Construction of a Data Science First Investment Pipeline - Comparison and Evaluation of Data Mining Models on Financial Data

Austin Watkins, Carlos Jimenez

Abstract—*The increase of financial data provides an opportunity to apply data mining models. Finance is a rich area of research and several models have been developed in the last few years that use data science principles. In this paper we explore and provide comparison between these different strategies. We have performed analysis on our models using data from the Deutsche Börse Group. Our approaches generally showed good results on our evaluation set. We have evaluated our data on late 2018 and early 2019. Although, this span of time has shown high variance we have seen success with our strategies.*

1 Introduction

The prediction of stock market prices is, to say the least, known to be difficult. Regardless of its notoriety we find the topic rich with potential applications of data-mining. Do there exist models that predict the momentum of a stock from day to day with high probability? Is it possible to determine signals on when to buy or sell a security? Is there an algorithm for building a diverse portfolio? Our interest is in building an investment pipeline that uses data mining principles from the ground up. That is, what positions would our models recommend we take if we chose to apply techniques from data mining. Using these methods to choose a portfolio and determining when to buy or sell based on price prediction or signal processing.

April 15, 2019

1.1 Finance

The financial services industry is a large and continuously growing industry. There are, broadly speaking, three schools of thought regarding financial data analysis and prediction. Qualitative analysis; choosing securities based on the underlying properties of particular companies or markets. Technical analysis; using past market data to identify idiosyncrasies in securities over time. Quantitative analysis; employing a rigorous statistical approach to the study of risk of securities and positions, and how to optimally distribute this risk. Although these are general definitions there is significant overlap and any particular strategy may be utilized by more than one school.

Technological applications to finance have become popular and entertain a lot of attention. Yet, we believe one should be skeptical of promises made by many “fintech” companies and firms. We should not expect data science approaches to bring us riches through stock prediction. Nevertheless, investment opportunities exist because the market is not perfectly efficient. Although the search for market irregularities is perhaps has become more difficult due to the proliferation of market data and high speed trading, if an investor is searching for arbitrage opportunities we believe a more mathematical and scientific approach is optimal. This naturally leads us to our interest in a data science first approach to investment.

Through our research we have found that there are many data-mining applications in finance and it is a active area of research. Due to the potential rewards and traditions, one would be hard pressed to find a more data-intensive industry than finance.

Of course financial analysis is not without difficulty. First, the efficient market hypothesis, which is widely referred to by researchers in the field, states that any potential insights gleaned from data is already realized by the price of a security. (i.e. the price of the security incorporates all the available information) Second, many of the powerful models that are used in mathematics and data science must assume that the data is IID, yet this is a patently unrealistic assumption when it comes to the market. After all, the market is a complex system with many factors and agents acting with interests than we could ever expect to capture in our model. Additionally, although there are copious amounts of data in finance, the ever-changing macro economic features of the economy slowly render our models antiquated over time. A great number of recent financial data was developed in an economy where interest rates were among historic lows, yet this is likely to change. The can lead towards training models on data from a fundamentally different economy than the one that model is meant to be utilized on. Thus, financial modeling and analysis is a notorious moving target. Finally, a time series of stocks has been shown to resemble random movements. Mandelbrot and Taylor in 1967 shown that much of equity prices can be modeled as the prior price plus noise from a Gaussian distribution. This makes modeling stock data inherently pathological [1].

1.2 Contribution

Our original hypothesis was that we could remove a significant amount of the noise by building features based on volatility measures such as EMA and SMA. We were considering if regression on this feature vector would reveal something about the underlying stock. Through testing we concluded our hypothesis was wrong and we wanted to know why. To compare with our original model we searched for strategies in the literature that we could use to apply our features to. The models we tested on were LSTM and an AR model, these two failed to find any meaning or pattern in the underlying data as well, suggesting that these features may not really incorporate very useful information about the underlying security. That is, we found that predictive analysis on normalized returns to be totally ineffective. All of these models, when trained on our feature vector, produced naive and uninteresting results, such as a constant output

Disappointed, we then explored to see if we could implement better models than the one we first formed. Building a gamut of tools and algorithms for the analysis of financial data, we formed an investment pipeline that used our data mining techniques from the beginning and performs the core equity trading tasks. From this we have implemented and compared a signal processing system that heavily uses regression, a clustering algorithm for portfolio selection, and a deep learning model, and autoregressie model for price prediction. These models perform discrete tasks, thus, in the interest of fair comparison between models we built a simulated market that contains training data to test the models on, as if it were streaming live. Evaluation of these models is performed by iterating over the trading data in intervals, giving each model the same interface and potential opportunities. We made this implementation decision to best simulate the real world application of streaming the live trading data to these models. The ultimate comparison on how well a financial predictive model works with a given strategy is backtesting it to see how well it performs. Thus, each model is compared on the return of its strategy over time span of the simulated market.

2 Related Work

A book that heavily influenced our approach is Advances in Financial Machine Learning by Marcos Lopez de Prado. We paid particular attention to the part about cross-validation and hyper-parameter tuning. Prado argues that the common methods used within traditional data science fail for financial applications.

It was Prado who convinced us that we should be concerned about deceptive results due to information sharing. This caught our attention because of our use of moving averages as our features. That is we knew that we would have information shared between our training set and our evaluation set. Prado advised a practice of ”purging” which removes the data between these two sets to remove unwanted ”cheating” by the model. What we did not foresee is that this was a largely useless endeavor early on due to the fact that our features were using a time series of returns from stocks to build our features. As returns are normalized it makes it impossible to overfit. No matter wether we purged or not, our model tended to make a constant zero prediction.

Nevertheless, purging held to be an generally effective strategy for our LSTM model, as we noticed that performance was unusually good near the start of the training period. We found the model performed slightly worse when we purged data, but perhaps, it was more general. Though we are not sure whether this dip in performance is due to training on less data or because we are removing information sharing. A caveat, however, is if the data between the training set and the evaluation set tended to be highly idiosyncratic we found the model did perform better.

3 Methods

3.1 Dataset

Our data is from the Deutsche Börse Group's public data set Xetra. The Deutsche Börse Group generously provides free data via their API, to which we have obtained a key by registering on their website. The API provides historical data at all times, and provides real time data when trading is ongoing.

We have a large amount of data. We have collected and stored data on 29 companies. Adidas, Allianz, BASF, Bayer, Beiersdorf, BMW, Continental, Covestro, Daimler, Deutsche Bank, Deutsche Börse, Deutsche Lufthansa, Deutsche Post, Deutsche Telekom, E.ON, Fresenius, Fresenius Medical Care, HeidelbergCement, Henkel, Infineon Technologies, Merck, Munich Re, RWE, SAP, Siemens, ThyssenKrupp, Volkswagen Group, Vonovia, and Wirecard.

A single data entry for a company is indexed by the day and the minute containing the price of the stock and the volume of trade. We have stored our data in files organized by company and day. Our data starts on July 3rd, 2017 and ends April 14, 2019. In total we have $12,774$ files. There are approximately 515 minutes in a trading day although not all stocks are traded every minute. After loading the entirety of the data into our Pandas DataFrame we have $6,657,690$ points of data (minute, stock, and price). We can then process this data into larger-grain intervals. In addition to the API data, we construct a pseudo DAX index that we calculate for each minute a trade happens. The traditional DAX index does not have an equal distribution of each stock. As this distribution is periodically readjusted we decided to form our own index that has an uniform distribution of weights between each company.

We predicted early on that we were going to have a large amount of data. Thus, we focused on building a collection of tools for data cleaning, pre-processing, and building features, before evaluating our models. For example, We built some unix scripts that query the API in an intuitive way allowing us to specify a company and a range of dates for easy, uniform gathering of data. We found that our collection of tools allowed us to utilize composition to quickly move from forming a hypothesis to experimentation. Overall, the forethought and investment payed off later on.

3.2 Models

3.2.1 EMA/SMA

The intuition behind the EMA SMA volatility features is that momentum might be detectable, when a sharp inversion occurs. One would hope to detect upwards or downwards momentum of a stock price by realizing that recent returns or prices have diverged from what might be expected.
For a simple linear model of this kind we get the following:

$$Y_t = \beta_0 + \beta_1 E_{t-1}^{w} + \beta_2 S_{t-1}^{w} + \varepsilon_t r$$

Where E_{t-w} and S_{t-w} refer to an exponentially weighted and simple moving average respectively at time $t-1$ over a prior window of size w. After analyzing the model's output, we realized that our model was fitting near zero coefficients that made this experiment less interesting.

3.2.2 AR(p)

The autoregressive (AR) model is particularly intuitive. The model revolves around the idea that the next price in a sequence depends linearly on it's previous values. This is a natural assumption of course, and it may work for processes that exhibit this behavior. Specifically, if we are using an AR(p) model, we are finding relations between X_t and the last p values for X.

$$X_t = \beta_0 + \beta_1 X_{t-1} + ... + \beta_p X_{t-p} + \varepsilon_t$$

In our case, we use a ridge regression model to estimate the parameters β_i based on a training set. For each stock, we fit coefficients first and use those for the entirety of the testing data.

3.2.3 LSTM (return price)

Long Short-Term Memory (LSTM) models are a popular approach to predicting sequences and time series [2]. We primarily attracted to this model to see whether a deep learning approach would have dramatically different results compared to our other models. It was a very interesting process implementing it via Keras on a Tensorflow backend. LSTMs are essentially a kind of Recurrent Neural Network, but they are especially well suited to handle sequential data.
We essentially have two implementations; one for evaluation of a single stock, and the other for evaluation of the entire market (29 stocks + pseudo DAX).

Both stocks, however, use the ADAM optimizer, which is an efficient, light, first order, gradient based, stochastic optimizer method, with an MSE loss function [2].

For the single-stock implementation, we have the input

$$x_t = \begin{bmatrix} p_{t-1} \\ p_{t-2} \\ \vdots \\ p_{t-p} \end{bmatrix}$$

And the expected output $y_t = p_t$. Every value for p_i is scaled to (-1, 1) before training and testing, in order to preserve some value neutrality. We are then, attempting to predict p_t given a series of these sequences.

This is then composed with three hidden layers h_i, and trained over the single stock's training data. Using various hyper parameters.

Additionally, our other model which tracks all 30 stocks simultaneously has the following input feature;

$$x_t = \begin{bmatrix} p^0_{t-1} & p^2_{t-1} & \cdots & p^{29}_{t-1} \\ p^0_{t-2} & p^2_{t-2} & & \\ \vdots & & \ddots & \\ p^0_{t-p} & & & p^{29}_{t-p} \end{bmatrix}$$

which tracks an output

$$y_t = \begin{bmatrix} p^0_t \\ p^1_t \\ \vdots \\ p^{29}_t \end{bmatrix}$$

corresponding to the prediction of all 30 tracked stocks in the market. This model also has 3 hidden layers h_i, and scales all input to (-1, 1). The result of this scaling is problematic, however, as the final output seems to be inferior to the single-stock network. We do believe that an LSTM model could improve effectiveness, if every datapoint corresponding to a particular stock was scaled independently.

3.2.4 Statistical Arbitrage

Arbitrage in the U.S. Equities Market [6] by Marco Avellaneda and Jeong-Hyun Lee introduces a model of generating trading signals using either PCA or by comparing a stock with other securities that believed to correlated. This second model is what we implemented. The relationship between two securities can be described by,

$$\frac{dP_t}{P_t} = \alpha dt + \beta \frac{dQ_t}{Q_t} + dX_t$$

where X_t is mean-reverting and P and Q are stocks. For our purposes we will assume that our pseudo DAX index is correlated with the stocks within the index.

From this theoretical foundation Lee and Avellaneda give a recommended signal for when we should buy the index and short the stock or short the stock and buy the index. To form this signal the authors give methods to approximate the needed variables. The above equation becomes,

$$R^S_n = \beta_0 + \beta R^I_n + \epsilon_n$$

where n ranges over the returns in the training set.

We performed regression to get β_0 and β then found the residuals. X_n is going to form a cumulative sum of the residuals, that is, $X_k = \sum_{j=1}^{k} \epsilon_j$ where k iterates over the residuals. We then perform an auto regression analysis by shifting every X_n and comparing this series with its unshifted self. By,

$$X_{n+1} = a + bX_n + \zeta_{n+1}$$

After performing this regression to find a and b, the line of best fit, we found the residuals ζ. Finally, the signal s, is found by,

$$s = \frac{X(t) - m}{\sigma_{eq}}$$

Where, $m = \frac{a}{1-b}$ and $\sigma_{eq} = \sqrt{\frac{Variance(\zeta)}{1-b^2}}$, and $X(t)$ is the residual for the line given by β_0 and β, that is, $R^S_t - \beta_0 - \beta R^I_t$.

We implemented this to form signals on all 29 stocks simultaneously for every duration under consideration in the simulation of our market. For example, if we set the duration as 15 minutes we would form a signal every 15 minutes. That is performing linear regression 58 times (2 for every company) every 15 minutes. We would then execute a number of buys or trades based the signals. The policy that we used is that if the signal was greater than 1.25 we sell the stock and buy the index. Otherwise, if the index is less than -1.25 we buy the stock and sell the index.

As this strategy is attempting to take advantage of mean-reversion we desire to have our overall market position be market neutral. Lee and Avellaneda point out that this can be achieved by always buying or shorting β's worth of the index for every unit of currency purchased or shorted of the stock. We followed this advice with our implementation, although we note that it would be desirable to have a portfolio that is impervious to market fluctuations we note in reality our position is probably not market neutral. Both because we are assuming a simple relationship between two securities and that β is an approximation. The evaluation of this model can be found in the results.

The intuition on why this model might work is because it is possible that two securities are being affected by the same factors. For example, if the return of a particular agricultural company becomes significantly lower than a collection of other agricultural companies we suppose that the company is undervalued. Given enough time we expect the company to revert to the mean.

3.2.5 Clustering

A key part of finance is portfolio selection. Marcos Lopez de Prado in Building Diversified Portfolios that outperform out-of-sample introduces Hierarchical Risk Parity [5]. This portfolio selection mechanism is mathematical and data science approach for diversification. Prado claims that it solves several major issues with other portfolio selection models. For example, traditional Markowitz based models of portfolio selection there is the problem that as the number of stocks you analyze increases the more unstable the results are. This is due to the fact that the condition number of the covariance matrix increases as we increase the number of stocks. Where the covariance is the distance between the smallest eigenvalue and the largest. This is highly undesirable as we believe that a desired expected return should lead to a portfolio that is only slightly different if the expected return changes.

From an intuitive perspective we should expect that diversification is achieved if we select securities that are mathematically different. Thus, an algorithm that forms a covariance matrix, to define dissimilarity of particular stocks and results in a hierarchical solution we believe is a good starting point for portfolio selection. As this is a portfolio selection technique the evaluation is straightforward particularly since we found an excellent tool by Robert Martin does Hierarchical Risk Parity [7].

4 Results

4.1 Evaluation Metrics

4.1.1 Avg. MSE

The Avg. MSE, clearly the average Mean Squared Error for the model, which nearly all of our models use in some way.

$$\frac{1}{n}\sum_{i=0}^{n}(Y_i - \hat{Y}_i)^2$$

This is computed for each prediction of every stock, and then averaged together to determine the model's score.

4.1.2 R^2

The coefficient of determination is a common evaluation metric for regression problems. Here we present the normal formulation as the square of the coefficient of correlation.

$$R^2 = r^2$$

4.1.3 Sign Acc.

Sign accuracy here refers to the sign agreement between our prediction's implied returns (the percent difference between a prediction and it's predecessor) and the true returns.

$$1 \implies \text{sign}(R_t) = \text{sign}(\hat{R_t})$$

$$0 \implies \text{sign}(R_t) \neq \text{sign}(\hat{R_t})$$

The measure here then gives the average value for this sign accuracy of a model over all of its predictions.

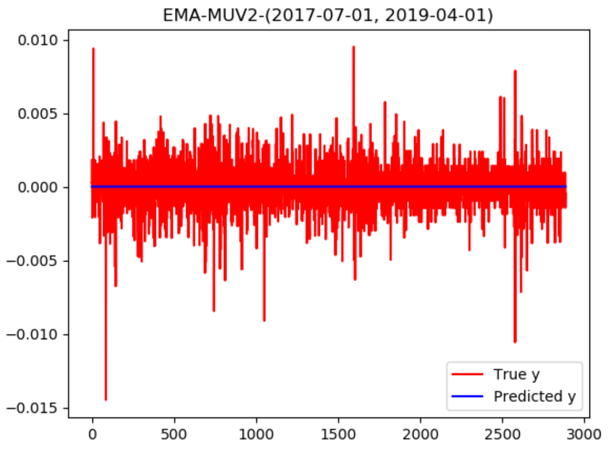
4.1.4 Simulated Market Return

For models that lend themselves to this evaluation, we implement a simple investment strategy to test their return estimation.
We begin with one dollar in cash, and for every next interval in a market simulation we move all of our investment to the stock for which our model predicted the greatest return. The end evaluation depends on our final simulated market return using this simple, if naive strategy.

4.2 EMA/SMA

Avg. MSE	R^2	Sign Acc.
0.0000	-0.0003	0.4713

The time interval is 15 minutes. Using a Ridge regression model with cross validation, and varied alphas it's performance is no better than guess work.
In fact it's predictions are usually constant zero.

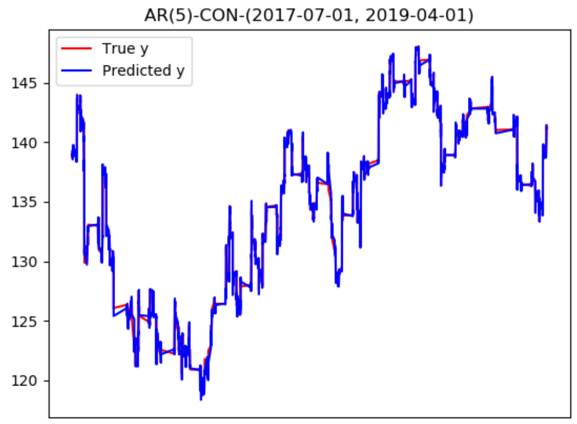


■

4.3 AR

Avg. MSE	R^2	Sign Acc.
0.0857	0.9963	0.5629

The time interval is 15 minutes.
We train on 0.75 of the data, purge 0.05 and test on 0.2.
We use an AR(5) model with ridge regression to estimate the appropriate coefficients. Interestingly the coefficients are not simple naive coefficients (return the last value) but it is not exactly clear what kind of information it encodes. We expected the R^2 and MSE to be fairly small, which they are, but we are pleased to see that the sign agreement is also more than just guesswork here. It would seem that the model has some predictive value. Additionally, it's predictions graphed look more impressive.



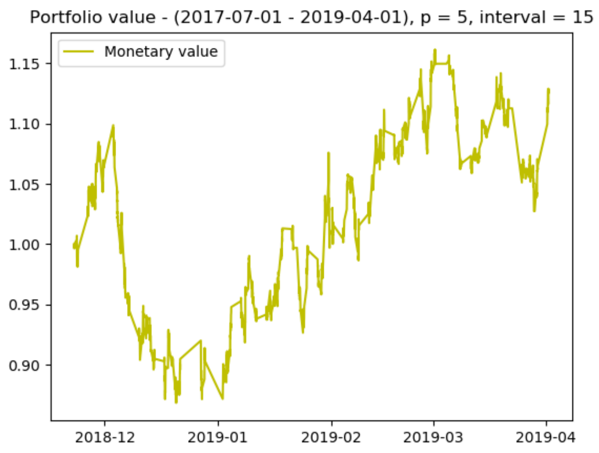
■

4.4 LSTM (price)

Avg. MSE	R^2	Sign Acc.
0.8341	0.9876	0.5105

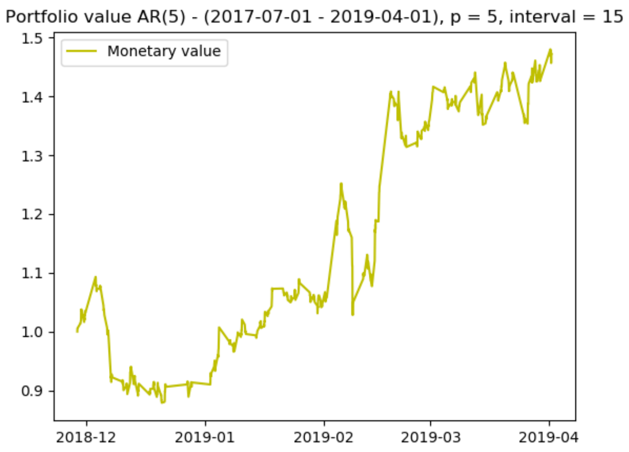
We train on 0.75 of the data, purge 0.05 and test on 0.2.
Here we are running an LSTM model for lagged values with a sequence window of 5, and time interval of 15 minutes.
Additionally, we have 3 hidden layers of 60 units and 10 epochs.
Although the evaluation metrics suggest that perhaps the LSTM model performed more poorly than the AR(5) model, one would not be able to tell this from simply looking at the models' predictions. We were surprised that the model's performance lagged behind the simpler (and much faster) AR(5) model. Additionally, it would not seem that it is making intelligent predictions on the directions of the models either, which is most significant.

Multi-stock model Evaluation
We train on 0.75 of the data, purge 0.05 and test on 0.2.
We evaluate the multi-stock model for LSTM which produced a surprising **12.78% simulated market return**. (surprising because our pseudo DAX index in general fell about 1 percent over this period)



■

Additionally we evaluate a multi-stock model for AR(5) seen below, and it produced a very impressive **47.25% simulated market return**



■

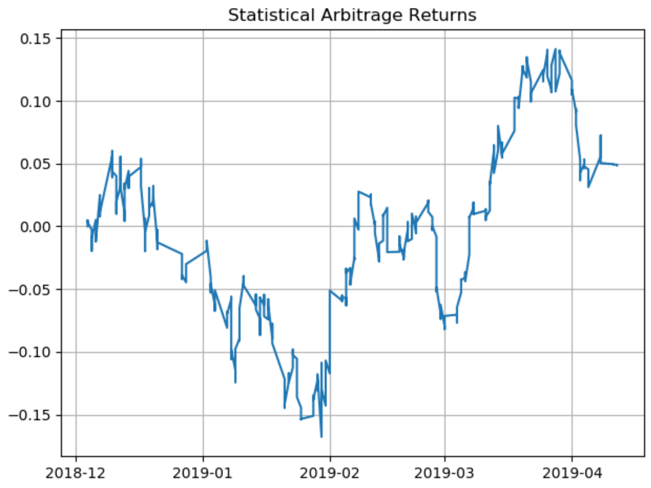
Despite the impressive returns here, the simplistic and naivete of the trading strategy is far too risky to implement in reality. This implementation was simply a experiment to see how these models might be used in a real world situation. We must have gotten lucky.

4.5 Statistical Arbitrage

This method of statistical arbitrage applied to our infrastructure for the 29 stocks lead to a high number of perceived trading opportunities. To experiment with this model we averaged prices of stocks over 60 minutes. This means that for when the market is open we consider trading every hour. Even with this particularly conservative strategy we ended up trading 6,087 times from mid December 2018 to current day.

Shown below is a graph of the return of this method over our evaluation evaluation set. The evaluation set is 20 percent of our data. As we can see it is highly volatile. The worst performance being in late January where we lost 15% of our investment. This is certainly significant. Yet this compares well with our best performance at the end of march when we are in a position of having 15% return on our investment. When we exited our position we had a 5% return on our investment.

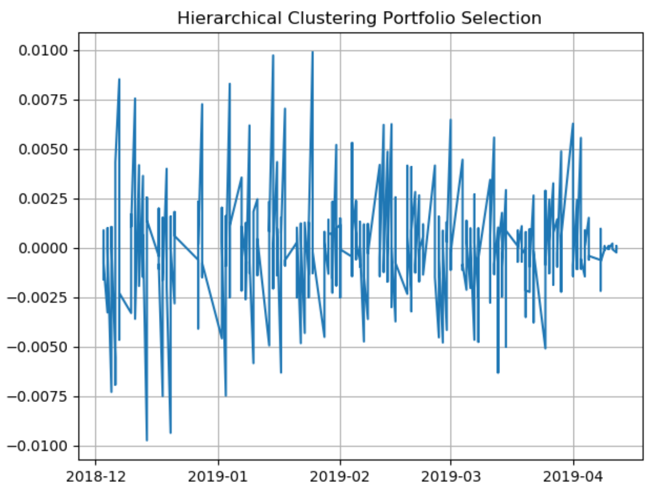
We noted that this model has difficulty scaling to a large number of stocks. We ran into performance issues evaluating this particular model to 1 minute intervals. Naturally this means running linear regression 58 times per minute. With 20% of our data being used for evaluation this would imply running a simple linear regression model over 69 million times.



■

4.6 Clustering

Performing the Hierarchical Risk Parity algorithm on our data and received the weighted sum in the table in Appendix C. In our simulated market we took a position that followed this distribution. Looking at the graph for the return of our portfolio we see a lot of variance. We expected this due to the fact that we are investing into every stock and the pseudo index. We then exited our position based on the price at the end of our evaluation data. The, end result is a 2.6% return over roughly a 5 month span.



■

4.7 What We Proved

There are two things that we believed when starting the project that have been disproved by our experimental results.

First, our concern for purging data was unjustified for our original hypothesis. Purging, the act of removing data from the training set and the evaluation set, was not needed as our original proposed model failed to work regardless of purging or not. Yet, we are glad that we did study the potential of information sharing as it did help towards applying LSTM and getting good results from it.

Second, our features were wrong. At least in the sense that it is incredibly difficult to gleam information from a moving average.

Both of these hypothesis were shown to be false by a single experiment. That is when we performed predictive analysis on our our data the estimator, biased or not, always predicted the price remaining the same.

5 Conclusion

By demonstration we have shown that there is applications of data mining, at least in theory, to finance. In this project we have implemented tools for gathering data, cleaning that data, and forming them into features. Resulting in over 69 million points of data. We have tested our original hypothesis, that moving average features can be regressed upon for predictive analysis. We have shown that our hypothesis is false. We have built more successful models using regression both in the form of auto regression on prices and on getting signals from performing regression on our time series. We have tested a portfolio building a strategy based on hierarchical clustering. We implemented a LSTM for price prediction. We built a simulated market for evaluation, training our models, and to practice streaming over financial time series.

We have shown that our models perform well on the evaluation data. All of this is in spite of the market going down during our evaluation time frame. Thus any of our models that consistently gets positive results is beating the market. We have had each of our models perform this well.

Nevertheless, we do have concerns and there is limitations to our approach. First, we have not analyzed our models for numerical error. Something that we suspect is playing a role either positively or negatively. Second, although we have a lot of data, it spans only the last year and nine months. This is nothing compared to how long the market has been opened. Third, our evaluations were on German companies in 2018 and 2019. We wonder if our models lack generality in both time and with different companies.

Yet, we have learned much from our efforts, here are three lessons of interest. First, that data science work on financial data is often unintuitive and difficult. It is particularly pathological. Yet, in a way, this was good because it forced us to implement many things that were out of our grasp in skill level before we started the project. Second, it is difficult to perform data analysis without data. Thus, we have both learned that when starting a data mining project one of the first points of focus is on getting data with ease with an elegant interface then cleaning that data properly. Third, when evaluating our efforts over the past two months we have found that a large factor of our success is that we focused on tool creation for data processing and feature creation. This allowed us to quickly test a hypothesis via composition of our existing functions. In completing this project we have built a small library of different models and features for investment analysis.

In the end, through this project, our interest in data science and good engineering practices has increased.

References

[1] B, Mandelbrot and H. M. Taylor. *On the Distribution of Stock Price Differences.* Operations Research 15, no. 6 (1967): 1057-062. http://www.jstor.org/stable/168611.

[2] A. Navon and Y. Keller. *Financial Time Series Prediction Using Deep Learning*. (Submitted on 11 Nov 2017): arxiv:1711.04174

[3] M. Prado. *Advances in Financial Machine Learning*. New Jersey: Wiley, 2018.

[4] Siami-Namini, Sima, and Namin, Akbar Siami. *Forecasting Economics and Financial Time Series*.

[5] M. Prado, *Building Diversified Portfolios that Outperform Out of Sample*. The Journal of Portfolio Management May 2016, 42 (4) 59-69; DOI: 10.3905/jpm.2016.42.4.059

[6] M. Avellaneda and J. Lee. *Statistical Arbitrage in the U.S. Equities Market*. Quantitative Finance, 2008.

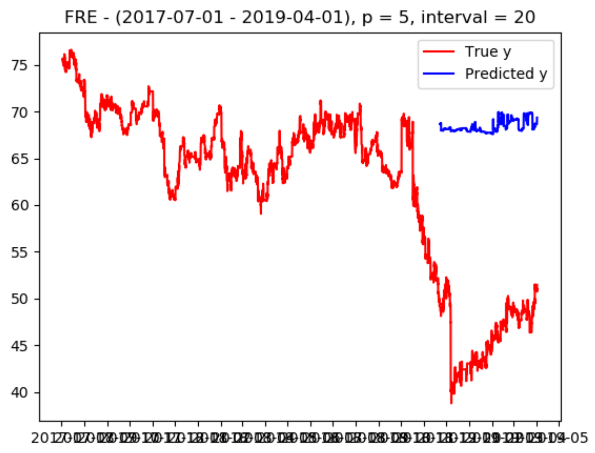
[7] R. Martin, *PyPortfolioOpt*, (2019), GitHub repository,https://github.com/robertmartin8/PyPortfolioOpt

Appendix A
Distribution of work

- Austin Watkins
  - Hierarchical Risk Parity analysis
  - Statistical Arbitrage implementation and analysis
  - Sign ratio function implementation
  - Gathering data and the managment of that data
  - ARMA and ARIMA exploration and implementation that was not ultimately put in our report.
- Carlos Jimenez
  - LSTM implementation and analysis
  - AR implementation and analysis
  - Evaluation metric generation and their uses in evaluating models
  - Download tool for gathering of data
  - Moving average feature tools for both SMA and EMA.
  - Market simulation system and class.

Appendix B
Graphs of LSTM

Graphs from LSTM
A most egregious pitfall from LSTM multi-stock model



■

Appendix C
Distribution Investments Via Hierarchical Risk Parity

1COV	0.0018855540260202571
ADS	0.001387858910081707
ALV	0.003821999979808915
BAS	0.012144548535399051
BAYN	0.0016395636994317895
BEI	0.0035160114709959903
BMW	0.007347055719092772
CON	0.0008671497044404641
DAI	0.0021038738346754013
DB1	0.001138455890756179
DBK	0.0012391046685170932
DPW	0.0013827106719862799
DTE	0.0005512384541618031
EOAN	0.01219088928150861
FME	0.007972269528673873
FRE	0.0012875198153261013
HEI	0.0022943709085333527
HEN3	0.0008559023640377087
IFX	0.0018679938738092009
DAX	0.8911760155550439
LHA	0.0009267842845526943
MRK	0.005042183500713999
MUV2	0.002323837185329553
RWE	0.0028937017690506897
SAP	0.004396652518503947
SIE	0.0008855161620366333
TKA	0.0016728596127915008
VNA	0.01662498062434953
VOW3	0.0074318841697170596
WDI	0.0011315132806540527