from high price to low price. The output variable is defined as a 'hold' (a one in the data frame) if the current day close is greater than the previous day, otherwise it is a 'no-hold' (a zero in the data frame).

B. *SVM Model Creation*

For training the SVM model, data is split for all dates before a cutoff of 0.8 to use for training. The initial model is trained with default SVM parameters (C value at one) and a radial basis function for the kernel. For MSFT, V, and MA the model performs relatively well with an accuracy of greater than 55% for all examples. This accuracy implies that the model is performing better than a random selection for these stocks, which is the desired objective. However, this is not the case for all stocks, as seen in figure 1**,** the model prediction for CAT accuracy is 44% which is worse than simply guessing the predicted signal.

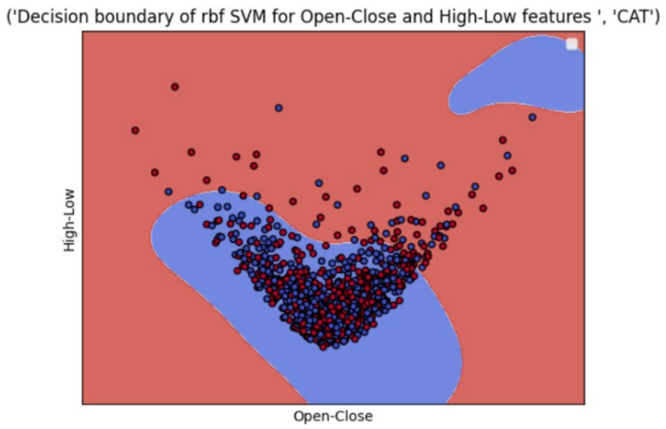


Figure 1- Simple SVM decision boundary for Caterpillar Inc.

In figures 1 and 2, the decision boundary shows the hold (red) and no-hold (blue) data points. The no-hold region in blue shows where the model predicts no-hold data points. For CAT, the model avoids this stock and underperforms as the stock increases due to most data points falling in the no-hold region in blue.

We plot the decision boundary for these stocks to understand what the model is doing with these features. For MSFT, V, and MA most of the data is clustered together mixing the red 'hold' and blue 'no-hold' into one large cluster that the SVM is not able to distinguish. For most of these cases, most stocks in this main cluster are then classified as 'hold'. However, the model in these examples can correctly classify a few outliers and outlying clusters of blue 'no-hold' examples. This ability for the SVM to correctly classify some of these outlier examples is what helps the models perform better than a random selection and increases the accuracy above 50%. For the CAT example, you will see that the model does not do a very good job of identifying any outliers or separating the data in any way. The model classifies most of the stocks as a 'no-hold' which hurts its accuracy performance significantly. In general, this model does seem to perform better when the stock has some outlier examples for the model to identify. Later we will discuss how our model is able to identify these outlier 'no-hold' signals and outperform the market when there is a market downturn.

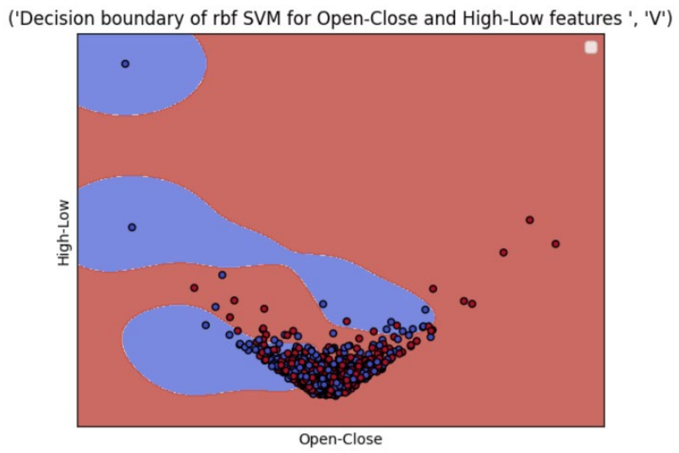


Figure 2 – The decision boundary for Visa. Here you will see that the model successfully classifies many outlier 'no-hold' (in blue) samples. This allows the model to be a bit more conservative than the market and avoid steep market downturns.

We continue to experiment with other kernel functions such as polynomial, linear, and sigmoid to see if any had improved accuracy. We use the AMZN stock to test how changing the kernel function affects the accuracy when compared to the radial basis kernel function. We found that the test accuracy was about the same for each of these, and then when we tried to use the sigmoid function for actual predictions, the model performed significantly worse than the market. When using different regularization parameters to see if adjusting the margins of the SVM would make any difference on this stock, we found that the regularization parameter did not have a significant impact on the accuracy of the model. Experimenting with the gamma parameter also did not have significant performance improvements.

We also experimented with a few other features such as a fourteen-day change in closing price, or a seven-day change in volume. For these features, the model performed better for some stocks and worse for others. Overall, the model accuracy and performance were not improved significantly when experimenting with these additional features, so we left them out for simplicity. We could probably improve the model even further by exploring different features with unique knowledge for various stocks to help make more catered predictions.

C. *SVR Model Creation*

SVR (Support Vector Regression) is a type of SVM used for regression problems opposed to classification tasks. SVR adapts the concept of SVM and uses it for generating predictions with continuous values. This makes SVR useful for understanding non-linear relationships in data by using kernel functions such as the one in this model, RBF (Radial Basis Function). This model works by selecting a bucket of stocks from the dataset then preprocessing and creating features such as previous closing price, 7-day and 14-day moving average, 7-day volatility, and 7-day alpha value. These features were chosen based on their presumed power to capture movement trends in the subset of selected stocks. The model was split into training and testing datasets using an 80-20 ratio. A hyperparameter search is performed using “RandomizedSearchCV’. This is combined with a “TimeSeriesSplit” cross-validation strategy because we are using financial data. Using this approach for our cross-validation strategy was important to preserve the ordering of our financial data, ensuring the model is validated relative to the training set. The best model is chosen from the search, and iteratively trained using the expanding window method. Simply put, this means as the model makes predictions, it is trained on increasing amounts of data. The model then generates trading signals: Buy, Sell, and Hold; it does this based on a threshold of 0.01. If the threshold is not met in either direction, the model maintains a hold. The portfolio is initialized with a value of $100,000 and split equally among the selected stocks. The back test of the model compares executing the trading strategy on the portfolio based on the signals, and simply buying and holding the portfolio over the same period.

D. *VAR Model Creation*

Given the prevalence of time series modeling in the context of stock price prediction, we created a VAR model to serve as baseline and show the improvements made by implementing our SVM and SVR strategies. To craft the VAR model we used the Statsmodel package in python. Our VAR created a predicition system using Amazon and Microsoft stock prices over a 6-year period (2013-2018) and created a forecasted on 2019 data withheld from the model training. For the VAR model the number of lags hyperparameter was tuned using k-fold cross validation, and the optimal number of lags was selected for the final model, as seen in figure 3.

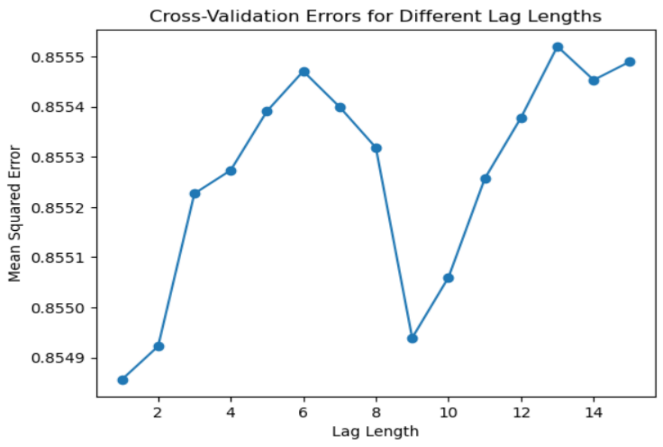


Figure 3- MSE for different Lag lengths in VAR model.

IV. Results

A. *SVM Model Results*

Using the SVM Buy-Hold strategy, our investment strategy can outperform the market in some cases. In figure 4 we use Master Card as an example, where our model is performing slightly better due to avoiding a market dip but then eventually underperforms during a strong market upswing. Over a 9-month period, our investment strategy closely follows the market but misses some large stock increases. In other cases, the model does better by avoiding strong market dips.

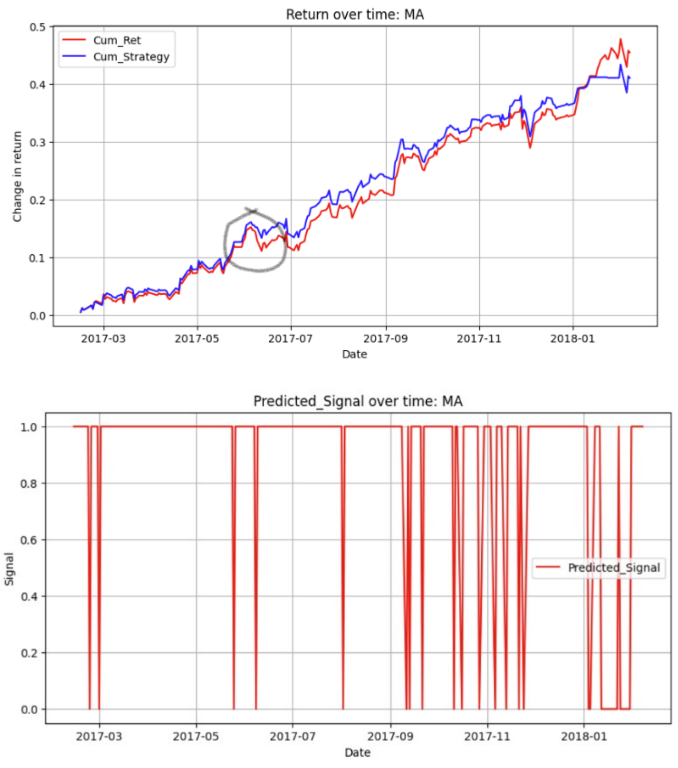


Figure 4- Returns and predicted signals of the tuned SVM model for MA stock price. You will see the market outperforms when it avoids a market dip (circled above).

For cases where the model does not perform well compared to the market price, it is helpful to look at the data the model was trained on. For example, for CAT we can see that the model was trained on data that was mostly in a downward trend from 2013 to 2016, and the model acts extremely conservatively and does not predict ‘holds’ for the period where the CAT stock price grows rapidly after 2017. In general, you will see that our model often performs better when avoiding downturns, but the model often misses large upswings in the market, which causes it to underperform particularly in situations where a stock was in decline in the training period but then has a rapidly increasing test period.

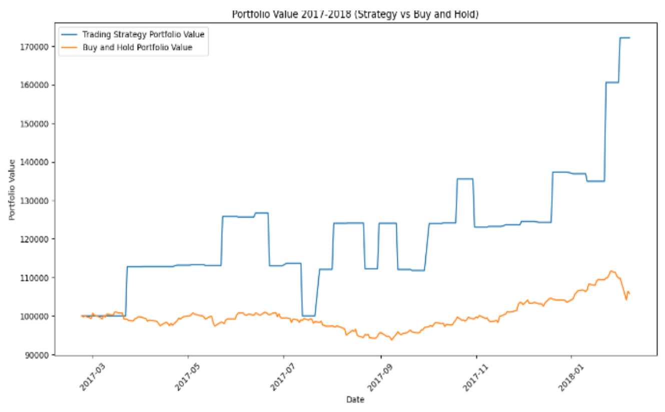
We also used K-fold time-series cross-validation to see how the model performs on different splits. Time-series cross-validation is like regular cross validation except that it maintains the date ordering so that the training date stops one day before the test data starts. By taking the average accuracy for all the stocks in this test, we found the model still performed slightly better than random, with a 52% accuracy. We find similarly promising results for the F1-score at an average of 56%, the precision at an average of 53%, and recall at an average of 67%. All of this suggests that the model is indeed able to identify some trends and correctly classify them, it is just missing important upward buy trends which is causing it to underperform the market in some cases. These results from cross-validation also help check the model for over-fitting, which we did not see any significant evidence of.

We also wanted to find more evidence for whether the model is performing better in avoiding downward market trends. We tried this with both a cross-validation approach (summing the cumulative returns) and a randomized split approach. For both approaches, we find that when there is a negative market return day, the model usually outperforms the market. For the experiment we ran, we randomized the split point in the data and compared the next day returns with our strategy compared to the market.

When we ran the experiment and calculated the number of negative market return days, we found that our model outperformed the market on forty of the forty-two downturns. This further suggests that our model performs well in avoiding downturn trends and can be used to help investors make decisions during market dips. This compliments the findings in mentioned before where we can see that the model is able to separate the 'hold' and 'no-hold' regions mostly around outliers of 'no-hold' data, suggesting the model can be more conservative than the market and help identify 'no-hold' days.

B. *SVR Model Results*

Now, let us discuss how our SVR model performed using the trading strategy relative to buying and holding from 2017-2018. As seen in figure 5, the Trading Strategy Portfolio is shown in blue, and the Buy and Hold Portfolio is shown in orange. The strategy assumes trades are made in regular intervals, every 7 days to avoid high transaction fees and market noise. The trading strategy portfolio outperformed the buy and hold strategy by a significant margin. The final value of the trading strategy portfolio is around $172,175 whereas the buy and hold strategy portfolio ended with $106,836. This indicates that the SVR trading strategy was able to capture the market's fluctuations and provided a higher return than the classical buy and hold approach. The large difference in final portfolio value reiterates the potential for algorithmic trading using machine learning techniques.



```
Final Trading Strategy Portfolio Value: 172175.30000000002
Final Buy and Hold Portfolio Value: 105836.92876707061
```

Figure 5- SVR Investment results

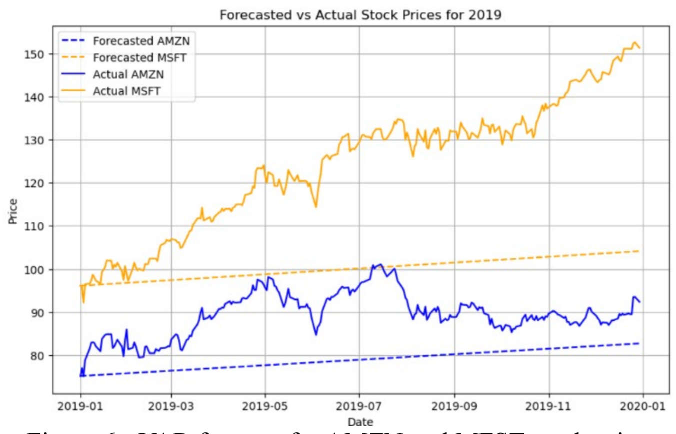


Figure 6 - VAR forecast for AMZN and MFST stock prices compared to actual prices in 2019.

C. *VAR Model Results*

The VAR model results seen in Figure 6 show the lackluster results given by the workhorse time series model and the significance of our SVM and SVR models, staple models of the machine learning field. The VAR model tends to underpredict stock prices and fails to capture any erratic day to day price movement, rather attempting to predict average price of the course of 2019.

V. Conclusion

Overall, our study on the effectiveness of SVR and SVM for investment strategies produced promising results. For the SVM model, the ability to predict a 'no-hold' day outlier allows the model to perform better in market downturns. We showed that our model outperforms the market during most single-day downturns, though it underperforms in market upswings. Throughout this study we concluded that with some limitations, SVM and SVR have potential for robust and effective investment strategies. Further testing and validation are needed before this could be applied to real-time investment strategies. Some features we think that would help these models be applicable in active market trading include real-time news/ information regarding the market and an API that pulls real-time financial data such as that offered by Yahoo Finance. Upon including these features along with others, we think that these models have the ability to be used in active trading. Our study underscores the potential for machine learning techniques in day-to-day algorithmic trading and solidifies the necessity for further adoption and analysis.

References

[1] All the code for our project can be found here: flpymonkey/SVM-Stocks (github.com)

[2] Predicting Stock Price Direction using Support Vector Machines - GeeksforGeeks

[3] S&P 500 || Time Series Forecasting with Prophet (kaggle.com)

[4] https://medium.com/@myselfaman12345/c-and-gamma-in-svm-e6cee48626be

[5] "Predicting Stock Price Direction using Support Vector Machines" by Saahil Madge

[6] "Stock intelligent investment strategy based on support vector machine parameter optimization algorithm" by Xuetao Li and Yi Sun

[7] Fine-tuned support vector regression model for stock predictions | Neural Computing and Applications (springer.com)

[8] Predicting Stock Prices with SVR. In this post, I'll go over how I used… | by Joseph Hart | Medium

[9] Walking through Support Vector Regression and LSTMs with stock price prediction | by Drew Scatterday | Towards Data Science