# Machine Learning Nano Degree Capstone Project - Time series momentum using Machine Learning

August 2016

## 1 Problem Statement

Trading stocks is complicated and challenging for individuals and investment firms. For a profitable strategy, one needs to identify the price to buy the securities and the price to sell the securities in the future. Despite this, investors are constantly reviewing previous prices history and using it to influence their future investment decisions. In this project, we apply the time series momentum strategy in paper [1] to predict stock returns. It would be trivial to back out future stock prices from its returns. Momentum has been a main factor for stock returns. The assumption is that the winners in the past are likely to be winners in the future. From behavior finance's perspective, the fear and greed of investors' are the drivers for this phenomenon. Why would one bet on losers instead of winners?

Broadly speaking, momentum can be further divided into cross-sectional momentum and time-series momentum. Cross-sectional momentum literature finds that securities that outperform their peers in the past are likely to continue the relative outperformance in the future. In contrast, time series momentum purely focuses on the securities' own past returns. Academic research states that the time series momentum reflects the behavior finance theories of investors' initial under-reaction and delayed over-reaction to new informations.

We build a stock return predictor based on time series momentum [1] and both regression and clustering machine learning algorithms. The regression model will predict the absolute value of the future stock return and our strategy makes bet based on the prediction. Classification models predict the sign of the return. We compare regression and classification methods to find the best model for each security. Regression methods used include linear regression, Decision tree and AdaBoost regression. Classification methods used include K neighbour classifier, Gaussian Naive Bayes and AdaBoost classifier. The trading strategy has been backtested in a broad space of asset classes, such as equity futures, commodity futures and treasury bond futures. A broad set of global securities across asset classes are used to demonstrate the universal existence of the time series momentum effect and the profitability We obtain the securities price data from Bloomberg. To the best of our knowledge, this is the first work applying machine learning on time series momentum. Results show that classification algorithms delivers a high Sharpe ratio and little correlation to passive benchmarks.

## 2 Metrics

To evaluate the fitness of the regressional learning models, root mean squared error (RMSE) is used. It indicates the absolute fit of the model to the data. Lower RMSE values indicate better fit. We use F1 score to evaluate the clustering algorithms, which is a measure of the precision and recall of the algorithm.

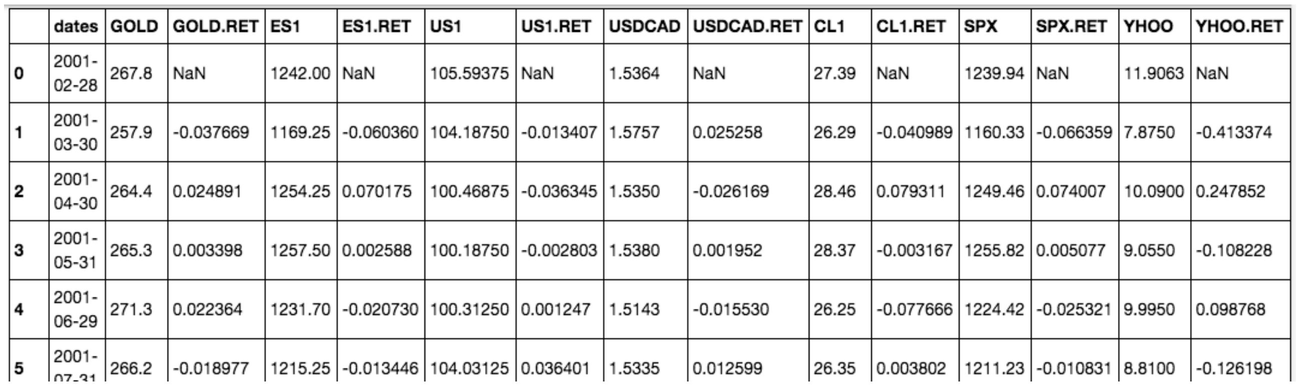
To evaluate the trading strategies, we use Sharpe ratio [2] to measure the return per unit risk. Sharpe ratio is defined as

$$Sharpe\ ratio = \frac{Annualized\ return}{Annualized\ standard\ deviation} \tag{1}$$

We also report the annualized return and annualized standard deviation, max return and max drawdown to evaluate the strategy from different perspective.

## 3 Data Exploration

We use various futures from different asset categories as the underlying security in our trading strategy including gold future, SPX future (ES1), tresury bond future (US1), oil future (CL1), currency pair (USDCAD), SPX Index and Yahoo US Equity. We present a piece of the raw data and summarize its descriptive statistics below. We observe that most futures have more positive returns than negative returns, though most of them have small Sharpe ratio. The exception is USDCAD, which have negative returns. However, we can trade CADUSD instead to make profits. The features for both of our regression and



| | dates | GOLD | GOLD.RET | ES1 | ES1.RET | US1 | US1.RET | USDCAD | USDCAD.RET | CL1 | CL1.RET | SPX | SPX.RET | YHOO | YHOO.RET |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2001-02-28 | 267.8 | NaN | 1242.00 | NaN | 105.59375 | NaN | 1.5364 | NaN | 27.39 | NaN | 1239.94 | NaN | 11.9063 | NaN |
| 1 | 2001-03-30 | 257.9 | -0.037669 | 1169.25 | -0.060360 | 104.18750 | -0.013407 | 1.5757 | 0.025258 | 26.29 | -0.040989 | 1160.33 | -0.066359 | 7.8750 | -0.413374 |
| 2 | 2001-04-30 | 264.4 | 0.024891 | 1254.25 | 0.070175 | 100.46875 | -0.036345 | 1.5350 | -0.026169 | 28.46 | 0.079311 | 1249.46 | 0.074007 | 10.0900 | 0.247852 |
| 3 | 2001-05-31 | 265.3 | 0.003398 | 1257.50 | 0.002588 | 100.18750 | -0.002803 | 1.5380 | 0.001952 | 28.37 | -0.003167 | 1255.82 | 0.005077 | 9.0550 | -0.108228 |
| 4 | 2001-06-29 | 271.3 | 0.022364 | 1231.70 | -0.020730 | 100.31250 | 0.001247 | 1.5143 | -0.015530 | 26.25 | -0.077666 | 1224.42 | -0.025321 | 9.9950 | 0.098768 |
| 5 | 2001-07-31 | 266.2 | -0.018977 | 1215.25 | -0.013446 | 104.03125 | 0.036401 | 1.5335 | 0.012599 | 26.35 | 0.003802 | 1211.23 | -0.010831 | 8.8100 | -0.126198 |

Figure 1: A piece of the raw dataset

classification methods are the past 12 months lagged return of the securities.

| Asset | Gold | ES1 | US1 | CL1 | USDCAD | SPX | Yahoo |
|---|---|---|---|---|---|---|---|
| Number of data points | 185 | 185 | 185 | 185 | 185 | 185 | 185 |
| Positive returns | 103 | 112 | 100 | 102 | 84 | 113 | 100 |
| Negative returns | 82 | 73 | 85 | 83 | 101 | 72 | 84 |
| Annualized returns | .11 | 0.04 | 0.03 | 0.03 | -0.01 | 0.04 | 0.08 |
| Standard deviation | 0.17 | 0.15 | 0.11 | 0.32 | 0.096 | 0.149 | 0.42 |
| Sharpe ratio | 0.58 | 0.24 | 0.31 | 0.08 | -0.11 | 0.24 | 0.18 |
| Cumulative returns | 403% | 74.5% | 65.2% | 52% | -15.2% | 75% | 220% |

## 4 Exploratory Visualization

Historical prices and returns, autocorrelation plot and partial autocorrelation plot for selected securities are presented below. We observe that the price of most securities are increasing over time , though there are some big drawdowns sometimes. And there are both large amount of positive and negative data points in returns. Our strategy is to try to filter out those drawdown ( negative return points ) and maximize positive return. From the autocorrelation and partial autocorrelation figures, we find there are certain lags that are highly correlated with current returns. Thus we use the past 12 months' monthly returns as the features of our predicting model. The mean of the returns is pretty around 0 while the median has a high estimation error.

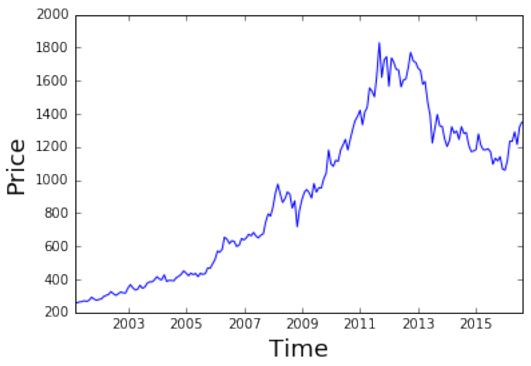
From the features scatter plot, we could tell that the features are independent among each other.

We show the price and return information of other securities' in future 3 and figure 4.

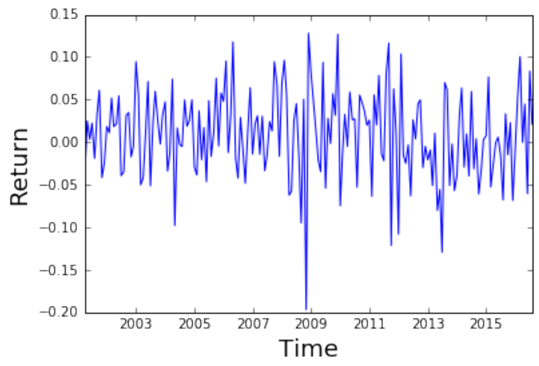
## 5 Algorithms and Techniques

Even though all securities tested have positive returns over time, their Sharpe ratio are pretty low, meaning that investors need to tolerate a lot of risk in order to have the returns. Our objective is to leverage time series momentum theory and machine learning prediction models to identify the timing to long a security.

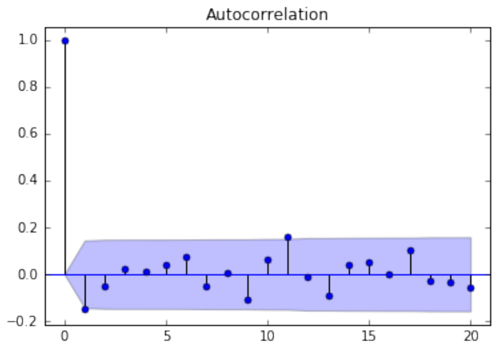
We use past 12 months' lagged returns to predict the return of next month. Both regression and classification models are used. We then compare results from regression and classification models in order to find the best model for a single security. The regression model is to predict the value of the future return. Long the security if the predicted value



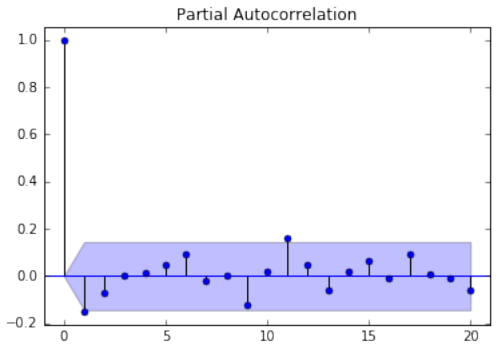
(a) Gold price



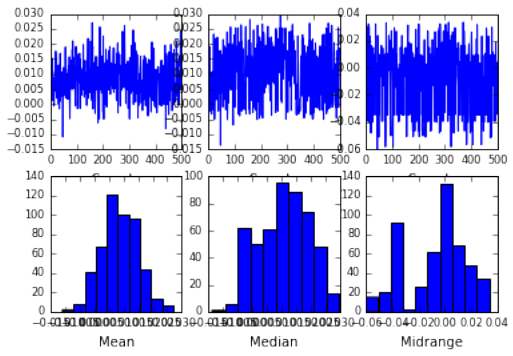
(b) Return of Gold



(c) Autocorrelation



(d) Partial autocorrelation



(e) Return of Gold

Figure 2: Gold data

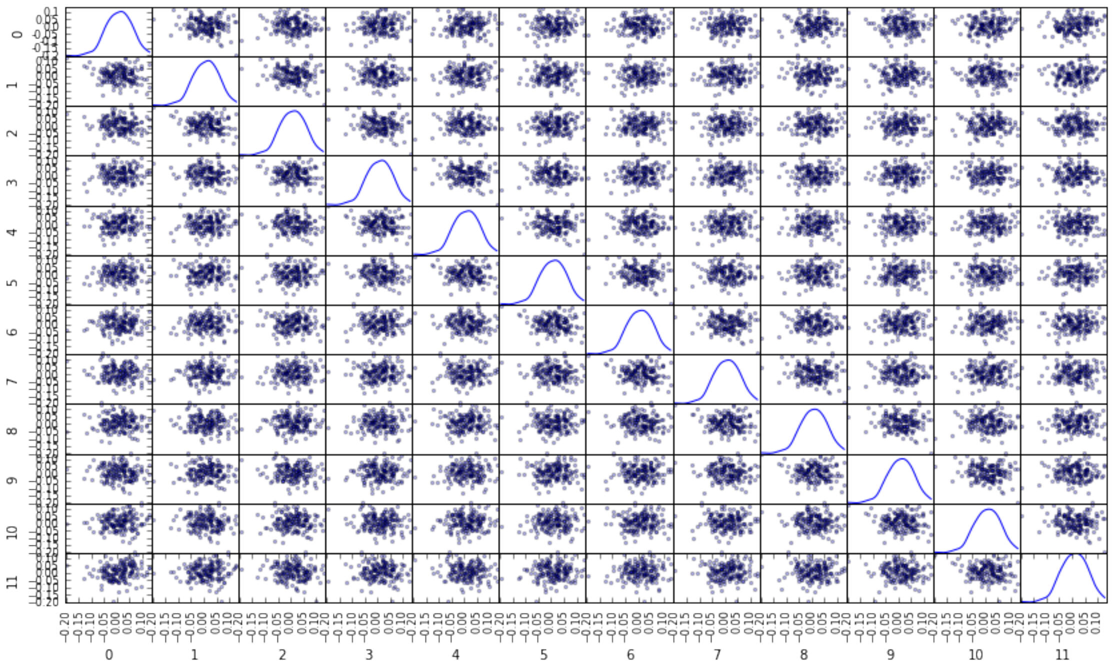


Figure 3: Feature scatter plot

is positive or greater than some threshold (see Refinement section), otherwise just held the cash. We also applied classification models. Long the security if the predicted sign of future return is positive.

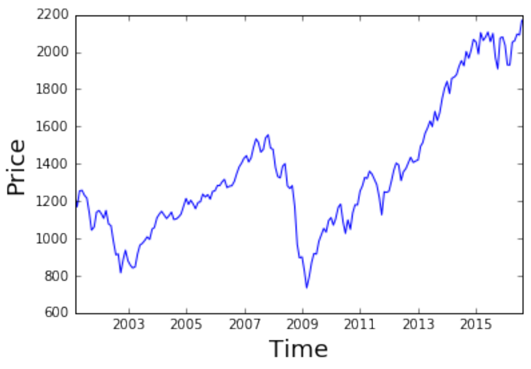
We use Linear regression, decision tree regressor and adaboost regressor and K neighbours classifier, Gaussian naive bayes and adaboost classifier.

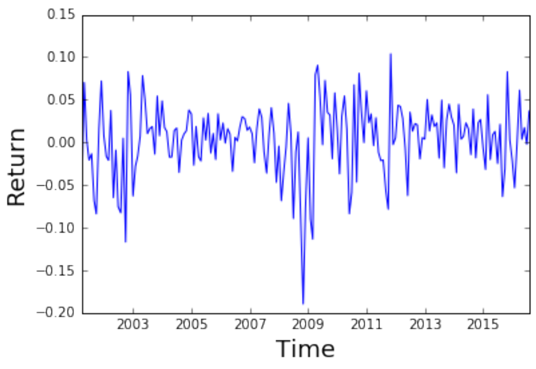
Basically the predicting models take the view of following regressions:

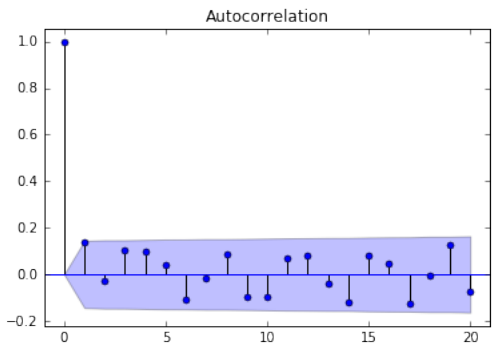
$$r_t = \alpha + \beta_h r_{t-h} + \epsilon_t \tag{2}$$

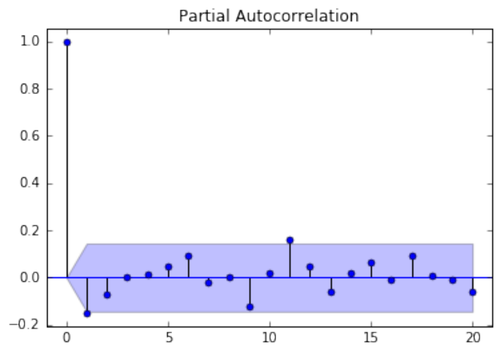
and

$$sign(r_t) = \alpha + \beta_h r_{t-h} + \epsilon_t \tag{3}$$


(a) ES future price


(b) ES future return


(c) Autocorrelation


(d) Partial autocorrelation

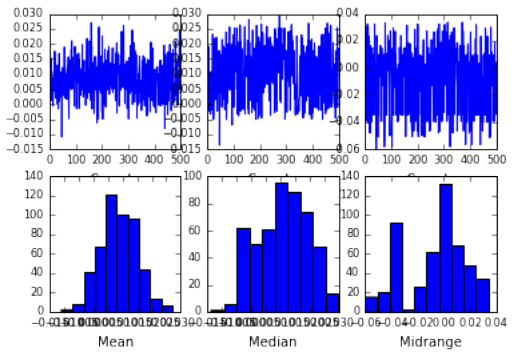
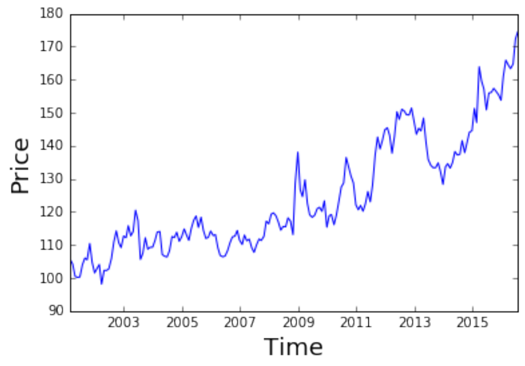
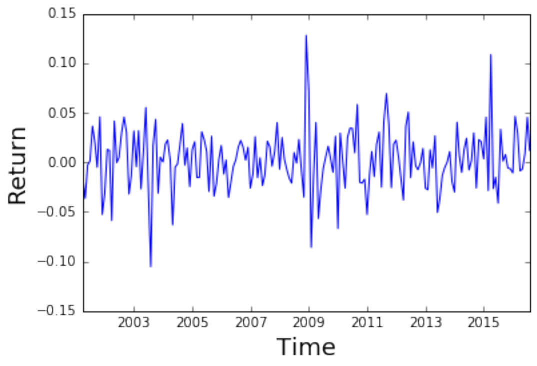
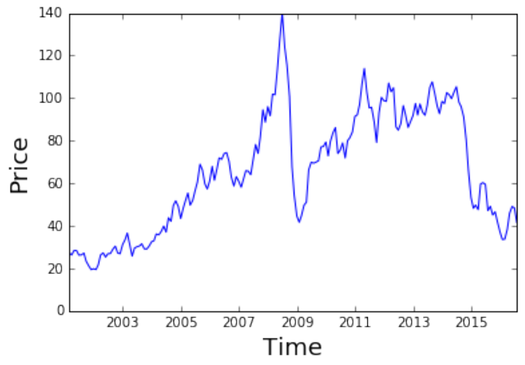
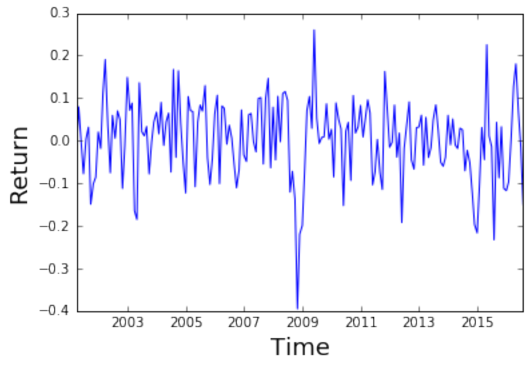

(e) Return of Es future distribution

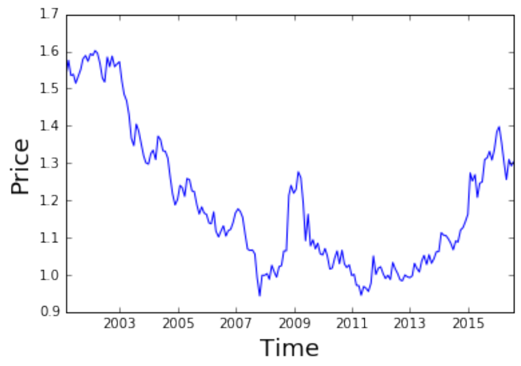
Figure 4: ES future data

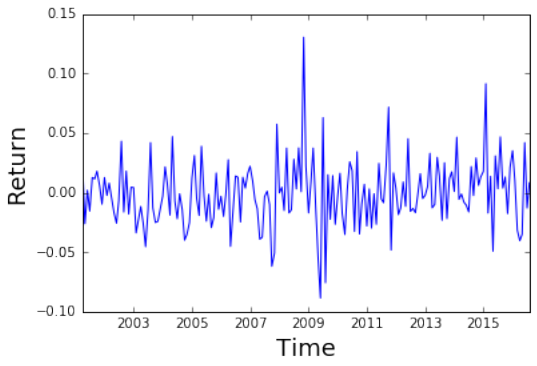

(a) US1 future price

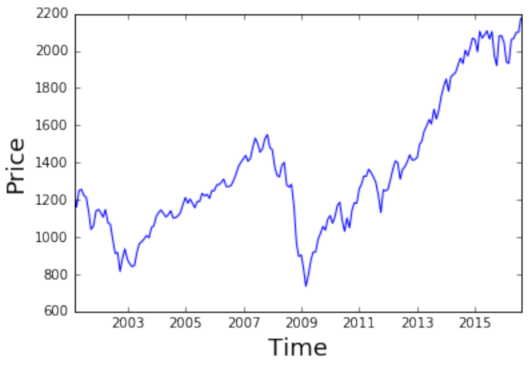

(b) US1 future return

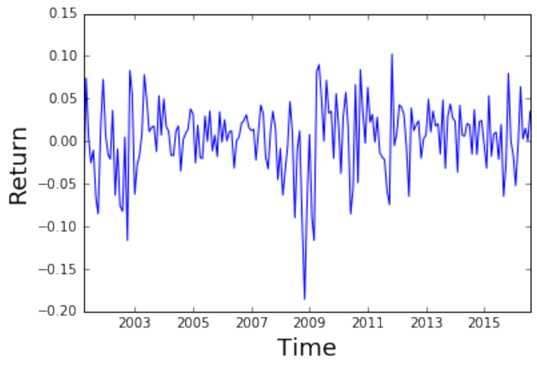

(c) CL1 future price

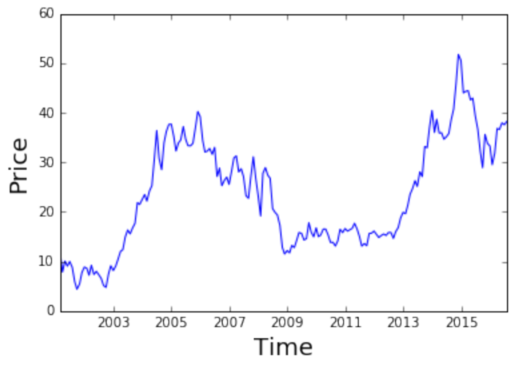

(d) CL1 future return


(e) USDCAD price

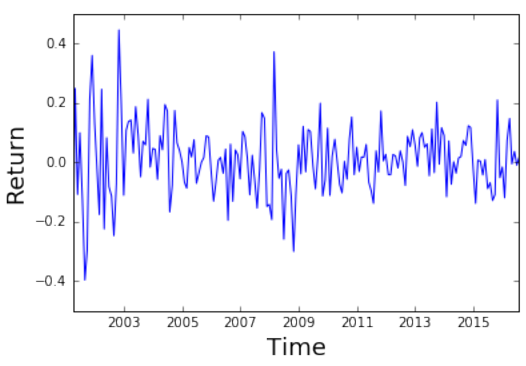

(f) USDCAD return


(g) SPX price


(h) SPX return



(i) Yahoo price



(j) Yahoo return

Figure 5: Other securites price and return

where $r_t$ is the return at time t, h is the lag, $\alpha$ is a constant, $\beta_h$ is the regression coefficient and $\epsilon_t$ is the error at time t.

We use linear regression to fit the data as it is used in the original paper. It is simple and can find a best fit line by minimizing the least-squared error between predicted and actual dependent variable. In addition, it is also very interpretable as we can directly tell which lagged returns are the best predictor for future returns.

Decision tree regressor is another algorithm we applied. It implicitly performs feature selection through the process of developing the tree. The top nodes of the tree are the most important ones as they determine the subsequent decisions to be made. Also, it is exhaustive to list all consequences and remain very descriptive.

Adaboost regressor is a ensamble learning model. It is used in conjunction with several other models and weight them to get a sum that represents the final output of the boosted regressor. Thus it transforms weak models to strong models. In this way, it will help minimize overfitting effects by averaging the predictive results from several models.

We also think to predict the sign of the returns is also helpful in justifying future returns. So we also test classification models to predict the sign of future returns.

We first apply K neighbours model on the problem. It is a simple and non parametric model, it does not make any assumption about the underlying data and the cost of the learning process is zero as it is a lazy learning model. But it is sensitive to the local structure of the data.

Naive bayesian classifier is based on Bayes theory with the assumption that all features are conditionally independent of each other, thus it also fits our dataset (see Figure 2) . Also it is fast in terms of training and making predictions. It only requires a small amount of data to train the model. But the disadvantage of this algorithm is that its precision and recall are low when using a small dataset. We use this model as it is fast to train and predict.

We also apply AdaBoost classifier due to reasons explained above. Its advantage in reducing reducing bias and variance will help reduce the overfitting effect.

In addition, we use a simple forward volatility model to determine the sizing of the portfolio as below:

$$\sigma_t^2 = 261 \sum_{i=0}^{\infty} w_i (r_{t-1} - \bar{r_t}^2) \tag{4}$$

where the weight function is selected to reflect the view the more recent variance are more correlated with future variance. And the sizing is $\frac{0.4}{\sigma_t}$ to minimize the overall volatility of the returns.

## 6 Benchmark

The baseline strategy is the market, namely the SPX Index. Thus our trading strategies shall beat its Sharpe ratio, which is 0.24. Since we use machine learning algorithms to avoid negative returns in order to maximize the Sharpe ratio, we'll also specify the false positive rate of the buy signal. We set it to be 0.5 as the predicted model shall outperform a random guess.

## 7 Data preprocessing

We transfer the price data to log return of the corresponding security. This to eliminate the unit root in the data. After this transformation, all the returns are within reasonable range. Please refer to data visualization section for return plots. All the data are generated by real stock market. So we believe each point represents the market under certain conditions. Therefore, there is no outlier to filter out.

## 8 Implementation

The price data of the futures contract is downloaded from Bloomberg. We then calculate the log return of each date. Each monthly returns up to 12 months are the features of our model. And the dependent variable is the return for this month (regression model) or the sign of the return for this month (classification model)

We implement both regression and classification learning algorithm in our trading strategy in Python. Sklearn library is used in implementing the training model and cross validation.

There are a few complications during the coding process. First of all, we need to construct the feature matrix that holds a running view of the lagged 12 months' returns. I use following function to get the matrix from a given array of returns:

```python
def running_view(arr, window, axis=-1):
    """
    return a running view of length 'window' over 'axis'
    the returned array has an extra last dimension, which spans the window
    """
    shape = list(arr.shape)
    shape[axis] -= (window-1)
    assert(shape[axis]>0)
    return np.lib.index_tricks.as_strided(
        arr,
        shape + [window],
        arr.strides + (arr.strides[axis],))
```

Listing 1: Running view of an array

In addition, researching each models and their pros and cons require some study on the sklearn documentation. One needs to understand why and how the model fits the data before implementing and optimizing them.

For the rest, it is straight forward to implement training the models and trading algorithms according to paper 1.

Regarding the methodology behind the refinement, we have applied GridSearch to each of the models on each of the securities. In this way, the hyper parameters of the models will be tuned based on the training data. Also, the trading strategy is enhanced by adding a threshold parameter. Please see the Refinement section for a deeper explaination.

## 9 Refinement

We first refine the prediction model by cross validating their hyper parameters. Decision tree regressor is tuned with its maximum depth (*max_depth*). We optimize AdaBoost regressor by tuning its learning rate (*learning_rate*) and number of estimators (*n_estimators*). Same parameters are tuned for AdaBoost classifier. K neighbors model is optimized with its number of neighbors (*n_neighbors*), algorithm, and leaf size (*leaf_size*) parameters. Linear regression and naive bayes have no hyper parameters.

After the supervised model is trained, we compare the predictive return with a predefined threshold. We'll be making our bet only if the predictive return exceeds the threshold. This is to minimize noise.

```
def sup_trading_strat(predict_returns, y_test):
    total_returns = []
    for i in range(1, len(predict_returns)):
        vol = volatility_estimate(y_test[0:i])

        if predict_returns[i-1] >= threshold:
            total_returns.append(y_test[i] * 0.4 / vol)
        elif predict_returns[i-1] < -threshold:
            total_returns.append(-y_test[i] * 0.4 / vol)

    avg_ret = np.mean(total_returns) * 12
    avg_std = np.std(total_returns) * np.sqrt(12)
    sharpe = avg_ret/avg_std
    cum = np.cumsum(total_returns)
    return avg_ret, sharpe, cum
```

Listing 2: Trading strategy based on regression

## 10 Results

The results using both strategies are summarized in table 1 to 7. It shows that different models have different predicting power to different time series. For example, linear regression is good for ES future, while decision tree is good for SPX Index while adaboost regression generally have good performance across all securities. K neighbour model performs well on gold future. Gaussian naive bayes solves the Yahoo dataset well and adaboost works well on almost all securities. In general, the proposed strategies outperform the benchmark. By using the strategies, we can increase the Sharpe ratio to 0.8 to 1.8, meaning the strategies help investors make more returns while taking less risk. Almost all of the models beat the benchmark in terms of Sharpe ratio and false positive rate except for the currency pair.

However, in this case, one can trade cadusd instead to turn negative Sharpe ratio into positive ratio.

| Regressor | Linear regression | Decision tree | AdaBoostRegressor |
|---|---|---|---|
| RMSE | 0.058 | 0.080 | 0.058 |
| Avg return | 0.053 | 0.067 | 0.06 |
| Sharpe ratio | 0.616 | 0.841 | 0.695 |
| False positive | 0.67 | 0.75 | 0.71 |

| Classifier | KNeighbour (n$_{n}eighbors = 5$) | Gaussian Naive Bayes | AdaBoostClassifier |
|---|---|---|---|
| F1 score (in sample) | 0.677 | 0.603 | 0.968 |
| F1 score (out of sample) | 0.509 | 0.627 | 0.577 |
| Avg return | 0.107 | 0.056 | 0.076 |
| Sharpe ratio | 1.484 | 0.706 | 1.075 |
| False positive | 0.76 | 0.65 | 0.68 |

Table 1: Trading strategy using gold. Decision tree param: as deep as possible. AdaBoost-Regressor params: n_estimators = 50, learning_rate=1. K neighbors params: algorithm = auto, n_neighbors = 5, leaf_size = 30. AdaBoostClassifier params: n_estimators = 50, learning_rate = 1.

| Regressor | Linear regression | Decision tree | AdaBoostRegressor |
|---|---|---|---|
| RMSE | 0.043 | 0.062 | 0.038 |
| Avg return | 0.127 | 0.066 | 0.126 |
| Sharpe ratio | 1.448 | 0.836 | 1.314 |
| False positive | 0.61 | 0.75 | 0.625 |

| Classifier | KNeighbour | Gaussian Naive Bayes | AdaBoostClassifier |
|---|---|---|---|
| F1 score (in sample) | 0.802 | 0.733 | 0.987 |
| F1 score (out of sample) | 0.730 | 0.769 | 0.642 |
| Avg return | 0.086 | 0.067 | 0.059 |
| Sharpe ratio | 1.040 | 0.761 | 0.749 |
| False positive | 0.76 | 0.685 | 0.739 |

Table 2: Trading strategy using ES future. Decision tree param: as deep as possible. AdaBoostRegressor params: n_estimators = 50, learning_rate=1. K neighbors params: algorithm = auto, n_neighbors = 5, leaf_size = 30. AdaBoostClassifier params: n_estimators = 50, learning_rate = 1.

| Regressor | Linear regression | Decision tree | AdaBoostRegressor |
|---|---|---|---|
| RMSE | 0.035 | 0.038 | 0.036 |
| Avg return | 0.038 | -0.048 | 0.023 |
| Sharpe ratio | 0.787 | -0.593 | 0.454 |
| False positive | 0.56 | 0.5 | 0.6 |

| Classifier | KNeighbour | Gaussian Naive Bayes | AdaBoostClassifier |
|---|---|---|---|
| F1 score (in sample) | 0.689 | 0.721 | 0.985 |
| F1 score (out of sample) | 0.612 | 0.712 | 0.5 |
| Avg return | -0.035 | 0.027 | -0.01 |
| Sharpe ratio | -0.459 | 0.372 | -0.156 |
| False positive | 0.384 | 0.571 | 0.45 |

Table 3: Trading strategy using US1 future. Decision tree param: as deep as possible. AdaBoostRegressor params: n_estimators = 50, learning_rate=1. K neighbors params: algorithm = auto, n_neighbors = 5, leaf_size = 30. AdaBoostClassifier params: n_estimators = 50, learning_rate = 1.

| Regressor | Linear regression | Decision tree | AdaBoostRegressor |
|---|---|---|---|
| RMSE | 0.087 | 0.113 | 0.0832 |
| Avg return | 0.053 | 0.053 | -0.009 |
| Sharpe ratio | 0.616 | 0.616 | -0.193 |
| False positive | 0.666 | 0.619 | 0.642 |

| Classifier | KNeighbour | Gaussian Naive Bayes | AdaBoostClassifier |
|---|---|---|---|
| F1 score (in sample) | 0.606 | 0.687 | 1 |
| F1 score (out of sample) | 0.55 | 0.565 | 0.625 |
| Avg return | 0.031 | -0.024 | 0.082 |
| Sharpe ratio | 0.57 | -0.375 | 1.375 |
| False positive | 0.6 | 0.6 | 0.772 |

Table 4: Trading strategy using CL1 future. Decision tree param: as deep as possible. AdaBoostRegressor params: n_estimators = 50, learning_rate=1. K neighbors params: algorithm = auto, n_neighbors = 5, leaf_size = 30. AdaBoostClassifier params: n_estimators = 50, learning_rate = 1.

| Regressor | Linear regression | Decision tree | AdaBoostRegressor |
|---|---|---|---|
| RMSE | 0.026 | 0.046 | 0.0279 |
| Avg return | -0.08 | -0.011 | -0.053 |
| Sharpe ratio | -1.164 | -0.099 | -0.57 |
| False positive | 0.5 | 0.428 | 0.1 |

| Classifier | KNeighbour | Gaussian Naive Bayes | AdaBoostClassifier |
|---|---|---|---|
| F1 score (in sample) | 0.603 | 0.545 | 1 |
| F1 score (out of sample) | 0.438 | 0.4 | 0.457 |
| Avg return | -0.036 | -0.036 | -0.107 |
| Sharpe ratio | -0.77 | -0.479 | -1.8 |
| False positive | 0.33 | 0.33 | 0.06 |

Table 5: Trading strategy using USDCAD. Decision tree param: as deep as possible. AdaBoostRegressor params: n_estimators = 50, learning_rate=1. K neighbors params: algorithm = auto, n_neighbors = 5, leaf_size = 30. AdaBoostClassifier params: n_estimators = 50, learning_rate = 1.

| Regressor | Linear regression | Gaussian Naive Bayes | AdaBoostRegressor |
|---|---|---|---|
| RMSE | 0.042 | 0.056 | 0.038 |
| Avg return | 0.12 | 0.07 | 0.10 |
| Sharpe ratio | 1.32 | 0.93 | 1.47 |
| False positive | 0.526 | 0.772 | 1.0 |

| Classifier | KNeighbour | Gaussian Naive Bayes | AdaBoostClassifier |
|---|---|---|---|
| F1 score (in sample) | 0.82 | 0.73 | 0.98 |
| F1 score (out of sample) | 0.69 | 0.57 | 0.51 |
| Avg return | 0.06 | 0.03 | 0.06 |
| Sharpe ratio | 0.74 | 0.46 | 0.87 |
| False positive | 0.67 | 0.58 | 0.695 |

Table 6: Trading strategy using SPX. Decision tree param: as deep as possible. AdaBoostRegressor params: n_estimators = 50, learning_rate=1. K neighbors params: algorithm = auto, n_neighbors = 5, leaf_size = 30. AdaBoostClassifier params: n_estimators = 50, learning_rate = 1.

| Regressor | Linear regression | Decision tree | AdaBoostRegressor |
|---|---|---|---|
| RMSE | 0.109 | 0.153 | 0.095 |
| Avg return | 0.065 | 0.045 | 0.06 |
| Sharpe ratio | 1.03 | 0.727 | 0.94 |
| False positive | 0.586 | 0.52 | 0.55 |

| Classifier | KNeighbour | Gaussian Naive Bayes | AdaBoostClassifier |
|---|---|---|---|
| F1 score (in sample) | 0.73 | 0.74 | 1 |
| F1 score (out of sample) | 0.67 | 0.62 | 0.54 |
| Avg return | 0.078 | 0.096 | 0.075 |
| Sharpe ratio | 1.02 | 1.258 | 1.07 |
| False positive | 0.57 | 0.645 | 0.625 |

Table 7: Trading strategy using Yahoo. Decision tree param: as deep as possible. AdaBoost-Regressor params: n_estimators = 50, learning_rate=1. K neighbors params: algorithm = auto, n_neighbors = 5, leaf_size = 30. AdaBoostClassifier params: n_estimators = 50, learning_rate = 1.

## 11 Justification

We present the results of final models below. All models are tuned and optimised according to Section refinement. Results show that the optimised models have smaller RMSE and higher F1 score. In addition, their average returns, Sharpe ratio and false positive rate are better than the models with benchmark setting. All models work on all securities except USDCAD and they beat the benchmark, Sharpe ratio (0.24) and false positive 50%. All the Sharpe ratios and false positive rates are computed by using out of sample data and thus the results will be robust to new dataset. The hyper parameters of the final models are presented in following tables.

| Regressor | Linear regression | Decision tree | AdaBoostRegressor |
|---|---|---|---|
| RMSE | 0.058 | 0.052 | 0.052 |
| Avg return | 0.053 | 0.128 | 0.172 |
| Sharpe ratio | 0.616 | 0.97 | 1.43 |
| False Positive | 0.67 | 0.72 | 0.8125 |

| Classifier | KNeighbour | Gaussian Naive Bayes | AdaBoostClassifier |
|---|---|---|---|
| F1 score (in sample) | 0.678 | 0.603 | 0.824 |
| F1 score (out of sample) | 0.51 | 0.627 | 0.64 |
| Avg return | 0.107 | 0.056 | 0.05 |
| Sharpe ratio | 1.484 | 0.706 | 0.65 |
| False Positive | 0.76 | 0.65 | 0.63 |

Table 8: Trading strategy using optimized model with gold futures. Decision tree param: max_depth = 1. AdaBoost param: n_estimators = 51, learning_rate = 0.02. K neighbors params: n_neighbors = 1, leaf_size = 25, algorithm = auto. AdaBoostClassifier params: n_estimators = 50, learning_rate = 0.1.

| Regressor | Linear regression | Decision tree | AdaBoostRegressor |
|---|---|---|---|
| RMSE | 0.043 | 0.039 | 0.038 |
| Avg return | 0.127 | 0.034 | 0.307 |
| Sharpe ratio | 1.448 | 0.34 | 2.67 |
| False Positive | 0.61 | 0.25 | 0.66 |

| Classifier | KNeighbour | Gaussian Naive Bayes | AdaBoostClassifier |
|---|---|---|---|
| F1 score (in sample) | 0.80 | 0.733 | 0.79 |
| F1 score (out of sample) | 0.73 | 0.769 | 0.77 |
| Avg return | 0.086 | 0.067 | 0.063 |
| Sharpe ratio | 1.04 | 0.761 | 0.67 |
| False Positive | 0.765 | 0.685 | 0.669 |

Table 9: Trading strategy using optimized model with ES futures. Decision tree param: max_depth = 1. AdaBoost param: n_estimators = 51, learning_rate = 0.01. K neighbors params: n_neighbors = 5, leaf_size = 25, algorithm = auto. AdaBoostClassifier params: n_estimators = 50, learning_rate = 0.02.

| Regressor | Linear regression | Decision tree | AdaBoostRegressor |
|---|---|---|---|
| RMSE | 0.035 | 0.035 | 0.33 |
| Avg return | 0.038 | -0.116 | 0.17 |
| Sharpe ratio | 0.787 | -0.85 | 2.16 |
| False Positive | 0.56 | 0.5 | 0.75 |

| Classifier | KNeighbour | Gaussian Naive Bayes | AdaBoostClassifier |
|---|---|---|---|
| F1 score (in sample) | 0.689 | 0.721 | 0.792 |
| F1 score (out of sample) | 0.612 | 0.712 | 0.631 |
| Avg return | -0.035 | 0.027 | 0.036 |
| Sharpe ratio | -0.46 | 0.372 | 0.48 |
| False Positive | 0.385 | 0.571 | 0.6 |

Table 10: Trading strategy using optimized model with US1 futures. Decision tree param: max_depth = 1. AdaBoost param: n_estimators = 52, learning_rate = 0.05. K neighbors params: n_neighbors = 1, leaf_size = 25, algorithm = auto. AdaBoostClassifier params: n_estimators = 48, learning_rate = 0.05.

| Regressor | Linear regression | Decision tree | AdaBoostRegressor |
|---|---|---|---|
| RMSE | 0.087 | 0.078 | 0.85 |
| Avg return | 0.053 | 0.005 | 0.17 |
| Sharpe ratio | 0.616 | 0.099 | 2.16 |
| False Positive | 0.666 | 0.64 | 0.75 |

| Classifier | KNeighbour | Gaussian Naive Bayes | AdaBoostClassifier |
|---|---|---|---|
| F1 score (in sample) | 0.6065 | 0.687 | 0.88 |
| F1 score (out of sample) | 0.55 | 0.565 | 0.58 |
| Avg return | 0.03 | -0.024 | 0.02 |
| Sharpe ratio | 0.56 | -0.375 | 0.36 |
| False Positive | 0.6 | 0.6 | 0.64 |

Table 11: Trading strategy using optimized model with CL1 futures. Decision tree param: max_depth = 1. AdaBoost param: n_estimators = 48, learning_rate = 0.02. K neighbors params: n_neighbors = 5, leaf_size = 25, algorithm = auto. AdaBoostClassifier params: n_estimators = 51, learning_rate = 0.1.

| Regressor | Linear regression | Decision tree | AdaBoostRegressor |
|---|---|---|---|
| RMSE | 0.026 | 0.034 | 0.0268 |
| Avg return | -0.08 | -0.21 | -0.63 |
| Sharpe ratio | -1.164 | -1.59 | -3 |
| False Positive | 0.5 | 0.1 | 0 |

| Classifier | KNeighbour | Gaussian Naive Bayes | AdaBoostClassifier |
|---|---|---|---|
| F1 score (in sample) | 0.603 | 0.545 | 0.77 |
| F1 score (out of sample) | 0.4375 | 0.4 | 0.4 |
| Avg return | -0.036 | -0.036 | -0.06 |
| Sharpe ratio | -0.77 | -0.479 | -1.11 |
| False Positive | 0.33 | 0.33 | 0.33 |

Table 12: Trading strategy using optimized model with USDCAD. Decision tree param: max_depth = 2. AdaBoost param: n_estimators = 48, learning_rate = 0.02. K neighbors params: n_neighbors = 5, leaf_size = 25, algorithm = auto. AdaBoostClassifier params: n_estimators = 51, learning_rate = 0.1.

| Regressor | Linear regression | Decision tree | AdaBoostRegressor |
|---|---|---|---|
| RMSE | 0.042 | 0.0348 | 0.037 |
| Avg return | 0.12 | 0.126 | 0.24 |
| Sharpe ratio | 1.32 | 1.29 | 3.46 |
| False Positive | 0.526 | 0.67 | 1.0 |

| Classifier | KNeighbour | Gaussian Naive Bayes | AdaBoostClassifier |
|---|---|---|---|
| F1 score (in sample) | 0.82 | 0.73 | 0.782 |
| F1 score (out of sample) | 0.69 | 0.57 | 0.742 |
| Avg return | 0.06 | 0.03 | 0.049 |
| Sharpe ratio | 0.74 | 0.46 | 0.52 |
| False Positive | 0.67 | 0.58 | 0.634 |

Table 13: Trading strategy using optimized model with SPX futures. Decision tree param: max_depth = 1. AdaBoost param: n_estimators = 51, learning_rate = 0.05. K neighbors params: n_neighbors = 5, leaf_size = 25, algorithm = auto. AdaBoostClassifier params: n_estimators = 51, learning_rate = 0.1.

| Regressor | Linear regression | Decision tree | AdaBoostRegressor |
|---|---|---|---|
| RMSE | 0.109 | 0.116 | 0.095 |
| Avg return | 0.065 | 0.09 | 0.085 |
| Sharpe ratio | 1.03 | 1.18 | 1.117 |
| False Positive | 0.586 | 0.5 | 0.523 |

| Classifier | KNeighbour | Gaussian Naive Bayes | AdaBoostClassifier |
|---|---|---|---|
| F1 score (in sample) | 0.73 | 0.74 | 0.805 |
| F1 score (out of sample) | 0.67 | 0.62 | 0.62 |
| Avg return | 0.078 | 0.096 | 0.093 |
| Sharpe ratio | 1.02 | 1.258 | 1.235 |
| False Positive | 0.57 | 0.645 | 0.612 |

Table 14: Trading strategy using optimized model with Yahoo stock. Decision tree param: max_depth = 1. AdaBoost param: n_estimators = 49, learning_rate = 0.02. K neighbors params: n_neighbors = 5, leaf_size = 25, algorithm = auto. AdaBoostClassifier params: n_estimators = 49, learning_rate = 0.02.

## 12 End to end summary

Step 1: preprocessing the data. We turn the price to log returns and turn the log returns into a running view of the last 12 months' returns to be ready for the models to consume.

Step 2: Use the preprocessed data to train various machine learning models.

Step 3: Use GridSearch to tune the parameters of the models.

Step 4: Generate predicted returns.

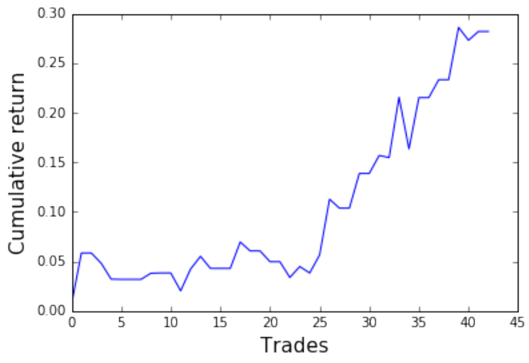
Step 5: Use the predicted returns to trade the time series momentum effect.

Step 6: Compare the average returns and Sharpe ratios of different models.

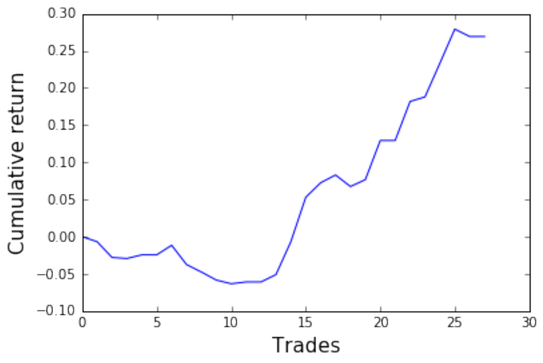
It is interesting the machine learning models, both before and after tuning can beat the benchmark significantly.

## 13 Conclusion

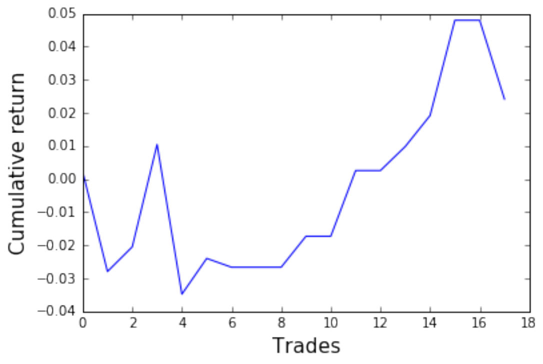
In this project, we study the time series momentum in various instruments, where investors can profit from the time series momemtum in securities. We apply both regression and classification models on two trading strategies that can enhance the Sharpe ratios of trading time series momentum. Additionally, we optimise the machine learning models by cross validation and we also add a threshold on the trading strategy. Results presented in table 5 show that the proposed trading strategies (pink bars) significantly outperform default models (blue bars) and the benchmark.



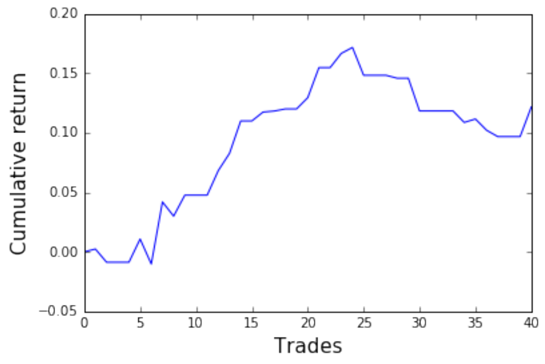
(a) Gold future cumulative return using K neighbors



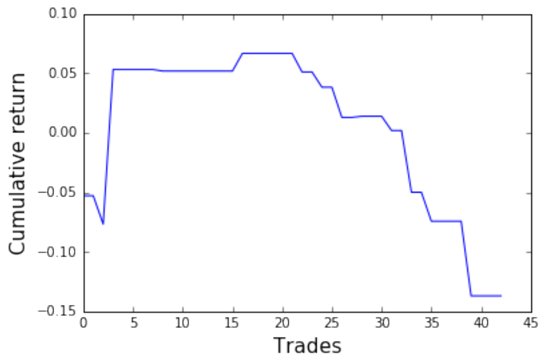
(b) ES future cumulative return using linear regression



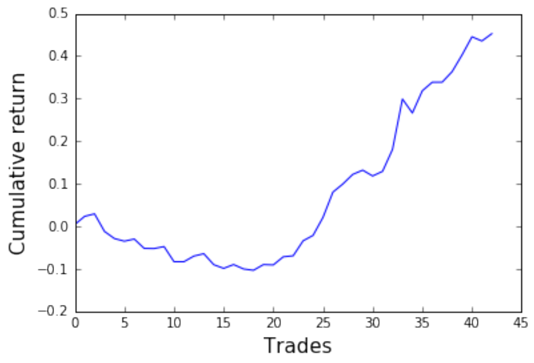
(c) CL1 future cumulative return using AdaBoost regressor



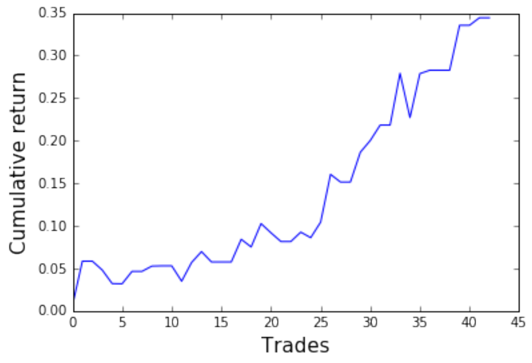
(d) US future cumulative return using AdaBoost Regressor



(e) USDCAD cumulative return using Decision tree

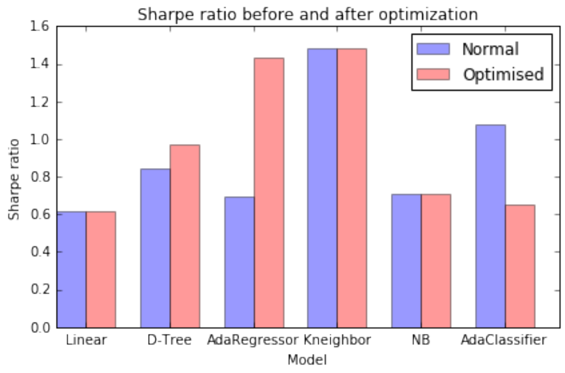


(f) SPX Index cumulative return using AdaBoost classifier

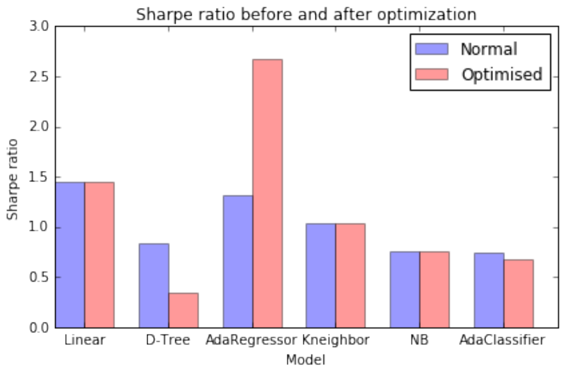


(g) Yahoo stock cumulative return using Gaussian Naive Bayes

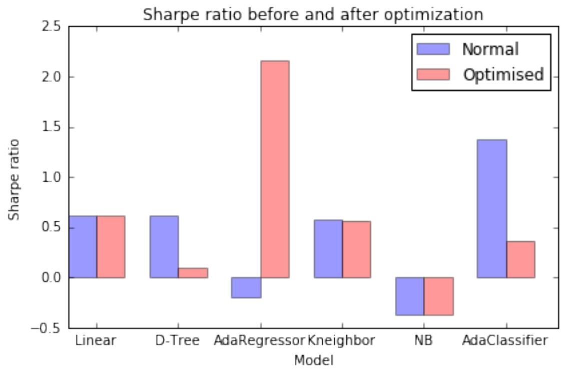
Figure 6: Cumulative returns using optimal models



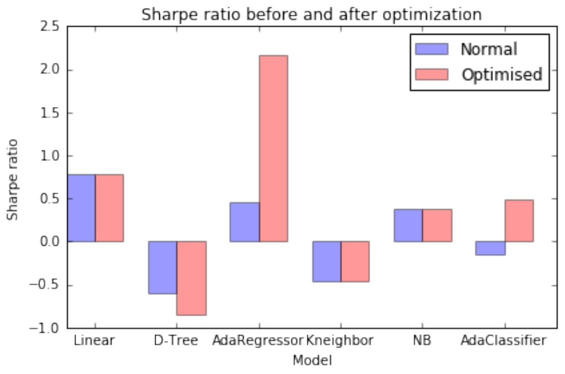
(a) Gold future



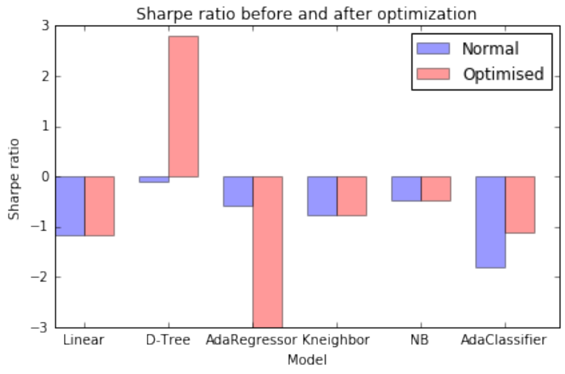
(b) ES future



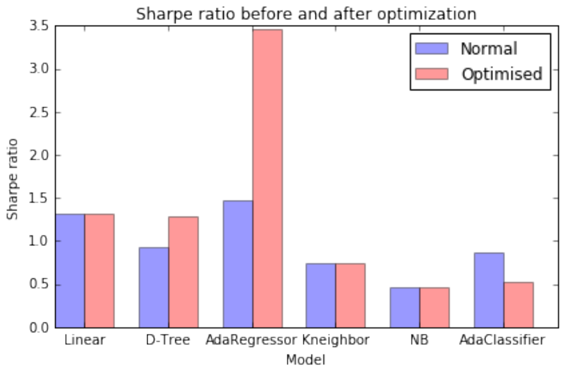
(c) CL1 future



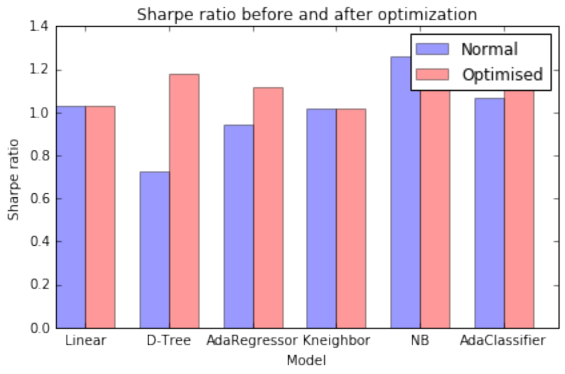
(d) US future



(e) USDCAD



(f) SPX Index



(g) Yahoo stock

Figure 7: Sharpe ratio comparison

## 14 Possible Improvements

More features that may predict future returns can be included. Such data may include macro economics data, such as interest rates, non-farm payroll and GDP quarter over quarter. Though we have thoroughly examine various algorithms on a number of assets, one can extend the the model space to include deep learning models to select those useful features. In addition, the trading strategy can be further refined, the decision can be made by taking the returns in the past few months into consideration rather than only one month.

## 15 Reference

1. Moskowitz, Ooi, Pedersen: Time Series Momentum, Journal of Economics, 2010

2. https://en.wikipedia.org/wiki/Sharpe$_ratio$