# HSBC Hackathon: report of Osman Zubair, Sean Billings, François Chalus {okz51, sb2219, fc443}@cam.ac.uk

The code can be found in the script mov\_avg.py (Python 2, see requirements at end of document)

Our feature engineering function computes the moving average of different quantities (bid/ask, bid/ask sizes, spread, book pressure) over different windows : such as the last {2, 5, 10, 320, 1280} times the order book changed or the last {20, 80, 1000, 16000} milliseconds. Then we compute the finite differences of those columns as we expect them to be smooth and provide some signal. We reused some of the features given in the notebook and slightly modified the book pressure feature to use all current orders in the order book. We also designed a volatility feature, computed over the last {1, 1e2, 1e4, 1e6} seconds. In designing the features we were very careful not to create signals that are not observable at the corresponding time.

In terms of data preprocessing, we discarded all rows that are NaNs. After preprocessing, our train and test sets are of size 96346 and 46019 respectively, they are roughly balanced in terms of up or down days (51.2% of the training and 49.4% of the test instances are up days).

We tried different regression models to predict the next move (and its magnitude) from the current features. Those included linear ridge regression, kernelized ridge regression using 1000 random Fourier features as well as two other methods that did not work (lasso methods which we could not train properly, SVM which suffered from the high number of datapoints and was too slow).

Our third predictor was that after an up move there came a down move so one of our predictor was just the previous timestep’s value. Since the performance on the training set was not great (51.3%) we only used this as a tie breaker between the two regression models used above, when ensembling them.

The classification rates of both ridge regression estimators are shown in the table below:

|  |  |  |
| --- | --- | --- |
| Classification rate | Train | Test |
| RFF Kernel ridge | 0.599 | 0.565 |
| Linear ridge | **0.606** | **0.580** |
| Mean reverting | 0.513 | 0.509 |
| Ensemble (equal weights, ties broken by prev day) | 0.603 | 0.573 |

Naturally, the ensemble model does not perform as well as the best model (linear ridge regression) since both regression models are very correlated.

# Appendix: package requirements

backports-abc==0.5

backports.functools-lru-cache==1.5

backports.shutil-get-terminal-size==1.0.0

cycler==0.10.0

decorator==4.3.0

enum34==1.1.6

functions==0.7.0

futures==3.2.0

ipykernel==4.8.2

ipython==5.7.0

ipython-genutils==0.2.0

jupyter-client==5.2.3

jupyter-core==4.4.0

kiwisolver==1.0.1

matplotlib==2.2.2

numpy==1.14.5

pandas==0.23.1

pathlib2==2.3.2

patsy==0.5.0

pexpect==4.6.0

pickleshare==0.7.4

pkg-resources==0.0.0

prompt-toolkit==1.0.15

ptyprocess==0.5.2

Pygments==2.2.0

pyparsing==2.2.0

python-dateutil==2.7.3

pytz==2018.4

pyzmq==17.0.0

scandir==1.7

scikit-learn==0.19.1

scipy==1.1.0

seaborn==0.8.1

simplegeneric==0.8.1

singledispatch==3.4.0.3

six==1.11.0

sklearn==0.0

statsmodels==0.9.0

subprocess32==3.5.2

tornado==5.0.2

traitlets==4.3.2

wcwidth==0.1.7