Comparison of Different Algorithmic Trading Strategies on Tesla Stock Price

Tawfiq Jawhar, *McGill University, Montreal, Canada*
tawfiq.jawhar@mail.mcgill.ca

***Abstract*—Algorithmic trading is the process of automating the buying and selling of stock shares in order to increase the profitability of investment in the stock market. Two classes of strategies are compared in this project on the Tesla stock price: moving average strategies and price movement classification strategies. The moving average strategies are optimized using Bayesian optimization and modified with a risk manager that uses Gaussian Mixture models. For the classification strategies, KNN, SVM, and GP classifiers are explored and tested. The risk manager using GMMs improved the result and risk of the moving average strategies. SVM classifier strategy performs the best. The result shows that different strategies performed better on different period of times. Which shows that for such problem with a high volatility signal a good approach would be a combination of strategies each used on a specific trend or context in a reinforcement learning environment.**

***Index Terms*—Algorithmic Trading, Machine Learning,**

I. Introduction

THE investment in stock market is a common way of investing money. People buy equity shares of a company and hold them (buy and hold strategy) in order to profit from the long term up trends in the market; when the price of the company's shares increase. Trading is the process of buying and selling shares based on analysis of stock price to outperform the profit return of the buy and hold strategy. To maximize the profit of an investment, a trader needs to buy shares at a minimum optima and sell shares at a maximum optima of the company's price signal over time. Algorithmic trading is the process of using an algorithm to do the trading automatically. If an algorithm can predict exactly when the price will go up or down then buying and selling can become easy. However, predicting the price movement of a stock is not easy. Especially with the risk it involves of losing money if a prediction is not accurate. In this project I use Gaussian Mixture models to detect risky regimes at days where the price is changing at a higher rate than usual days. Then I compare 2 classes of algorithmic strategies; Moving Average and Movement Classification strategies. The next section explains the trading environment used and assumptions made.

II. Trading Environment

The trading environment will be used for back-testing is Zipline[1] which is an open source Python algorithmic trading library. Zipline is used as the back-testing and live-trading engine powering Quantopian investment firm[2].

[1]Zipline Github Repo: https://github.com/quantopian/zipline.
[2]Quantopian: https://www.quantopian.com/

To simplify the environment the following assumptions are made:

- The trading environment has no slippage. Slippage happens when the price of the trade and the price for when the trade is actually executed are different.
- The trading environment assumes no commission or cost when making orders. Which is not accurate in real life, but this is considered for simplification.
- All the strategies explored can do 3 actions only, Buy shares with the full money available, Sell all the shares, and Do Nothing. When the strategy is already invested (shares bought before) and a Buy action occurs, nothing will be executed. Similarly when a strategy is not invested and a Sell action occurs, nothing will be executed.
- The strategies will be buying and selling shares from one stock only.
- The strategies will be making a decision on a daily basis after market opens.
- The training set is the Tesla stock price from 2014-1-1 to 2015-12-31 (2 years).
- The testing set is the Tesla stock price from 2016-1-1 to 2017-12-31 (2 years).
- The capital money each back-test will start with is USD 100,000.

Looking at the return on investment at the end of the trading back-testing is not enough to decide whether a strategy is doing well or not. If a strategy has a high volatility (standard deviation of the return over the period of trading) then even if the return is high at the end, the strategy has a high risk factor. A good strategy should have a good return on investment with low risk. The Sharpe Ratio [1] is a popular performance measure used in algorithmic trading. Sharpe Ratio calculates the risk-adjusted return as follows:

$$SharpeRatio = \frac{R_p - R_f}{\sigma_p} \tag{1}$$

where:
R_p is the expected portfolio return
R_f is the risk free rate
and σ_p is the portfolio standard deviation

Zipline uses free risk rates from the US treasuries data repository. The Tesla stock price (TSLA) is an interesting stock to apply algorithmic trading on because of its high volatility. When a price has rapid and large up and down movement the possible profitability is higher. However, the risk also becomes higher, which makes it more difficult to

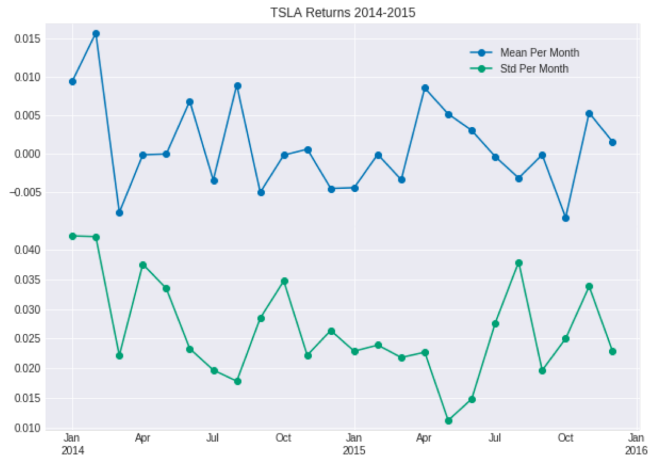


Fig. 1: Mean and Standard Deviation of daily logarithmic returns per month of the Tesla stock price between 2014-1-1 and 2015-12-31. The difference in the mean and standard deviation from month to month represents the non-stationary of the price signal.

apply algorithmic trading on. Stock price can be seen as a non-stationary (different mean and variance over different period of times) time series. Figure 1 shows the mean and variance of the monthly logarithmic return of TSLA on the training time period which represent the non-stationary of the signal. Logarithmic return is usually used in finance because it is symmetrical. For example, if an investment of $100 that has a logarithmic return of +50% then -50%, the return value will be $100.

$$LogarithmicReturn = \log\left(\frac{V_{t+1}}{V_t}\right) \tag{2}$$

where V_x is the value at time x.

Although the price signal is a time series, however, in many cases through out this project the daily prices or the daily returns are assumed to be independent and identically distributed.

III. Detecting Risky Regime Using GMM

A common assumption that is usually made that the movement of the price at a period of time follows a Gaussian distribution. To test this hypothesis Kolmogorov-Smirnov (KS) [2] test is applied on the Tesla logarithmic returns. Two-sided KS test is a non-parametric goodness of fit test that compares the CDF of the logarithmic returns of Tesla price with a CDF of samples coming from a Normal distribution with the same mean and variance of our returns. The test is done over the 2 years of training period and each year separately. The test provides D-value and p-value. D-value represent the maximum distance between the two CDFs, the closer D-value is to 0 the more likely the two samples are coming from the same distribution. The p-value represents the confidence of evidence behind the null hypothesis. In this case the null hypothesis is that the logarithmic returns for Tesla follow a Normal Distribution. In order to reject the null hypothesis p-value should be less than the critical value of 0.05. Figure 2 shows the result of the test with a p-value of 0.0589 over the two year period of 2014 and 2015, 0.1346 over the period of 2014 and 0.1558 over the period of 2015. In all 3 cases the null hypothesis could not be rejected. However, both years separately gave a higher p-value than the ks-test over the 2 years that resulted in a p-value close to 0.05. Based on this we assume that any decision based on the distribution of logarithmic return following a Normal distribution will consider at most the past 1 year. Although the KS-test did not reject the null hypothesis, we use Gaussian Mixture Models to detect outliers in the movement of the price. An outlier is a return higher than the usual daily returns. When an outlier occurs, the trading becomes riskier as the price is moving in an abnormal way. Using $sklearn.mixture.GaussianMixture$

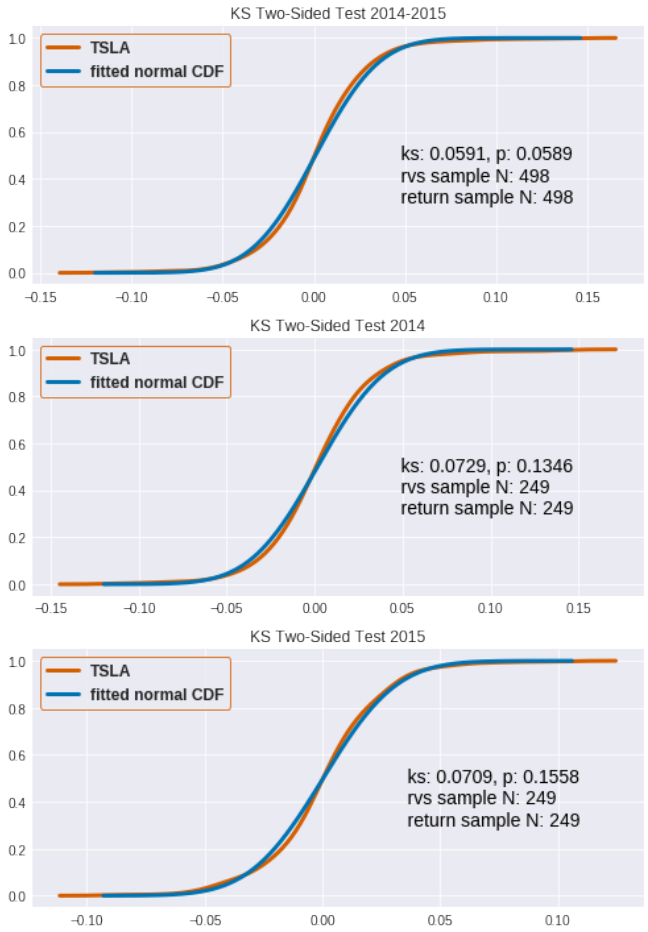


Fig. 2: Two-sided Kolmogorov-Smirnov (KS) test on the logarithmic return of Tesla on different periods of time compared to a Normal Distribution.

module the logarithmic return data is fit on GMMs with different number of components and different covariance matrices; spherical, diagonal, full and tied. To compare the different parameters, Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC) is used to determine the best number of components to use. AIC and BIC adds a penalty to the likelihood as numbers of components increase. BIC has a higher penalty than AIC for having more parameters in the model. Which can be a better measure to not over-fit.

Figure 3 shows the results. The parameters best fit the data are 2 number of components with a spherical covariance matrix.

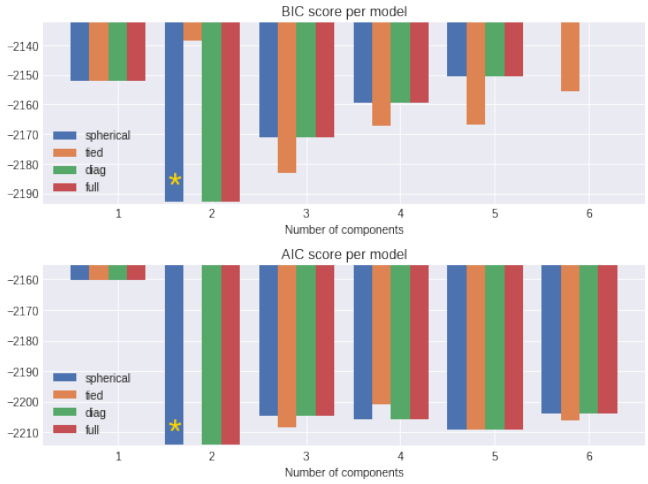


Fig. 3: Gaussian Mixture model parameter selection. For each set of parameters (number of components and covariance matrix) the GMM model is fit on the logarithmic return of the training data of years 2014 and 2015. The results shows that the best parameters to use is with 2 components and a spherical covariance matrix.

The GMM model GMM_{test} is fit on the training data of 2014-2015 and is then used to predict the testing set of 2016-2017. In order to compare the accuracy of prediction, a GMM model GMM_{true} is fit on the testing set and used to predict the testing set as well. The prediction of this model is assumed to be the true labels of the data which is compared to the prediction of GMM_{test}. To compare the prediction of both models, the adjusted mutual information (AMI) score [3] is used. AMI measures the information shared between the clusters in both models. AMI score of 1 means for each cluster predicted in GMM_{test} there exist a cluster in the GMM_{true} predictions that has the same data or information. The result shows (Figure 8 in Appendix A) an AMI score of 0.4 which strengthen our assumption that any decision based on the distribution of the logarithmic return needs to be made with at most 1 past year. In other words, to predict whether the current day logarithmic return belongs to the risky regime, the GMM model needs to be fit on the past 1 year of data every time. In the next section, GMM is used as a risk manager in the moving average strategies.

IV. Algorithmic Trading Strategies

A. Buy and Hold

Buy and Hold strategy is used as a benchmark. At the beginning of the trading period, shares from the Tesla market at the value of the available capital money is bought. The shares are held until the end of the trading period. The portfolio value at the beginning of the trading is equal to the capital money. At the end of the trading the portfolio value compared to the capital money represents how much profit is made in this investment.

B. Simple Moving Average - SMA

Moving Average strategies are simple and well used in algorithmic trading. SMA(N) is the N-day simple moving average which is the mean of price for the past N days. In our trading environment, the trading is happening every day at market opening time. SMA will use the closing prices for the days before and the open price for the day of trading. SMA is calculated:

$$SMA(N) = \frac{\sum_{i=t-N}^{t-1} ClosingPrice_i + OpeningPrice_t}{N} \tag{3}$$

The SMA strategy is simple. $SMA(N)$ and $SMA(M)$ are calculated, where $M > N$. Then the two SMA values are compared. If the short window SMA has a higher value than the long window SMA then the price is going up and vice versa. The strategy is as follow:

```
IF SMA_short > SMA_long THEN
    IF NOT Invested THEN
        BUY
ELSE
    IF Invested THEN
        SELL
```

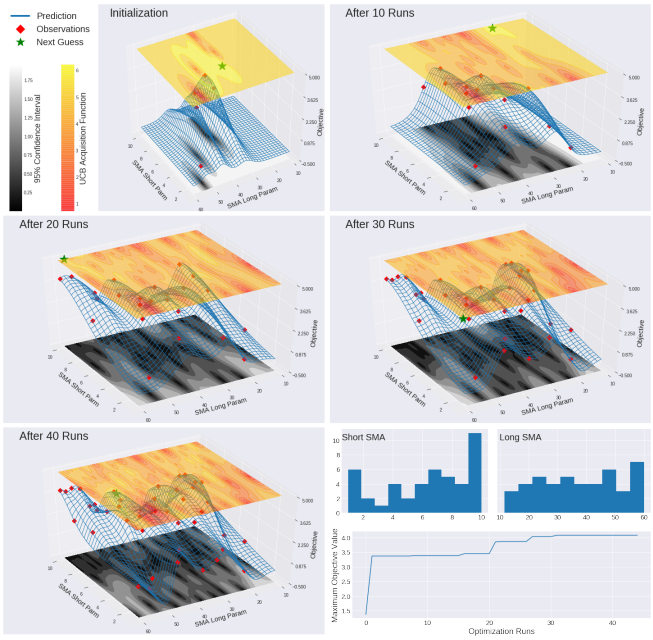


Fig. 4: SMA optimization using Bayesian optimization with GP-UCB acquisition function represented at different stages through the optimization. Plots on the bottom right represent histograms of the parameters selected and the maximum objective value through out the optimization steps.

However, choosing short and long windows (M and N) can be difficult. The strategy is optimized using Bayesian optimization with a GP-UCB acquisition function [4]. When the optimization was tested with an objective value of the Sharpe ratio at the end of the back-testing of the training period, the optimization showed signs of over-fitting where the GP predicted surface was not smooth with multiple high

peeks. To ensure that the optimization is not over-fitting on the training period of time, the objective function runs the back-testing on 4 different time periods (6 months each) of the training period time of 2014 and 2015. And the objective value is the sum of the 4 Sharpe ratios for every time period. The limits specified for SMA_{short} and SMA_{long} are $[1, 10]$ and $[11, 60]$ respectively. 5 random points are used for initialization and the optimization then runs for 40 iterations. The values of SMAs are rounded to 0 decimal places when passed to the objective function for testing. The best values for short and long SMA parameters at the end of the optimization are 7 and 22 with an objective value of 4.08. Figure 4 shows the optimization results at different iteration steps.

C. Simple and Exponential Moving Average Strategy - SEMA

SEMA strategy uses both Simple Moving Average (SMA) and Exponential Moving Average (EMA)[5]. The exponential moving average is similar to the simple moving average but it gives a higher weight for the most recent prices depending on the number of periods in the moving average.

EMA for period t is calculated as follow:

$$EMA_t = \begin{cases} Y_1 & t = 1 \\ \alpha \cdot Y_t + (1 - \alpha) \cdot EMA_{t-1} & t > 1 \end{cases} \quad (4)$$

Where Y_t is the price value at time period t. EMA is calculated using TA-lib.[3]

The SEMA strategy is similar to SMA strategy. SMA_{short}, SMA_{long}, EMA_{short}, and EMA_{long} are calculated and the short window moving averages are compared to the long window moving averages. If both SMA and EMA makes a decision of buying then the strategy will buy. If any of the moving averages indicate selling then the strategy will sell. The strategy is as follow:

```
IF SMA_short > SMA_long AND
EMA_short > EMA_long THEN
    IF NOT Invested THEN
        BUY
ELSE IF SMA_long > SMA_short OR
EMA_long > EMA_long THEN
    IF Invested THEN
        SELL
```

Similar to SMA strategy, Bayesian Optimization is used to optimize for the short and long parameters for SMA and EMA. The same objective function is used as before. The ranges of the parameters are set as follow: SMA short [1,10], EMA short [2,10] and long SMA and EMA [11,60]. The optimization is initialized with 10 random points and after 500 optimization iterations the best parameters found are EMA long: 60, EMA short: 10, SMA long: 17 and SMA short: 9 with an objective value of 5.94. Figure 5 shows the different parameter combinations that were explored and the maximum objective value of the optimization over the 500 iterations. The EMA parameters are at the upper limit of the range specified, which indicates that the optimization needs to be repeated with larger upper limits for EMA parameters. However, due to the time constraint the optimization was not repeated.

[3]TA-Lib: Technical Analysis Library https://www.ta-lib.org.

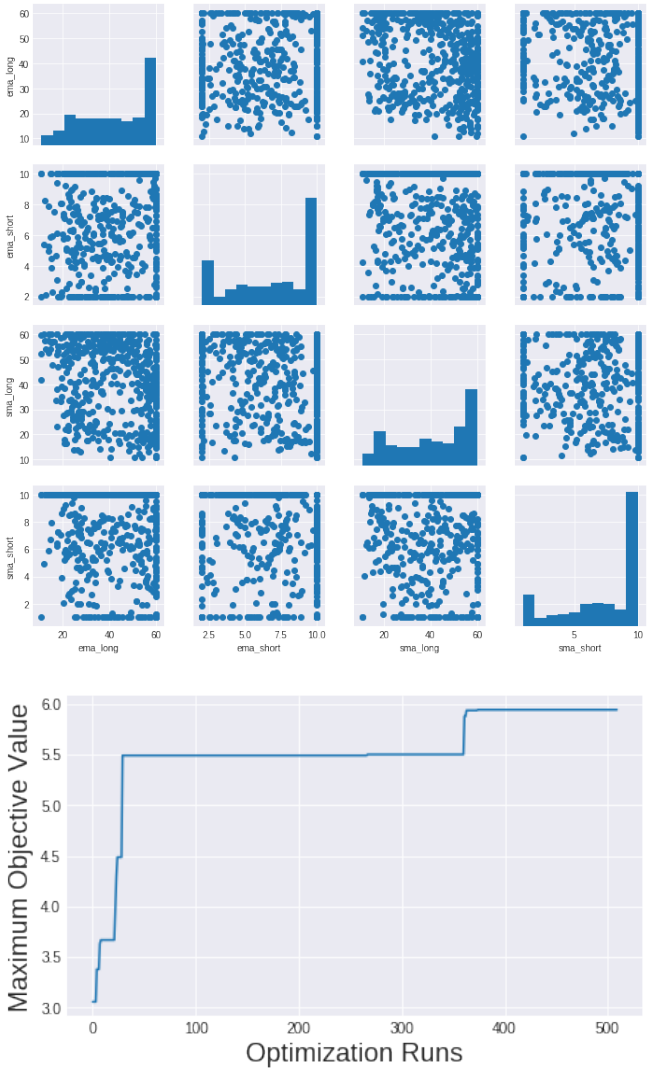


Fig. 5: Top plots represent the different combinations of parameters that were tested in SEMA optimization using Bayesian optimization with GP-UCB acquisition function and the bottom plot represents the maximum objective value through out the 500 optimization steps.

D. Moving Average with GMM Risk Manager

In section III GMM is used to cluster the days based on the logarithmic return to detect a risky regime. If the trading belongs to the cluster with higher volatility then the trading on that day is considered risky. MA with GMM strategy combines the moving average strategies (SMA and SEMA) with GMM. GMM is used as a risk manager. The strategy is as follows:

```
Fit the GMM on the past 255 days
Predict the cluster of the opening price
logarithmic return of the trading day
IF the prediction is in high volatility
cluster THEN
    RISK = TRUE
```

```
ELSE RISK = FALSE
IF NOT Invested AND MA is Buy and Not
RISK THEN
    BUY
ELSE IF Invested AND MA is Sell THEN
    SELL
ELSE IF Invested AND RISK AND Return of
trading day is negative THEN
    SELL
```

MA is either SMA or SEMA strategies. The strategy does not allow buying shares at a risky detected day. It also forces a sell of all invested shares if a risky day is detected and the return on that day is negative. This strategy is intended to detect the small outliers during the trading and use a safe approach to deal with those outliers. The parameters for SMA and SEMA are the same parameters used in B. and C.

E. Price Movement Classification

This strategy uses a classifier to predict the movement of the price at t+1. The two classes are Up or Down which represent whether the price the next day went up or down. The classifier takes an input a feature vector representing the day to day percentage change of the past 30 days. The output is +1 or -1 for up and down respectively. Three classifiers are tested: K-Nearest Neighbour, Support Vector Machine and Gaussian Process. For each classifier the parameters are tuned using 5 fold cross validation on the training set of 2014 and 2015. The data is balanced with 235 Down labels and 232 Up labels. The folds mean accuracy of prediction is used to compare the different models. Figure 6 shows the results of the tuning of the different parameters.

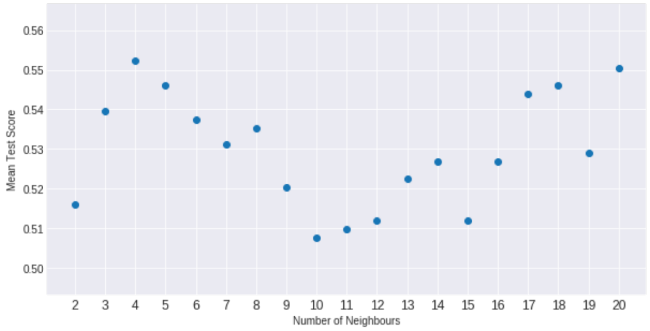
1) K-Nearest Neighbours: Values of K (the number of neighbours) from 2 to 20 are tested. The best mean score is at K=4 with score of 0.552.

2) Support Vector Machine: The RBF kernel is used with the SVM classifier. A grid search of 15 gamma values and 15 C values are tested equally distributed between 0.01 and 20 for C and 0.001 and 0.05 for gamma. The choice of ranges for both parameters is selected after a smaller grid search size with larger ranges. The best mean score is at C=2.8657 and gamma=0.0045 with a mean score value of 0.551.

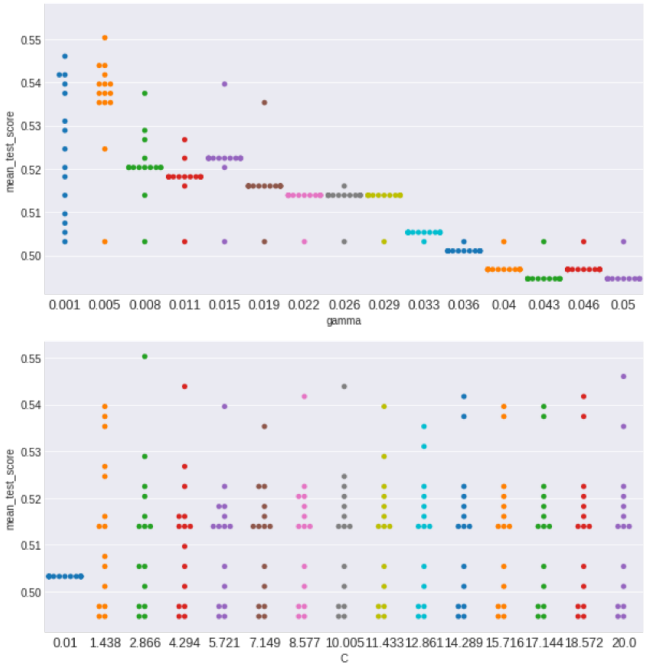
3) Gaussian Process: 8 different kernel combinations are tested using the same 5 fold cross validation method. The kernels are:

1**2 * RBF(length_scale=1)
1**2 * Matern(length_scale=1, nu=1.5)
1**2 * RBF(length_scale=1) + WhiteKernel(noise_level=1)
1**2 * Matern(length_scale=1, nu=1.5) + WhiteKernel(noise_level=1)
1**2 * RBF(length_scale=1) * 1**2 * Matern(length_scale=1, nu=1.5)
1**2 * RBF(length_scale=1) * 1**2 * Matern(length_scale=1, nu=1.5) + WhiteKernel(noise_level=1)
1**2 * RBF(length_scale=1) + 1**2 * Matern(length_scale=1, nu=1.5)
1**2 * RBF(length_scale=1) + 1**2 * Matern(length_scale=1, nu=1.5) + WhiteKernel(noise_level=1)

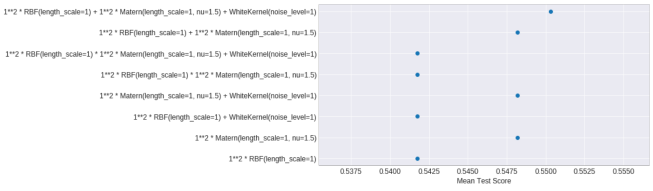
The kernel with the best mean accuracy score is 1**2 * RBF(length_scale=1) + 1**2 * Matern(length_scale=1, nu=1.5) + WhiteKernel(noise_level=1) with score of 0.550.



(a) K Nearest Neighbours



(b) Support Vector Machine



(c) Gaussian Process

Fig. 6: The parameters of the three classifiers are tuned using grid search. The plots represent the mean accuracy score of the 5 fold cross validation at each parameter.

After finding the right parameters, all three classifiers are fit on the full training set and then used in the back-testing on the testing period of 2016 and 2017. The classification strategy for each classifier is as follows:

```
At every trading day Predict
the movement of the price
IF NOT Invested AND classifier
```

	Buy&Hold	SMA	SMA_GMM	SEMA	SEMA_GMM	SVC	KNN	GP
Sharpe Ratio	0.64	0.49	0.77	0.49	0.7	0.89	0.12	0.17
Portfolio Value	139249	120764	139688	119023	130613	146630	98424	101124

TABLE I: Results of the back-testing of the strategies on the Tesla stock price on the testing period of 2016 and 2017.

```
predicts Up THEN
    BUY
IF Invested AND classifier
predicts Down THEN
    SELL
```

V. Results

The 8 strategies are back-tested on the trading testing environment on Tesla stock price from 2016-1-1 to 2017-12-31. Table I shows the results of the Sharpe ratio and the portfolio value at the end of each back-test. 4 strategies (SMA, ESMA, KNN and GP) performed worse than the Buy and Hold strategy. SVM Classifier strategy performed the best with 0.89 Sharpe ratio and over USD 7000 in profit over the Buy and Hold benchmark. The GMM risk manager shows a significant improvement with an increase of the Sharpe Ratio from 0.49 for SMA and SEMA to 0.77 and 0.7 respectively. SMA with GMM risk manager strategy is the second best performance after SVM. Although the profit return is slightly higher than the Buy and Hold strategy, the Sharpe ratio is higher, which indicates that SMA with GMM risk manager strategy has an effect on decreasing the risk of the trading. KNN and GP classifiers performed the worst with a Sharpe ratio of 0.12 and 0.17 respectively. None of the strategies performed well at every month over the 2 years period. Figure 7 and Appendix B shows the detailed results of the back-testing with every trading action and monthly returns over the testing period. Interesting to notice that different strategies performed differently on specific trends. The moving average strategies with and without risk manager tend to lose heavily on the 7th month of 2017, where the price has a fast drop. Yet they are able to predict the drop in the first month of 2016. The SVM classifier strategy performs in almost an opposite way with a large loss at the first month of 2016 and with a positive return on the 7th month of 2017.

VI. Conclusion

The Tesla stock price is highly risky choice for investment, however it is also the most potentially profitable because of the many minimum and maximum optimums in the price over time. The algorithmic strategies explored in this project are all relatively simple strategies. From the results observed none of the strategies are able to perform well on every month of the trading period. However, the strategies behaved differently on different trends in the price signal. For future work, this result shows evidence that one single strategy is very risky to always perform well on a high volatility price signal. A good approach is to combine multiple strategies, each working well on a different price trend, together using an expert model bandit algorithm like LinUCB[6].

References

[1] W. F. Sharpe, "The sharpe ratio," vol. 21, no. 1, pp. 49–58. [Online]. Available: http://jpm.iijournals.com/lookup/doi/10.3905/jpm.1994.409501

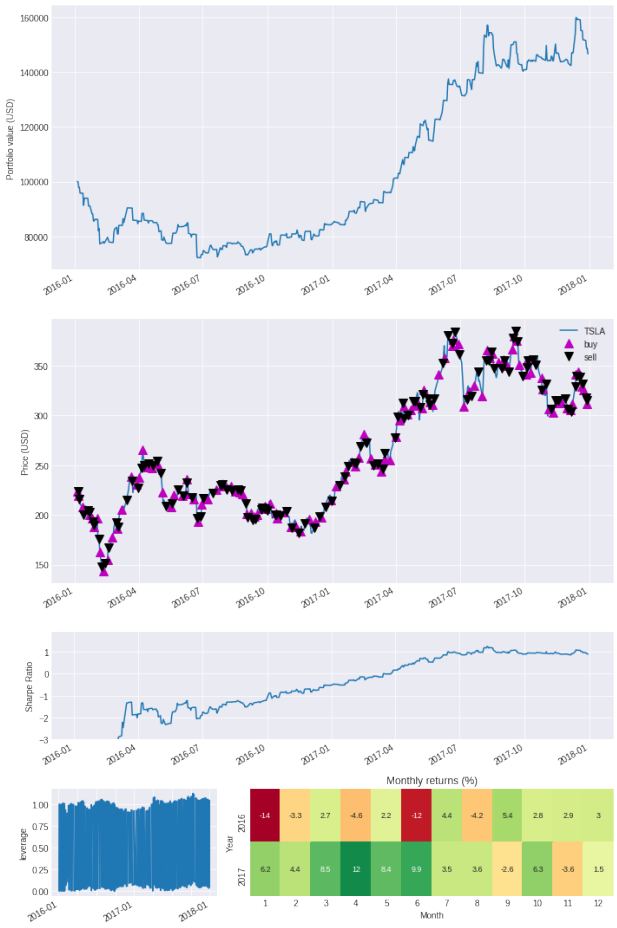
[2] "Kolmogorovsmirnov test," in *The Concise Encyclopedia of Statistics*. Springer New York, pp. 283–287. [Online]. Available: https://doi.org/10.1007/978-0-387-32833-1_214

[3] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: is a correction for chance necessary?" in *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*. ACM Press, pp. 1–8. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1553374.1553511

[4] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms." [Online]. Available: http://arxiv.org/abs/1206.2944

[5] E. W. Weisstein. Exponential moving average. [Online]. Available: http://mathworld.wolfram.com/ExponentialMovingAverage.html

[6] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation." [Online]. Available: https://arxiv.org/abs/1003.0146



(a) SVM Classifier Strategy

(b) SMA with GMM Risk Manager Strategy

Fig. 7: Back-testing results of the two best performing strategies.

APPENDIX A
GMM REGIME DETECTION TEST RESULT

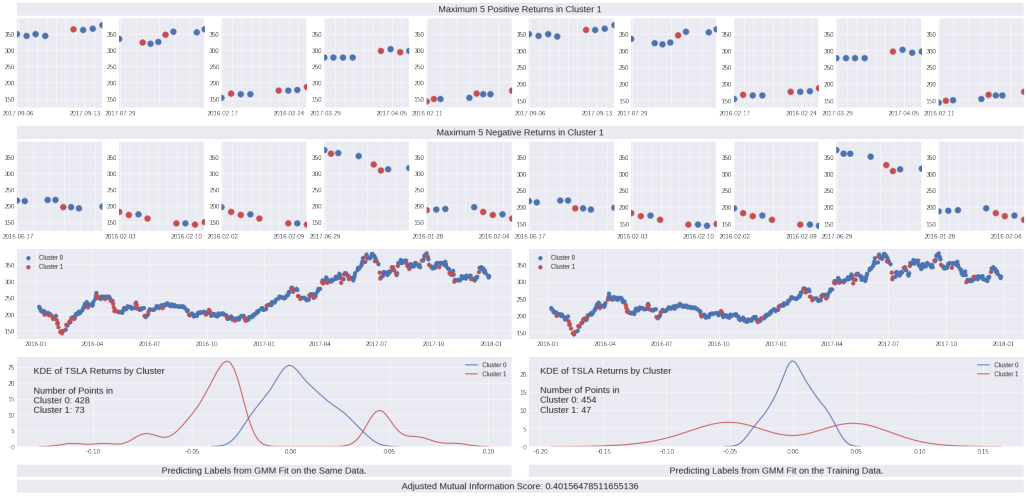
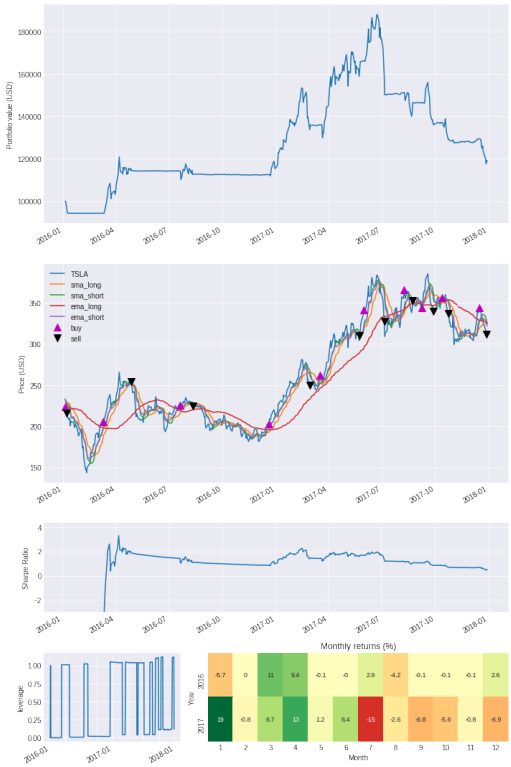


Fig. 8: Two GMM models are used to predict the logarithmic return of the testing set. GMM_{test} is fit on the logarithmic return of the training set of 2014-2015 and GMM_{true} is fit on the logarithmic return of the testing set of 2016-2017. The results shows that the 2 models did not detect the same outliers in the logarithmic return, however the top 5 positive and negative outliers predicted are the same.

APPENDIX B
BACK-TESTING RESULTS

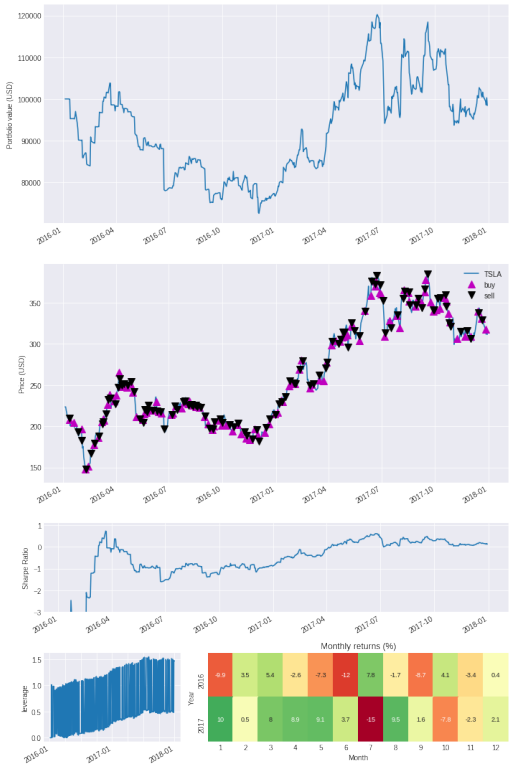
(a) Buy and Hold Strategy

(b) SMA Strategy

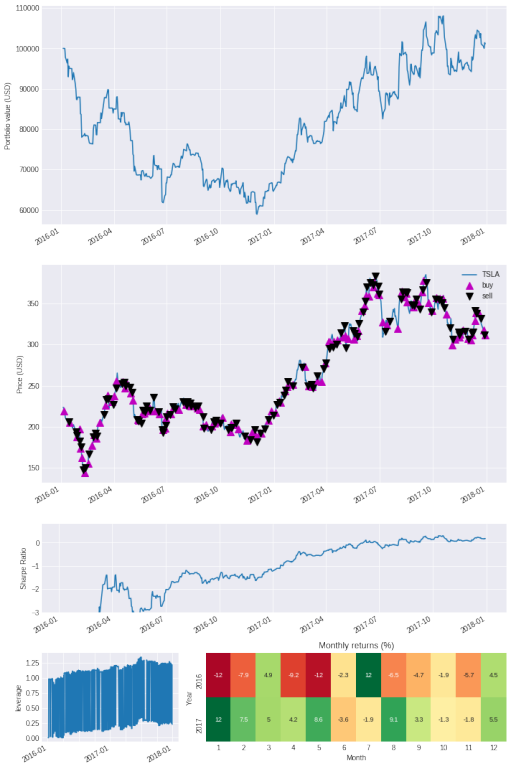


(c) SEMA Strategy

(d) SEMA with GMM Risk Manager Strategy



(e) KNN Classifier Strategy



(f) GP Classifier Strategy

APPENDIX C
REPRODUCIBLE WORK

All the code used for this project can be found presented in Jupyter notebooks on GitHub:

https://github.com/tawjaw/AlgoTradingML

To simplify the installation a docker image is created and uploaded to DockerHub. The docker image is based on Dockerfile found in zipline repo. The image includes the Zipline environment and other libraries used in this project. Some libraries installed using !pip command in Jupyter.

Installation instructions for the docker image can be found on the GitHub repository of this project.