ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/338435971

Machine Learning & Foreign Exchange Rate Prediction

Presentation · December 2019

CITATIONS
0

READS
56

1 author:

Akin Wilson
University of Nottingham
10 PUBLICATIONS **3** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Non-negative Matrix Factorisation: Insight into Ill-posed aspect of the technique, rectifications & variations View project

All content following this page was uploaded by Akin Wilson on 25 January 2020.

The user has requested enhancement of the downloaded file.

Reviewing the Implementation of Statistical Machine Learning Methods with the Aim of Foreign Exchange Rate Prediction using Python

Akin Antony Wilson
2016 - 19 Mathematical Physics BSc.
2019 - 20 Machine Learning in Science MSc.
University of Nottingham, United Kingdom

***Abstract*— Statistical machine learning techniques have been applied to financial markets since the dawn of the computer age. In particular, the foreign exchange market due to its continuity, liquidity and trading volume is a perfect contender to apply machine learning methods to. Trading in this market is conducted on a continuous basis all around the globe, leading to stable numerical data and low transaction fees in comparison to other markets such as the equity, fixed-income and the derivatives market. This report summaries the implementation of four different statistical methods using Python. The models will be trained on the basis of predicting the base currency USD and counter currency GBP through a period of one month; Monday 2^{nd} September to Friday 11^{th} October, after training the models on a trading period of three months; Monday 3^{rd} of June to Friday 30^{th} August. In this report, coding choices will be highlighted, commenting upon advantages, disadvantages and areas for further improvement. The statistical models compared are:-**

- **Elastic Net Regularized Linear Regression**
- **Ridge Regularized Linear Regression**
- **K-Nearest Neighbours Regression**
- **Multi-Layer Perceptron Regression**

The report structure will be as follows: The data sourcing, handling and preparing will be reviewed first. Thereafter the implementation of the statistical methods will be commented upon and lastly, areas for further improvement and development will be discussed.

I. Data Sourcing, Handling & Preparation

A. Sourcing

Sourcing accurate and reliable data can be consider as important as the algorithms themselves. The industry maxim *garbage in garbage out* captures this principle very well: The quality of statistical predictions are highly depend on the quality of input data. Raw data used in this project stemmed from a Bloomberg Professional Services Terminal and Google Trends to ensure this review overcomes this maxim . The sourcing of the data could be made more efficient; packages such *Zipline* can be used to source technical indicator data on instruments and more. A step beyond readily built specialised packages would be to implement a custom program to automate the sourcing of data, in a similar fashion to *Zipline*. For this, packages such as *BeautifulSoup*; a web-scraping package or *Selenium*; a web interface automation package, would be essential.

B. Handling & Preparation

Python's data analysis library *Pandas* has all the functionality needed for the handling of data, and it is an industry standard for such tasks. This is why it was used for the data handling in the project. Data frames were grouped together using Python's dictionary method. This allows one to easily call upon various data frames in a order manner. Having have said that, an improvement would be to only use the built-in functionality of Python. Although at this scale, the choice of *Pandas* was made and would make little difference, the built-in functions lend themselves well to larger projects. Given their correct use, these would out perform any Python data handling packages in terms of speed of computation. In terms of the data itself, the temporal aspect of the data was key to keep track of. A reliable choice to aid with this is Python's timezone library *pytz*. It allows time naive data to become timezone aware and therewith one can convert between timezone if need be. The Python file titled *dataCleaningAndPreparing.py*[1] contains all the data cleansing and evaluating of the project. The file is sectioned off with the first implementing the preparation of raw data and the following evaluating and selecting features for later training and testing. There are plenty areas to improve upon in this section of the project. For example, the automation of filling gaps in raw data could be explored. Often, it was simply the prior or posterior sequential value in a time step of a series that was used to fill missing values. This process deems itself to be very labour-intensive and therefore, such functions would significantly improve future workflow. But, a potential issue with such a function would be how task specific it could become if one is not careful, rendering it useless for re-use. Consider line 185 to 281 in *dataCleaningAndPreparing.py*: A function could be implemented to load the indicator file, splitting it, adding missing midnight entries for the time series and replacing any of their missing values. But to be made reusable, the function would have to handle various date and time formats, automatically locate and infer missing values and furthermore, re-order the time series. This would drastically increase efficiency in the workflow, but could deem to be too

[1]To gain access to this file and all others mentioned in this report, please find them in my Github account: github.com/akinolawilson/Foreign-Exchange-Rate-Predicition

task specific to create. Such a function will be consider for the improvement of this project.

II. Feature Selection & Evaluation

A. Feature Evaluation

The evaluation of features was performed using various functions found in the *scikit − learn* library; a library for machine learning. It's use in the project can be found towards the end of the *dataCleaningAndPreparing.py* file. A project specific improvement here would be to base the comparison of the three different selection methods used (correlation heat map, Pearson's correlation coefficient and mutual information regression) on numerical methods. Instead, a visual evaluation was considered by investigating their plots. Implementing a function which would provide an overall score, cross-validating over all selection methods, would be a more robust and systematic path to take.

B. Feature Selection

Following the exploratory stage of feature evaluation, the final step of removing irrelevant features was taken in the last section of *dataCleaningAndPreparing.py*. This step was made thoroughly easy due how the data frames had been prepared in the earlier section of the code. Finally, the training and testing sets are exported as CSV files and can be found in the folder *preparedDataFiles* under *Train.csv* and *Test.csv*.

III. Model Training & Evaluating

The models compared in this project are:-

- Elastic Net Regularized Linear Regression
- Ridge Regularized Linear Regression
- K-Nearest Neighbours Regression
- Multi-Layer Perceptron Regression

A mathematical distinction can be made between the first two and latter models; they are parametric and non-parametric models respectively. For the task at hand, non-parametric models are proposed to be more effective, which is a hypothesis this project aimed to also investigate.

A. Model Training

The training, testing and evaluation of all models is contained in the file titled *trainingAndTesting.py* All the models used were sourced from the *scikit − learn* library. On top of the models themselves, the library provided metrics, data normalisation and cross-validation methods. A key design choice made here was to use a special k-fold splitting strategy for the parameter optimisation. This strategy splits the training set into sequential chunks, adding each chunk to the previous and then performing the optimisation again. By doing this, the temporal aspect of the data is taken into account when finding these parameters. One aspect of the training period that could be improved through coding alterations is the time taken to train the models. Making use of Python's *multiprocessing* library could decrease the training time of the models by distribution the computational workload across either all threads of a CPU or even better, multiple CPUs. Additional libraries such as *Dask* (a library for scalable analytics in Python) could be considered. In particular, the *Dask.distributed* module allows one to systematically manage computations across a cluster of CPUs on different machines. One downfall with applying such a package would be the design modifications required to be made on the existing code. Often, these changes made are executed by applying *decorators* to existing functions in the code. This is easily done with simple functions but becomes increasingly complex when implementing them on larger projects.

B. Model Testing & Results

Model testing was easily performed once the training period had been conducted. The results of the training and testing period are presented bellow:

	Model	Training Absolute Mean Error	Training Root Mean Square Error
0	Elastic Net	3.348734e-03	0.004135
1	MLP	3.268717e-03	0.004084
2	KNN	1.642297e-07	0.000005
3	Ridge	3.310354e-03	0.004043

Fig.1 shows the methods accuracy during the training phase, which we expect to be much lower than the test phase.

	Model	Test Absolute Mean Error	Test Root Mean Square Error
0	Elastic Net	0.024212	0.029382
1	MLP	0.033222	0.042282
2	KNN	0.013294	0.015096
3	Ridge	0.025745	0.031297

Fig.2 presents the results of applying each technique during the evaluation stage, where all the algorithms were acting on predicting unseen data.

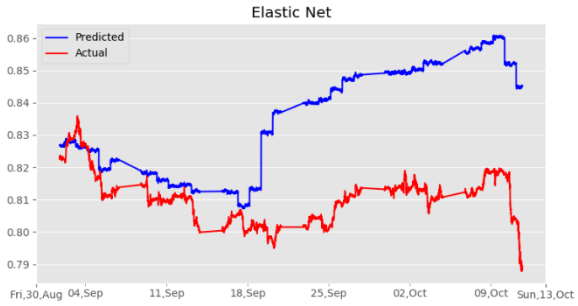

Fig.3 there is a noticeable divergence in almost all models, bar with the KNN approach; one of the non-parametric models. It can be seen from the elastic net regression predictions that around September 18^{th} 2019, a exogenous variable impacted their predictions. This was the change in monetary policy the Federal Reserve of the United States issued on the date, they had lowered their interest rates by 20 basis points from 2.00 to 1.80 percent.

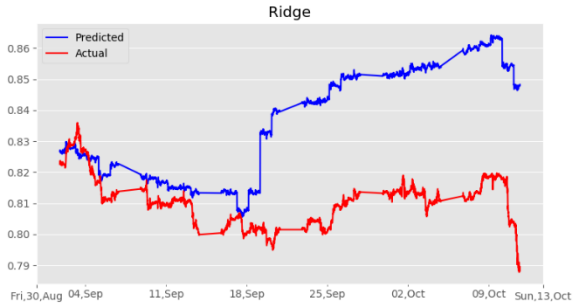

Fig.4 given their mathematical similarity, it should be no surprise that ridge regression produced similar results to elastic net regression.

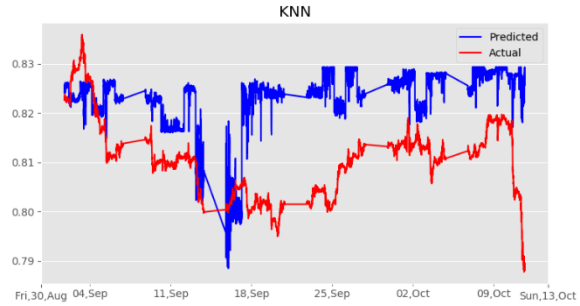

Fig.5 although K-nearest neighbours regression achieved the lowest mean error, it's volatility in predictions is a massive drawback of the technique, given the small amount of features used.

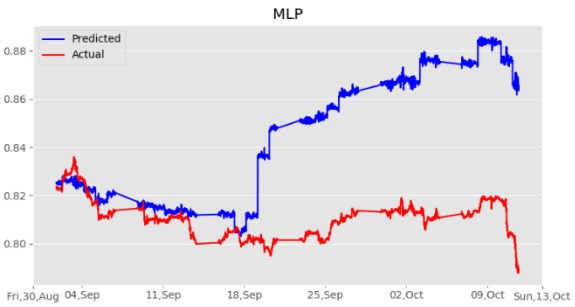

Fig.6 one would expect the multi-layer perceptron regression to perform the best, since the neural network architecture it is based on is able to capture highly non-linear relations between features. But again due to the small amount of features used, it fell to the same deception as the two parametric models, elastic net and ridge regression, although it itself is a non-parametric model.

IV. Discussion

A. Code Development and Outlook

This project has illustrated the feasibility of using statistical methods for predicting FX rates. The models predicted the closing price with the highest mean error of 150 pips of the true price. Although impressive in-of-itself, for the application of intraday, this error would have to decrease by around a factor of ten. To achieve this more features need to be incorporated into the models. This will be achieved by first selecting more relevant features and automate the sourcing of the most current feature data using a web-scraping approach. The implementation of a distributed workload system will also be tested with the aim to improve efficiency of training. Once a larger amount of features are included, this is likely to be key to making the computations tractable. Lastly, the methods are currently just predicting the closing price, but we are of course interested in placing positions to maximise yield. These aspects of FX trading will also be included in the later code development to take one step closer to live autonomous trading.

View publication stats