BMO Hedging Assignment

Christopher Luciuk[1]

[1] *1708 - 925 Bay St., Toronto, ON, M5S 3L4*

(Dated: October 15, 2017)

The purpose of this exercise is to build and validate a model that selects securities that track the target instrument's performance. The trading scheme requires that the performance of a predictive algorithm be successful after a time T. The underlying model uses simple linear regression or an exponential moving average to extrapolate existing data forward in time. This is then incorporated into the trading scheme to validate the model.

Data Exploration and Cleaning I began by cleaning the data set. An initial exploration showed that all the date information was present but not each security price was present for every day. To fill in this data in the quickest way I used the security price from the two days preceding a missing value to linearly extrapolate to the new value. The function **clean** performs this operation on any one-dimensional array of data. Alternatively, I considered interpolating between the dates with complete data preceding and following the missing value. However, if the data was being collected in a constant data stream while running the model this method would fail as the following data would be missing. In addition, the extrapolation method seemed most computationally efficient as it didn't require a determination of the number of missing data points.

I was originally worried about the sampling of the data in time (the dates periodically increased by three) but soon realized that the data was built from trading days only and so the date variable was really a dummy variable and could be swapped for an arbitrary array with evenly spaced sampling.

Having cleaned the data I began exploring it by plotting sample security prices. In a cursory examination of trends in the time-series data it appeared that there were some salient features I needed to be aware of:

- *How noisy is the data?* Some data sets are relatively smooth while others have large random fluctuations. The "noisiness" will impact the certainty of the model's prediction.

- *Are there any large and sudden changes in the security?* Some securities have large and sudden decreases (or increases) in their value. A good model will try to minimize the effect of these volatile securities.

- *Do changes in the data look linear? Quadratic? More complicated?* Often a linear regression is simplest to describe small chunks of data. In this case, my best guess for a function to describe the data is linear.

I include a plot of some characteristic data exhibiting these features in Fig. 1. In both sets of time series data shown the trends are likely best approximated by a linear function over a short period of time. However, the blue data exhibits oscillatory behaviour which would result in a large uncertainty when performing a linear fit. While it may be useful in trading to make use of this periodic behaviour to make a profit (i.e., buy low and sell high), this security is likely too variable to be of much use in this project.

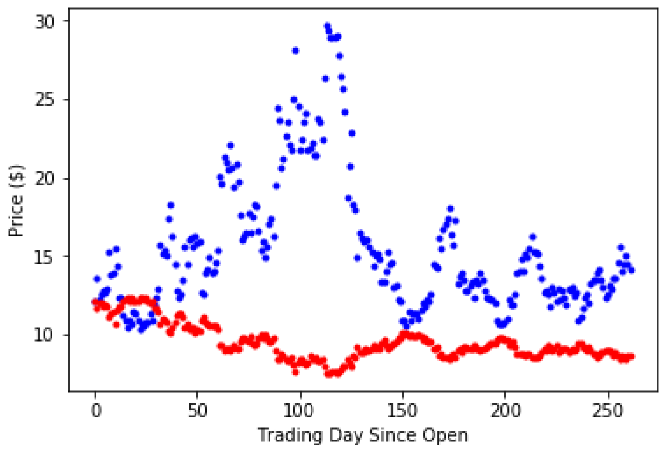

FIG. 1. The time series behaviour of two sample securities is shown. By eye, I would identify the blue data as being quite volatile and changing over large amounts and rapidly. The red data is much more smooth.

Building a Model The first step in building a model is to accurately predict the behaviour of a test security after time T. I started with the naive approach of performing a linear regression using the first n points in the data set and extrapolating for a time T. Here n is a hyper parameter that can be chosen at the discretion of the user.

Figures 2 and 3 show the residuals (predicted value minus true value) for two data sets. In Fig. 2 $n = 5$ while in Fig. 3 $n = 25$. It seems that by increasing n the residuals decrease in amplitude but when the number of points included in the fit is too large, rapid changes in the data are not captured by the predictive model.

The model also depends on the value of T chosen. In Fig. 2 and Fig. 3 $T = 5$. As one might expect, as T

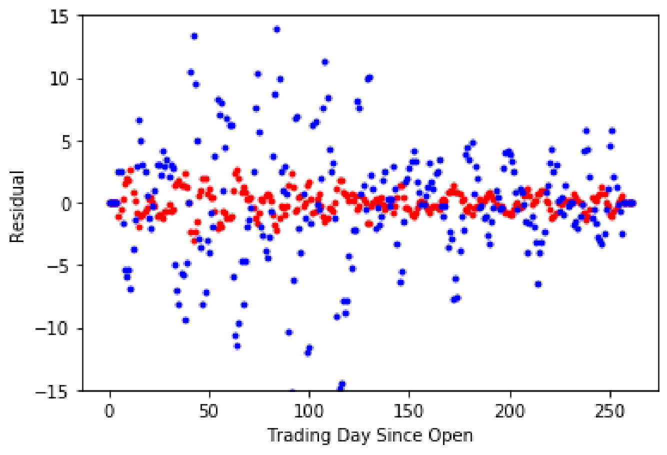

FIG. 2. The residuals for two sample securities from a linear fit using the previous 5 points. Colours are as in Fig. 1.

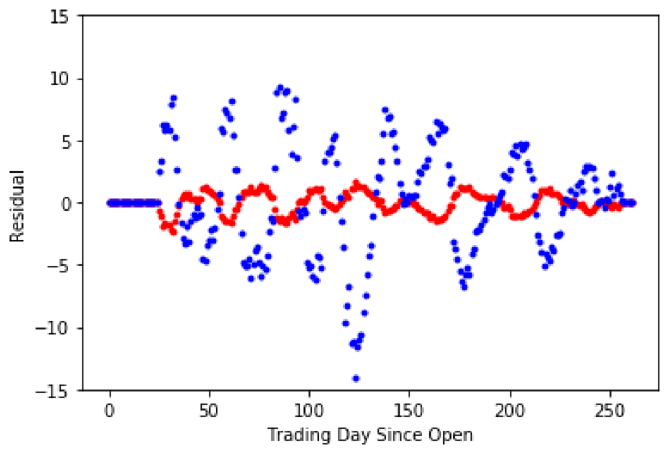

FIG. 3. The residuals for two sample securities from a linear fit using the previous 25 points. Colours are as in Fig. 1.

increases the residuals grow in amplitude. This is shown in Fig. 4 where $T = 25$ and $n = 25$.

By looking at the plots of the residuals it is interesting to note that there is substantial structure. In fact, the structure looks sinusoidal. A fit using a sinusoid finds a period that is consistent with the choice of n. It appears that by using a linear fit I am constantly over estimating and under estimating the true value of the security. This might not be an awful strategy if the residuals themselves are small. Perhaps some regularization is needed to rectify this issue.

To contrast the linear model I built I also tried extrapolating using an exponential moving average. To implement this algorithm I calculated weights for the previous data and then used the calculated exponential moving average for time t as the predicted value for day $t + 1$. I then appended this value to my data for the next iteration of the calculation until I reached day $t + T$. The performance of the exponential moving average (and also

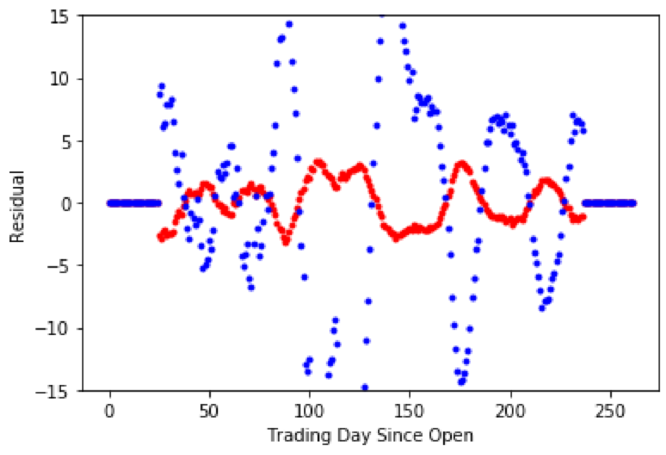

FIG. 4. The residuals for two sample securities from a linear fit using the previous 25 points where $T = 25$. Colours are as in Fig. 1.

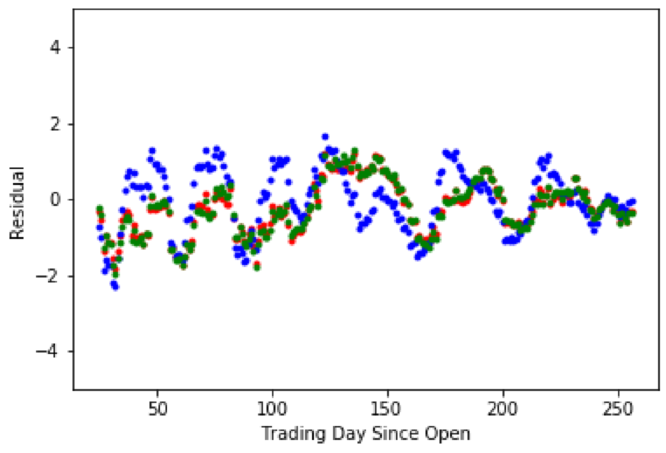

FIG. 5. A comparison of the various models using 25 previous points to predict forward in time 5 days. The magnitude of the residuals is comparable for a simple moving average (green), an exponential moving average (red) and a linear extrapolation (blue).

a simple moving average) in comparison to the linear extrapolation is shown in Fig. 5 for the data shown in red in Fig. 1. The red points show the exponential moving average while green points show the simple moving average. They track each other very well, although the performance of the exponential moving average is marginally better. The performance of the linear extrapolation is shown in blue. The magnitude of the residuals is comparable to that for the moving averages.

Having found a model that works reasonably well on "well-behaved" data I began to move forward to building a basket of securities to track the target's behaviour.

Choosing a Basket Selecting an optimal basket is complicated by the feature identified above: the volatility or how "well-behaved" the data is significantly impacts the validity of the prediction. As this exercise is about minimizing risk I want to identify securities that have low risk/volatility and are "well-behaved" to minimize errors in the predictions. One quick measure of the volatility of a security is the standard deviation. As the prediction algorithm looks at the preceding data, the volatility measure will also use the same past data. This implies that as the algorithm propagates forward in time the measure of volatility changes to reflect the most recent data.

To implement the algorithm that chooses the basket I first calculate the standard deviation for each security using a number of points n before the current time t. I order these securities from least volatile to most volatile. I then use the exponential moving average algorithm to predict the security price after T days for the least volatile securities. After this step I have a set of predicted changes in security price ordered from least volatile to most volatile.

The other ingredient in building the basket is predicting the change in the target instrument. I use the exponential moving average to predict the change in target price, δ. Having predicted δ I can build a basket from the least volatile securities to best track δ. To find the basket that tracks the target I first look at the permutations of the securities and the sum of their predicted changes. I then calculate the difference in the prediction for the target and the basket and if it is less than a threshold (or tolerance) I choose those securities to fill the basket. If the threshold is not met I increase it (by doubling, then tripling, etc.) until a basket is found.

To validate the performance of the model I follow the trading algorithm outlined:

1. Initialize the model with a random basket and at a random day in the data set.
2. Sell the basket and buy the target.
3. Predict the expected change in the target.
4. Wait T days
5. Find the basket that will best match the predicted change in the target when sold.
6. Sell the target and buy the basket.
7. Wait T days.
8. Sell the basket and buy the target.
9. How well did the predicted basket match the true target and predicted target?

I repeat this validation algorithm N times while varying the other model parameters to minimize the error in the prediction. The best performance is shown in Fig. 6. The residuals are scattered about zero and are generally constrained to be in the range $-1 < residual < +1$.

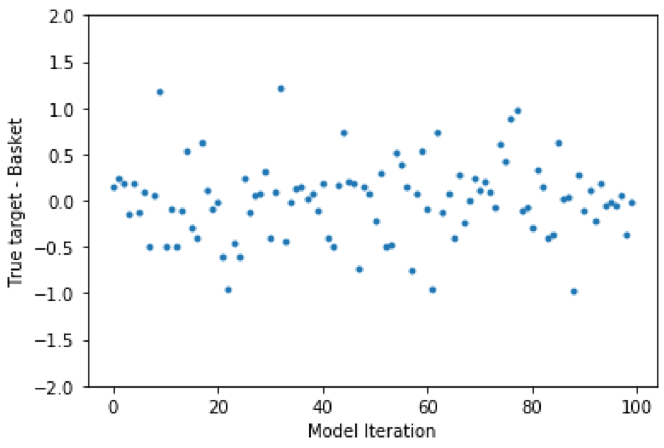

FIG. 6. The calcuated residual for the model with optimized hyperparameters. The largest uncertainty enters through the prediction of the target behaviour but is minimized by decreasing the tolerance in the basket building.

Capital Allocation Given \$10 million to invest in each target security I would choose to allocate capital to the securities in the basket to minimize risk. This corresponds to choosing the least volatile securities to invest most heavily in. In a sense, the least volatile securities (or the securities with the smallest standard deviation) would receive the highest weight in calculating the cost of the basket. This would help to further suppress the risk in hedging with highly volatile securities.

Scaling If instead of one observation per day there were thousands my model would scale well. The relevant hyperparameter to change would be the number of previous points that the model is trained on. With more samples the model does run the risk of overfitting small fluctuations in the data. As a result, the linear extrapolation may work more efficiently to predict the securities.

If instead of hedging four securities, I was hedging thousands I would need to reevaluate the method I chose to find the securities to track the target behaviour. Perhaps the **combinations** function called would run too slowly to be effectively used. If the algorithm using this function was not prohibitively slow, the predictive model for building the basket should scale easily.

As the model only relies on a small sample of time series data preceding the current data, the model should run without needing to be retrained. Perhaps, an evaluation of the hyperparameters like the number of points and threshold for tracking the target could be optimized as more data is collected.

As stated above, an advantage of the model is the relatively small memory requirement. For each security to be hedged a small slice of time series data preceding it is required. This means a full stream of time series data could be stored in a server database while only the relevant slice of data could be pulled to run the predictive model.

Deployment of this model really depends upon the trading period T for which it is run. If the predictive period is a number of days the model could be run using security data at the end of each time period T to return a prediction for the next hedging period. As the period T decreased the speed of the model and utilization of the prediction would need to be adjusted such that it met the requirement of operating faster than the period T. A constant evaluation of the model could be carried out in parallel and the value δ could be adjusted in each iteration of the prediction to make up for the error in the previous iteration. As the model is quite general, it could also be easily modified to hedge to make a profit.

In sum, this model has very low memory requirements and can be easily deployed to make fast predictions. However, as the trading period T decreases the speed of the model will need to be evaluated to see if it scales appropriately.

Code Organization In the attached code I have organized the functions responsible for the model at the top of the file. The validation, exploration and plotting of the data is done in a notebook style but each section could easily be wrapped in a function. Calling the trading algorithm should be a simple function in the main body of the file.

Conclusion In this exercise a predictive model was created to find four securities to hedge to track a target instrument's behaviour. The time series data is analyzed and a linear extrapolation and exponential moving average were compared as predictive techniques to determine the security's behaviour after a time T. A proxy basket is then constructed out of these securities to minimize the volatility of the basket and to match the predicted behaviour of the target within a fixed tolerance.

This model should scale as the number of time series observations or securities increases, although consideration must be given for the time the algorithm takes to run as compared to the trading time T. Further, this model can easily be extended such that the proxy basket can be chosen to satisfy any criteria including generating profit in the hedge.