Imperial College London

Department of Electrical and Electronic Engineering

Final Year Project Interim Report 2019

Project Title:	**Feature Engineering for Fundamental Factor Models**
Student:	**Tim Hung Wu**
CID:	**01195701**
Course:	**EIE3**
Project Supervisor:	**Dr. Carlo Ciliberto**
Second Marker:	**Professor Yiannis Demiris**

Abstract

This project explores the effectiveness of recurrent neural network architectures (RNN) for feature identification within a set of companies that form part of the STOXX Europe 600 Index. In particular, the focus will be on long short-term memory (LSTM) variants which are often used in sequential data applications that require learning of long term dependencies. The objective is to develop a model that will analyse a company's fundamental features and identify the underlying drivers that cause price movements in a stock. The model will then be tested against an industry benchmark heuristic to see if any outperformance can be managed. The fundamentals data is supplied by an external long/short equities hedge fund. The various processes required to develop a solution to this machine learning problem such as data preprocessing, network structuring, optimisation etc. will be studied and analysed.

Acknowledgements

A special thanks goes to Dr. Carlo Ciliberto of Imperial College London for providing expertise about developing and designing a machine learning experiment and Dr. Marco Bianchi and Michele Ragazzi of Giano Capital for providing the financial background knowledge to help bring this project to a real world application.

Contents

1. Project Specification

1.1. Project Motivation

Public corporations are required to periodically report various fundamental financial data such as income statements and balance sheets. This is legally required in most countries for tax purposes and also provides key performance indicators for stakeholders and investors into the general financial health of a corporation. The owners of these companies are a mixture of the general public and ownership is fractionally divided via shares of a company's stock. These shares are able to be freely bought and sold and the process of valuing them has become of predominant interest to investors. Consequently, the expectation of how these valuations will deviate and change in the future is interesting as it provides the opportunity for financial gain by taking the profit through a difference in price.

Academic research has demonstrated that some key factors of the reported financial data have historically had high correlation with the returns enjoyed by the stock of a company. An example of a popular methodology is analysing the asset turnover ratio ($sales/assets$), which approximately indicated the efficiency with which a company deploys its assets to generate sales.

The process of automating the identification of features which are able to deliver profitable investment opportunities falls into the domain of systematic factor investing. Increasingly, this has become more popularised because of the exponential technological improvements in data storage and computational power. Additionally, the digitalisation of data allows it to be more standardised and of higher quality, further lowering the limitations boundary for automation.

1.2. Problem Definition

Using the SXXPI Dataset supplied by Giano Capital, this project aims to establish the key fundamentals that affect the returns of a company's stock price. This is achieved by developing algorithms that can assist in predicting the future prices of the stock. The algorithms utilise several inputs such as book value, sales and profit of the specified company to train and learn a function that correlates with the price. The performance is then benchmarked against an industry standard to see if the algorithms have achieved any outperformance compared to traditional models. This problem falls in the scope of machine learning, which has been frequently used to produce solutions in data-driven problems in recent times. Focus will be directed at RNNs with LSTM units. We choose well established and tested techniques used in literature to extract features from the dataset and attempt to learn patterns in the data. We will then tune the hyperparameters to optimise the model, and evaluate its potential performance in a pseudo-generated real world application.

1.2.1 Intended Deliverables

- A cleaned dataset capable of being trained on.
- A piece of software that implements a predictive model.
- Analysis into the performance of the model.
- Understanding of prior research in the area and how to design a machine learning experiment.
- A report and poster summarising the project and its experimental results.

1.3. Importance in Industry

Outperformance, defined by generating excess returns above the market average, is termed as alpha (α) in the financial industry. This metric is widely used to assess and compare the abilities of actively managed funds. The process from which alpha is generated is termed "signal generation" and determines when a security, such as a stock, should be bought or sold. However, as more people utilise a certain signal, it becomes weaker as it, by definition, tends towards becoming part of the market average. The objective then evolves to be consistently producing new, stronger signals. As mentioned in section 2.1, there are various methodologies people use to generate signals. Although there is no limit to how complex a signals can theoretically be, people tend to focus on just a handful of parameters for ease-of-use and understanding. This begs the

question of whether a machine learning model, with increased numbers of inputs, are able to assist in developing or even create complex methodologies that result in strong signals.

As such, this project is of interest to money managers as a means from which to develop alternative and new solutions to the problem of generating new signals to produce alpha. Additionally, the consequence of being able to produce improved returns are ultimately handed to the investors. This project focuses particularly on the fundamental components of companies, which tends to be in the scope of long term investment. The investors in these areas often include institutional clients such as pension funds meaning that these potential improvements are eventually passed down to working individuals, who do not possess the ability to generate this income on their own for their future retirements.

1.4. Software Usage

Python will be the language used for software development given the extensive number of libraries that offer the opportunity for the fast development required for the project. The key packages that will be used are Pandas and Keras, which are data manipulation and machine learning tools respectively. Correctness of the code is a necessity whereas performance is treated as a luxury. The report will be written in LaTeX.

All software development will be within the student's personal Google drive, with external view access granted to only those that are necessary. A web based environment, Jupyter Notebooks[1], will be used in order to develop a clear and structured front end document that can be easily used and interacted with by users. Consequently, the ipynb file format will be used for the source. Data is stored as csv files.

Cloud services will be used for the computational power and elements of data storage. Google Colaboratory offers good support for the Notebook and has previously been used as part of the EE3-25 Deep Learning module and so has a low barrier to entry. Given the fairly small size of the dataset (at approximately 300MB), the required computation power required is estimated to be well below the maximum threshold of the environment.

2. Background

2.1. History of Investing Methodologies

The fundamental question that the project is attempting to answer is what drives stock price and their underlying returns. This has been tackled through a whole range of varied approaches ever since shares of stock were first offered to general public in the early 1600s by the Dutch East India Company[18]. In 1934, Benjamin Graham et al. introduced the first formal framework for security analysis[8]. In his book, he describes the differences of market behaviour in the short and long run. Graham observed that fear, greed and emotions were the main drivers of short-term market fluctuations which can cause discrepancies between the pricing and true value of a company's stock. However, over long periods of time, Graham perceived that a company's fundamentals are the critical price driver, which causes the market pricing of the stock to converge with its true value. These observations formed one of the first formal distinctions between speculation and investment.

The first formal model[22] of analysing stock returns was introduced by Treynor et al. in the 1960s. It was called the Capital Asset Pricing Model (CAPM) and focused on the idiosyncratic and systematic risk of an asset relative to the market. The formula for pricing a security is as follow: $E(R_i) = R_f + \beta_i(E(R_m) - R_f)$ where $E(R_i)$ is the expected return of investment, R_f is the risk free rate, β_i is the beta of the investment, $E(R_m)$ is the expected return of the market. In other words, the expected return of a stock can be viewed as a function of its correlation to the market.

An alternative theory[17] about what drivers stock returns was later proposed by Ross in 1976. Arbitrage pricing theory (APT) postures that the expected return of a financial asset can be modelled as a function of various macroeconomic factors. Ross is often accredited with popularising the term "multi-factor models." Crucially, APT performed better than CAPM because it did not explicitly state a set of fixed factors. Instead, factors were assumed to vary across markets and differ over time. The challenge then boiled down to identifying the factors that best describe a specific region at a point in time.

Arguably, the most utilised factors for analysis are fundamental factors, which capture the characteristics of a stock. One of the most well known attempts in this space was derived from Fama and French in the early 1990s. Their model[7] explained US equity market returns from three factors: size, value and correlation as mentioned in the CAPM model. The fundamental objective of factor investing is to achieve enhanced diversification, above-market returns and reduced risk exposure. The

numbers of quantifiable factors are vast but typically, are broken down into five predominant categories:

- Value: The stocks usually trade at low prices relative to their fundamental indicators. These are identified via indicators which are typically ratios such as $\frac{bookvalue}{price}$, $\frac{earnings}{price}$ or $\frac{dividend}{price}$. When these ratios are significantly higher than the average company within the same sector, the company can be classified as having value stock. It should be noted that average ratios vary between sectors and so cross-industry comparisons cannot be made.
- Size: Eun et al. demonstrate in their research[6] that small capitalisation stocks tend to exhibit statistically significant returns and diversification than large capitalisation stocks when allocated in international funds. Capitalisation is defined as $capitalisation = no.share * shareprice$. Small-caps are defined to range from valuations of $300 million to $2 billion.
- Momentum: Stocks that have exhibited outperformance in the past tend to follow the same trend in the future. Strategies involving momentum often involve analysing, ranking and allocating to the highest returning stocks within a specified time frame such as one year or three years.
- Quality: These stocks are typically categorised by certain fundamentals such as low debt, average earnings with low standard deviation i.e. high stability and consistent asset growth i.e. the derivative of book value with respect to time is fairly uniform. Common financial metrics used to help identify this include the return on equity ($ROE = \frac{NetIncome}{ShareholderEquity}$) or the debt-to-equity ratio ($\frac{D}{E} = \frac{TotalLiabilities}{TotalShareholderEquity}$.)
- Volatility: Empirical research by UBS[13] suggests that low volatility stocks earn greater risk-adjusted returns than other stocks, with particular outperformance during stressed market environments. These are often ranked and selected by measuring the standard deviation of a stock's return.

2.2. Long Short-Term Memory Networks

2.2.1 Overview

LSTM networks are derived from RNN models, which were first introduced into literature[19] in 1986 by Rumelhart et al. Rumelhart was a psychologist by training and the initial objective was to provide an alternative framework for understanding cognitive perception. The fundamental problem was to highlight that human perception and learning were formed through temporal experience and so a model was required that could process and partially remember sequential data. Rumelhart mathematically modelled this as $h_t = f_W(h_{t-1}, xt)$ where h is the state, x is the input with respect to the time interval t and f is a function generated by the parameters W as determined by the configuration of the LSTM cell. This was in contrast with more traditional neural networks, such as the multilayer perceptron (MLP), are structured in a feedforward fashion that do not share weights, meaning that there are no cycles formed within the nodes.

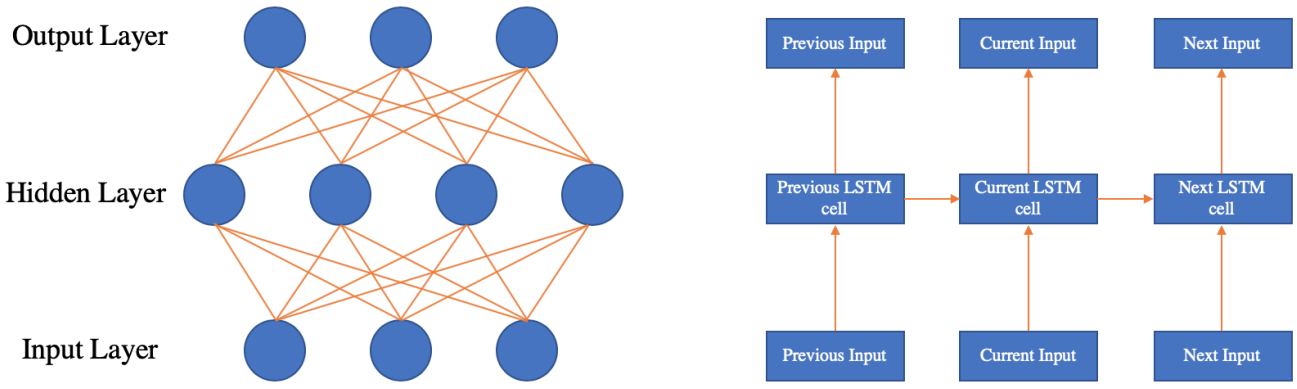

Figure 1. A classical MLP (LHS) simply generates an output with respect to its input whereas in the RNN (RHS), the output is governed by the input and previous input.

The use cases of RNNs have since expanded to multiple, practical domains that heavily use sequential data, such as text generation, visual recognition and time series prediction. Part of the advantages offered by RNNs is the various topologies in which it can be structured. The most vanilla case is with a direct one-to-one mapping. However they often change to many-to-one, many-to-many or one-to-many to suit the task at hand.

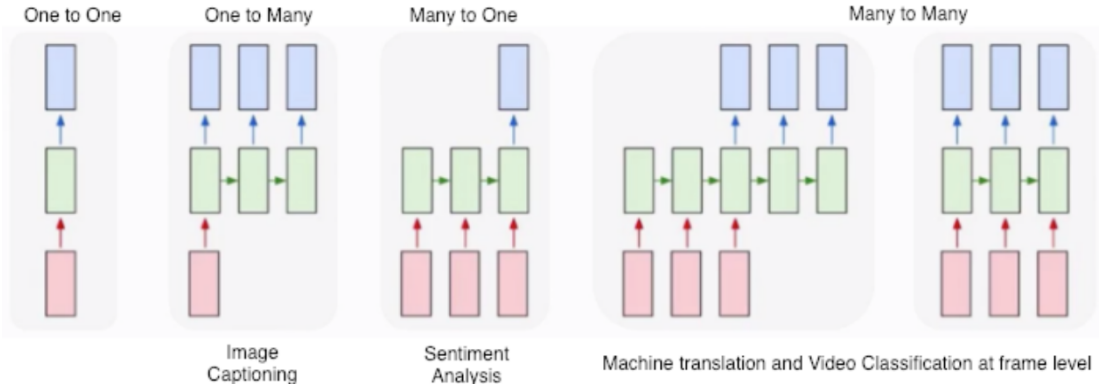

Figure 2. Numerous varying topologies for specific sequential data tasks.

An issue with RNNs stem from the inherent property of operating with a sequential nature. The time difference between multiple cell often causes issues during backpropogation involving repeated multiplication of the gradient signal. This means that if the partial derivatives of the error gradient are either too large or small, they can explode or vanish respectively during the procedure. Exploding gradients cause the learning to diverge whereas vanishing ones can cause the learning rate to become very slow and eventually stop, as it tends to 0.

Proof. Let x, y, h, θ, f be the input, output, hidden state, weights and basis function respectively: (Bengio et al.)

$$h_t = \theta f(h_{t-1} + \theta_x x_t)$$
$$y_t = \theta_y f(h_t)$$

The loss function is then:

$$\frac{\partial E}{\partial \theta} = \sum_{t=1}^{S} \frac{\partial E_t}{\partial \theta}$$

$$\frac{\partial E_t}{\partial \theta} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial \theta} \qquad \text{(where } k \text{ are the layers before time } t\text{)}$$

The third term of the chain, $\frac{\partial h_t}{\partial h_k}$, is a product of Jacobians such that:

$$\frac{\partial h_t}{\partial h_k} = \prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} = \prod_{i=k+1}^{t} \theta^T diag(f'(h_{i-1}))$$

$$||\frac{\partial h_i}{\partial h_{i-1}}|| \leq ||\theta^T|| ||diag(f'(h_{i-1}))|| \leq \gamma_\theta \gamma_f \qquad \text{(where } \gamma \text{ are the constants that upper bound the two terms)}$$

$$||\frac{\partial h_i}{\partial h_{i-1}}|| \leq (\gamma_\theta \gamma_f)^{t-k}$$

This implies that if $\gamma_\theta \gamma_f > 1$ or very small, the gradient will explode or vanish given a large $t - k$. LSTMs were first introduced by Hockreiter et al.[11] in 1997 and were effective at correcting the vanishing gradient problem. LSTMs use gating, also known as component-wise multiplication, which allows for more control over the gradient flow and thus enables better preservation of long term dependencies. Three gates exist in the LSTM cell forget, input and update, which are mathematically defined as following:

$$\text{Forget gate} : f_t = \sigma(W_f(h_{t-1}, x_t) + b_f)$$
$$\text{Input gate} : i_t = \sigma(W_i(h_{t-1}, x_t) + b_i)$$
$$\text{Update gate} : u_t = \sigma(W_u(h_{t-1}, x_t) + b_u)$$

As explained by Olah[14], the forget gate controls the fraction of how much of the previous cell's information should be retained. Next, a decision needs to be made about what new information should be stored in the current cell. This is achieved by the input gate. Additional to this, a candidate cell, $\tilde{C}$, is produced and scaled appropriately with a $tanh$ layer. By combining these two it is possible to create an update to the state.

$$\tilde{C}_t = tanh(W_C(h_{t-1}, x_t) + b_C)$$
$$C_t = (f_t \times C_{t-1}) + (i_t \times \tilde{C}_t)$$

The block output will then be a filtered version of our newly updated cell state, with the filter being the update gate. The sigmoid layer in the update gate decides what parts of the cell state will be output.

$$y_t = u_t \times tanh(C_t)$$

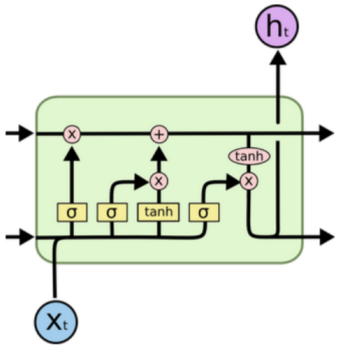

Figure 3. Breakdown of a LSTM cell depicting the input, forget and update gates.

2.2.2 Cross Validation

An issue with LSTM networks arises when using classical cross validation methods. K-fold cross validation involves dividing the data into k folds before assigning the initial fold as the test data and the rest as the training data. After each iteration of training, the test data assignment is then incremented until its position has permeated through all of the positions. However, with sequential data, this method causes disjoints in the training data since training for fold $n+1$ will be involve information from $n-1$ rather than n. In 2015, Bergmeir et al. introduced a new method[4] to handle cross validation for time-series data. Their system involved a forward chaining concept such that training data is never interrupted. Although it resolves the issue, the generalisation error is then affected by the change in training data size. Below is a heuristic for the method:

- Divide the data into k folds.
- 1st iteration: Training on fold 1. Testing on fold 2.
- 2nd iteration: Training on fold 1, 2. Testing on fold 3.
- Repeated to $k-1$th iteration: Training on fold 1, 2 ... $k-1$. Testing on fold k (last fold).

An alternative solution[15] was also proposed by Racine. He observed that what was truly needed is independence between the training and test data sets. This can be approximated by removing adjacent folds which will be highly dependent on and correlate with the test fold. This needs to be performed on both sides of the test fold as dependence is symmetric when moving forwards and backwards in time. This approach is known as hv cross validation where v is the test fold and h are the number of observations on each side of the test fold. Below is a heuristic of the method:

- Divide the data into k folds. Assume $h = 1$
- 1st iteration: Testing on fold 1. Fold 2 is discard. Training occurs on the rest of the folds.
- 2nd iteration: Training on fold 2. Folds 1 and 3 are discarded. Training occurs on the rest of the folds.
- 3rd iteration: Training on fold 3. Folds 2 and 4 are discarded. Training occurs on the rest of the folds.
- Repeated to kth iteration Testing on fold k (last fold). Fold $k - 1$ is discard. Training occurs on the rest of the folds.

This does not introduce the generalisation error as in Bergmeir's method. However, it is more complex to measure the h required to approximate a good independence and there is still a consistent inefficiency in data usage given the discarded folds are not trained or tested with. This efficiency is only better than Bergmeir when $h \leq 2 \times (\frac{k^2}{2}) = k^2$. Both methods will be used during the project to determine which one is optimal.

2.2.3 Applications

An exciting application of LSTM networks can be found in one of Google Deepmind's project called Deep Recurrent Attentive Writer (DRAW). In their paper[9], Gregor et al. outline the architecture for a network capable of image generation. They use a variational autoencoder (VAE) as the generative model. This encodes a raw image into a lower dimensionality vector, which forces the encoder to learn the features of the image. LSTMs and attention mechanisms are then used to iterate this encode-decode procedure. This network was used to generate digits after being trained on the Modified National Institute of Standards and Technology (MNIST) database.

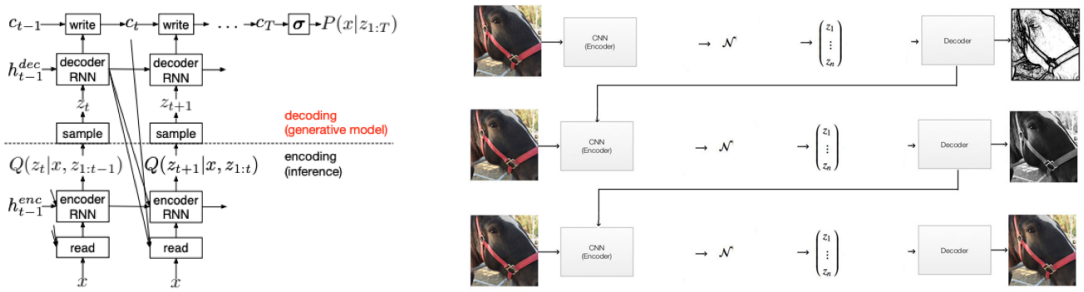

Figure 4. LHS: DRAW network from Google Deepmind, featuring an attention mechanism formed by an encoding and decoding RNN (with the LSTM architecture). RHS: Simulation of how each iteration improves on the previous when using the network.

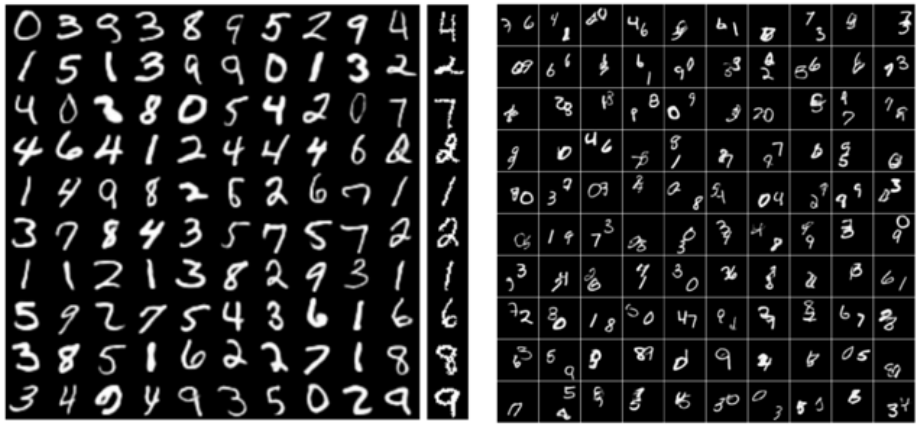

Figure 5. LHS: Single digits generated from the MNIST database, with the right-most column depicting original images from the database. RHS: DRAW's attempts at generating double digit values.

2.3. Review of Machine Learning Research in Investing

Machine learning research in finance applications often span two domains; applied research in financial firms often has a more pragmatic approach and a more theoretical approach is usually observed in academic institutions. Man Group, a quantitative investment management firm, have been conducting research[16] within the machine learning domain for decades but have only in the past five years begun actively applying models in a live environment. This is because there have only been enough developments recently in infrastructural support to help scale these models onto a production level. They separate their approaches into two major categories:

- Deep Learning: Models using artificial neural networks which are trained on large data sets to identify special features. The traditional application of these techniques stem from areas such as image recognition but can also be used to learn predictive patterns in financial datasets.
- Natural Language Processing: This mainly involves the automation of analysing written reports, such as a company's financial statements or a shareholder letter. Techniques involve assigning numerical values that measure the sentiment of these reports, which are often correlated with the price. This provides an unbiased and deterministic method for evaluating stocks.

The model used in the project falls under the deep learning categorisation and is specified in section 2.2. A few questions come to surface regarding the technical challenges of the implementation. Some are listed below and answered by various approaches recommended in numerous papers:

- Normalisation: Standardising the features of a model is often important because there is a need to compare measurements which have different units.
- Missing Inputs: A way of handling incomplete datasets is needed. Methodologies often fall into the two categories of either regenerating or masking the missing values.
- Encoding Additional Information: Adding human derived formulas such as $\frac{sales}{assets}$ can be seen as generating human bias or assisting the model with classification depending on perspective.
- Cross Validation (Section 2.2.2): A problem with time series structures is the autocorrelation of data points. This means that typically cross validation methods such as leave-one-out introduce issues by missing information in the series and leaking information from the future.

One of the earliest published papers[2] regarding the subject matter was written by Ahmadi in 1990. It was primarily focused on sourcing an alternate approach to generating a more comprehensive system to analyse factors in the APT model. This was done as following: Given a $N \times T$ matrix, $r_{N,T}$ is defined as the return of asset N at time period T. For this matrix, there is a smaller one of $K \times T$, where $f_{K,T}$ represents the Kth factor score for time period T. Thus, the return of a given asset is: $R_N = E_N + b_{N,1}F_1 ... + b_{N,K}F_K + e_N$ where E_N is the expected return, e_N is a random zero mean error term and $b_{N,K}$ are the estimated coefficients for each factor. Using the coefficient estimates, it is possible to employ another variable, λ, to determine whether it is statistically significant and thus whether the factor is significant. This is done with the following equation: $E_N = \lambda_0 + b_{N,1}\lambda_1 ... + b_{N,K}\lambda_K$.

The problem Ahmadi wanted to resolve was the heavy dependency on the sample that factor analysis relied on. There was a significant chance that, given 42 groups of 30 stocks each, different significant factors would be discovered in each group. He therefore employed a neural network so that, rather than optimising and identifying the statistically significance in each feature individually, the patterns stem from analysing the whole vector space. Ahmadi provides some details about the network, highlighting that the inputs used were the unemployment and inflation rates, rate of returns of assets, rate of change in the stock market and the change in gross national product (GNP). He also specifies that the inputs were normalised, two hidden layers offered the best optimal performance and the learning plateaued after 5000 iterations.

As machine learning research became more formalised, the structure of experimentation was more defined. Kim published a paper[12] in 2003, focused on understanding how support vector machines (SVMs) fared at predicting the Korea Composite Stock Price Index (KOSPI). Performance, P, was measured using the following equation:

$$P = \frac{1}{m}\sum_{i=1}^{m} R_i \qquad \text{(i = 1, 2, ... m)}$$

where R_i is the prediction result for the ith trading day as defined by:

$$R_i = \begin{cases} 1, & \text{if } f(x) = y \\ 0, & \text{otherwise} \end{cases}$$

where $f(x)$ is the prediction and y is the actual value.

He tested the structure with both the polynomial and Gaussian radial basis functions for the kernel. Most importantly, he benchmarked the performance against two other models: a three layered backpropogation network (BP) and case based reasoning model (CBR) using nearest neighbour. However, there are no specific details about the architectures of these models.

The best prediction performances of SVM, BP, and CBR (hit ratio: %)

	SVM	BP	CBR
Training data	64.7526	58.5217	
Holdout data	57.8313	54.7332	51.9793

Figure 6. Kim's results of SVM compared to BP and CBR.

An interesting alternative approach[3] is demonstrated by Alberg et al. who tackles the problem from a different perspective and adapts the classical problem from predicting prices to predicting how the fundamentals of a company will change. These predictions could then be used in turn to indirectly predict the price at that future time point. Alberg then backtested this idea, accessing existing fundamentals data as the 'future predicted fundamental', equating its price and evaluating the performance of the model. The paper goes into great depth about the set up of the experiment. A key area of interest is the methodology used to establish the binary output classifier:

$$f(x) = \begin{cases} 1, & \text{if } x > \frac{n+1}{n}th \text{ term} \\ -1, & \text{if } x < \frac{n+1}{n}th \text{ term} \end{cases}$$

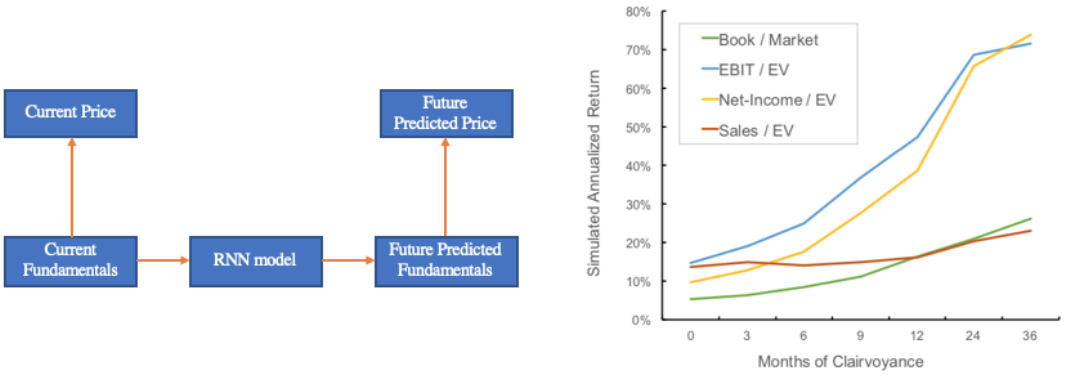

Figure 7. (LHS) The overview of how the model functions. (RHS) Backtesting results from Alberg with various factor models between 2000-2016. Clairvoyance x-axis equates to the time period difference between the current and future time in the model.

Rather than predicting the prices directly, which could rely on a multitude of factors outside just fundamental data, the model is structured so that it is interested in learning about what makes a company outperform the median return. A weakness

of this classifier is that, in economic downturns, the median return could still be negative and so the model is not learning what fundamentally drives 'good' returns in an absolute sense. It also does not predict the degree to which the chosen stocks will outperform. However, this is in interesting structure for this project given the relative nature of the evaluation metric selected as mentioned in section 4.

Additionally, normalisation of the inputs is carried out with two methods. The first involves ranking the performance of stocks against a certain ratio, such as $\frac{BookValue}{MarketCapitalisation}$. The percentiles of their rankings are then inputted alongside the raw scores in order to produce a more comparable set of features between various stocks. The second involves transforming all data points into a specified domain for ease-of-use to the learning algorithm whilst maintaining the same relative fundamental values between companies. Alberg achieves this by dividing the input feature by the L2 norm of all the features ($x_i \rightarrow \frac{x_i}{||x||}$). This is in contrast with two of the more popular methods of normalisation: z-score ($x_i \rightarrow \frac{x_i - \mu_x}{\sigma_x}$) where μ and σ are the mean and standard deviation and min-max scaling ($x_i \rightarrow \frac{x_i - x_{min}}{x_{max} - x_i}$).

Finally, in 2019, a summary of the literature[10] that had been written about financial prediction applications using machine learning was produced by Henrique et al. They constructed a table comprehensively detailing the various attributes and methods used in the papers they reviewed. This provides a very general overview into how the academic field is generally designing experiments for this problem domain.

- Assets Evaluated: Typically stocks or indices. These fall into the same region as the problem tackled by this project. Interestingly, many index models were focused on predicting the prices of the index itself whereas stocks typically offered more dynamic evaluation metrics such as selecting good stocks and calculating their returns.
- Predictive Variables: Predominately technical analysis or prices, with a few fundamentals and one text and news. It is understandable that most models which predict price use technical analysis indicators given that the perceived relationship between these is fairly strong. Technical analysis was designed for price analysis whereas fundamental data looks at the valuation of stocks, which is a slightly different task. However, this project focuses on a long term investment horizon and so is hoped that fundamental features will be more significant in this time domain. The text and news cases were used to analyse sentiment, which often affects price on a very short time horizon.
- Output Prediction: Mainly prices, return or direction, with a couple targeting volatility. Price and returns are the ideal goal but it is interesting to see that a significant number of models use alternate methodologies. Both direction and volatility offer a less demanding output objectives and so could achieve better results overall if trained correctly. This project will analyse a variety of output predictors and compare their performances.
- Methodology: Mostly neural networks, with some SVMs and fuzzy logic. This project uses a RNN with LSTM units and so falls under the set of neural networks. The objective is to explore how LSTM models fare in a different domain, given their primary applications being in image, audio and text generation/recognition.
- Performance Measure: A mixture of return, Sharpe ratio, mean absolute error (MAE) and mean squared error (MSE). MAE and MSE are generally used when classifying the test error of a price prediction. However, the project aims to deliver a real world grounding and so metrics such as Sharpe ratio will be used during the final evaluation. It is very likely that MAE and MSE will be analysed during the experimentation stages.

3. Implementation Plan

The preliminary breakdown of the work expected to be required and achieved through the project are listed in the below Gantt charts. Thus far, focus has mainly been on building knowledge and theory in order to develop a viable solution to the problem. After the interim deadline report, the majority of time will be spent on completing the physical deliverables. It should also be noted that since the project is still at a fairly early stage with respect to the technical aspects, the initial estimates of the tasks may not be entirely accurate and will be adjusted accordingly as time goes on to reflect a more accurate representation of the progress. Fallback plans have been established for each deliverable if problems in software development and debugging cause significant time delays. If necessary, major adjustments will be made to the proposals to ensure that the project is delivered to schedule.

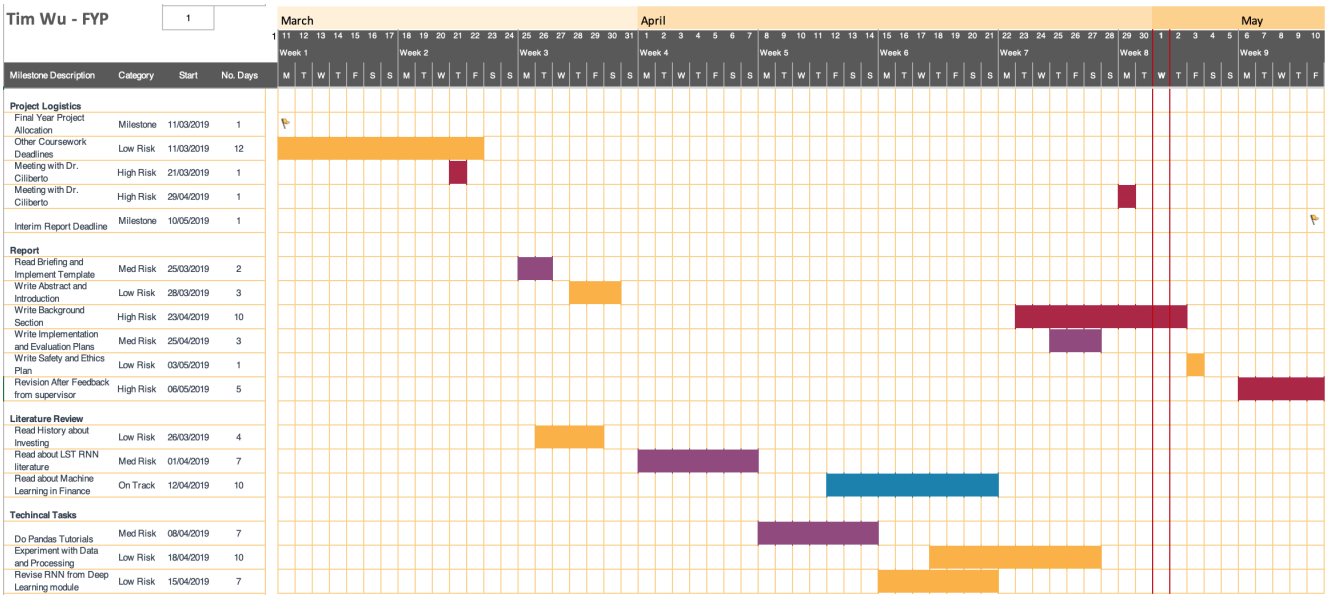

Figure 8. Gantt chart indicating completed tasks to interim report deadline.

3.1. Scheduled Tasks

1. Data Preprocessing: Using methods that have been researched and understood, the datasets supplied by Giano Capital will be cleaned and formatted in an appropriate fashion for use as the inputs of the LSTM model. This is a vital element of the project and it can be argued that the quality of model performance is critically dependent on this deliverable. However, it is important to observe the schedule as this is the initial deliverable and going overtime would put the other components in jeopardy. In a worst case scenario, priorities will be made to certain fundamentals that are theorised to be the most important and the others discarded to reduce workload.

2. Building and Testing Benchmark: The design for a suitable industry-standard benchmark is mentioned in section 4. Going forwards, the objective will be to build the tools necessary to compare the benchmark with the output data from the model. Similar to the fallback plan in Data Preprocessing, efforts will be solely concentrated on the key benchmark criteria (the most important being absolute returns) if there are heavy time constraints.

3. Implementing and Testing Model Architecture: Official documentation about the Keras API alongside example notebooks from the EE3-25 Deep Learning module will provide the basis for developing a well structured and coherent architecture. Initially, a simplistic baseline model will be constructed and tested in order to provide a fundamental level from which to develop a better and more sophisticated model. Parameters will be experimented with by applying similar logic to the research methods discussed in section 2.3. If time constraints become an issue, a simpler model will be developed and tested instead, with the baseline model being used in the absolute worst case scenario.

4. Integration Testing and Debugging: The prior three deliverables form components of the system and so will need to be tested when combined to see if the behaviour and results are as intended. This deliverable is flexible as testing can be performed to a wide degree of formality. A rigorous framework will require the construction of test datasets and their intended outputs, accounting for all edge cases that may occur. Consequently, significant time may need to be spend debugging and restructuring the codebase to account for these edge cases. The primary focus is for the development of a robust rather than an entirely correct system. The fallback plan is therefore to solely focus on accounting for errors that may occur in the Giano dataset as ultimately, this is what the model will use to perform analysis.

5. Hyperparameter Optimisation: To improve the performance of the model with respect to the benchmark, various modifications can be applied to areas of the algorithm such as the training method, network depth, learning rate optimisers etc. A significant proportion of time has been allocated to this since it involves the lengthy process of retesting and reevaluating the results and performance given every adjustment made. This final deliverable is arbitrarily long, given

the fact that analysis of components can be inexhaustibly extensive and so it will be flexible in the sense that as much experimentation will be performed as allowed for by time.

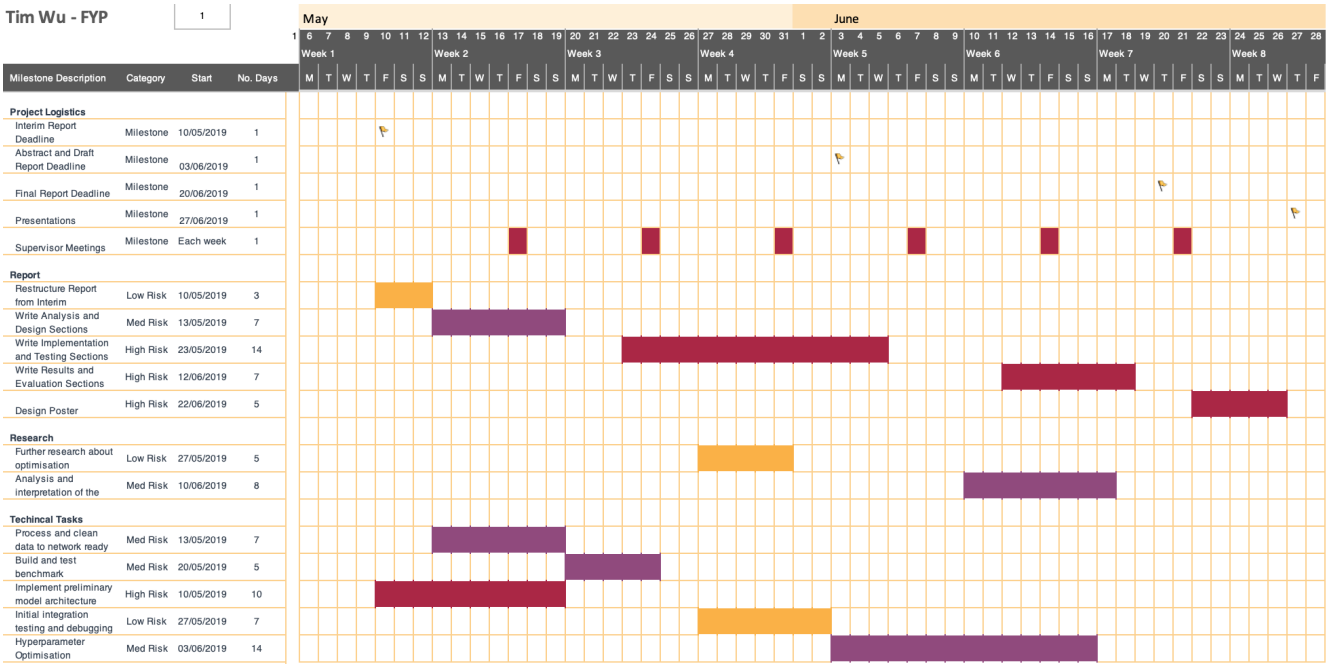

Figure 9. Gantt chart indicating future tasks after the interim report deadline.

Additional to the software deliverables, the required report writing, poster presentation and logistical meetings are also accounted for in the Gantt chart. Estimates for these tasks are fairly fixed since, unlike the software elements, it is unlikely that there will be significant, unforeseen problems and errors that will be time consuming.

4. Evaluation Plan

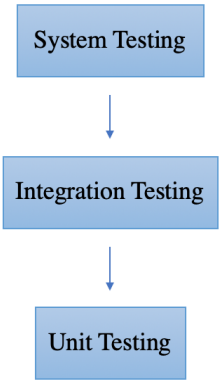

Figure 10. Levels of software testing from high (Top) to low (Bottom)

The procedure of evaluating the achievements of the project will vary according to the stage of the software development cycle. On the atomic level is the unit test, which will be used to validate that each of the separate modules and functions of the software are functioning as expected. An example[21] of how unit testing will be implemented is shown below. By using the assert statement in Python, it is possible to determine whether the function, $sum()$, is outputting the expected result of 6. An incorrect result will raise an $AssertionError$ and output the message $not6$.

```
def test_sum():
    assert sum([1, 2, 3]) == 6, "not6"
```

The next level up involves integration testing, which focuses on the correctness of functionality in combined components of the software. An example of this would be examining the outputs of a function which takes as an input a processed dataframe from another function. It is important to test the combined behaviour of these two components after individual unit testing to ensure that it is in line with expectations. Given the data orientated nature of the project, this will mainly be achieved through the development of small, well-defined test datasets to compare and contrast with the outputs of the components.

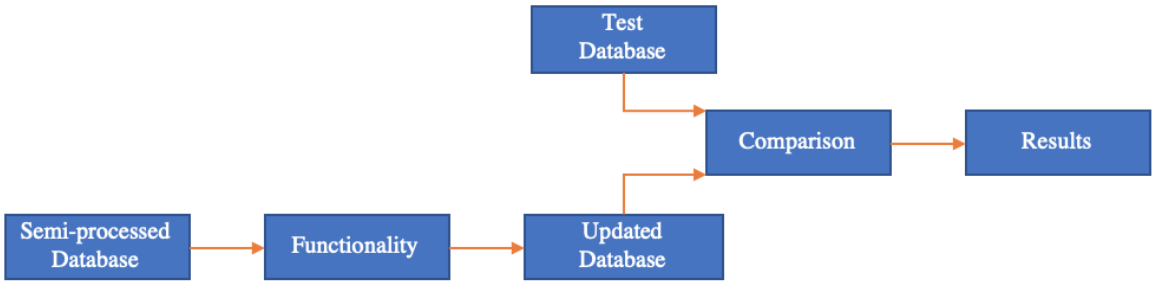

Figure 11. Flow diagram of how an integrated test may be executed.

System testing will involve evaluating the realised performance of the whole system. This will then allow for the model to be comparable to other methodologies developed in the field and industry. Our selected benchmark performance metric will be derived from the naive diversification choice heuristic. This means that for a given universe of N stocks, a portfolio will be constructed by allocating a $\frac{1}{N}$ weighting to each stock. The interest here is in the efficiency of such a portfolio compared to those which are optimised to favour certain stocks achieved by overweighting them such that their allocations are greater than $\frac{1}{N}$. DeMiguel et al. conducted investigations[5] regarding 14 mean-variance optimisation models and found that there was no statistically significant improved performance relative to $\frac{1}{N}$ diversification. As such, it is of interest for investment managers to construct mathematical models that can outperform the simplistic model often utilised by less sophisticated investors for its ease-of-use.

In order to evaluate this, the output values will be ordered and a theoretical portfolio constructed with respect to the percentage of the total sum of the outputs ($weighting(i) = \frac{output_i}{\sum_{i=1}^{N} output_i}$). It should be noted that the architecture of the network will be devised such that a larger output is better e.g. implies a higher return, lower volatility etc. Although this portfolio layout is by no means optimised for best performance, it succinctly describes and roughly equates to what stocks the model has selected to over and underweight. It is then possible to evaluate this portfolio with respect to three basic performance measure and compare and contrast to those of the naive diversification heuristic.

- Absolute Return: This is the most basic method of evaluation - observing how much return is generated by a portfolio over a given period of time. The simplicity of the measure allows it to be easily implemented and direct comparisons can be made between the model generated and benchmark figures. It is calculated by $100 \times \frac{V_{end} - V_{start}}{V_{start}}$ where V is the value of the portfolio and $start$ and end are the beginning and finish times of the investment. A higher absolute return indicates better performance.

- Sharpe Ratio: First introduced by William Sharpe[20] in 1966, this is a commonly used method of measuring risk adjusted return. Since a more risky investment should in theory offer a greater potential reward, a more insightful method of evaluation is judging whether the returns received are in line with the risk taken in a given portfolio. A simplified version will be used that does not account for the risk free rate to reduce complexity in calculations: $SimpleSharpeRatio = \frac{R_p}{\sigma_p}$ where R_p and σ_p are the return and its standard deviation of the portfolio respectively. A higher Sharpe Ratio indicates better performance.

- Portfolio Turnover: This measures how often the stocks within the portfolio have to be bought and sold to rebalance and maintain the model's strategy. Although it does not theoretically affect the performance, the transaction costs of

buying and selling in a real world scenario can significantly affect a portfolio's profitability. Therefore a lower turnover is preferred. It is calculated with the following equation: $turnover = \frac{min(s_b, s_s)}{NAV}$ where NAV is the Net Asset Value and s_b and s_s are the cost of stocks bought and sold respectively.

5. Ethical, Legal, and Safety Plan

This section details the issues relevant to the project and the best practices that will be taken. Principally, in order to conduct appropriate research, a common values and ethics code will be observed for the entire duration of the project. The following common areas were evaluated for this project:

- Carefulness: Mistakes in the research will be mitigated by frequently requesting feedback and reviews from the supervisor. Any mistakes identified and their subsequent results will be swiftly rectified as to maintain correctness within the research.
- Confidentiality: The privacy of the data that is provided by external parties will be upheld given the sensitivity of the information. This will be achieved by storing it in a private drive only accessible to the researcher and supervisor.
- Honesty: Results will be clearly stated and the methodology used to arrive at them clearly explained. This will improve the reproducibility of data, ensuring that nothing is fabricated or misconstrued.
- Human Subjects Protection: Any conflicts of interest between external parties and those at the College will be managed with clear communication so as to arrive at appropriate agreements.
- Integrity: All methods and subject matters that are sensitive will be screened for viability by the supervisor so that the public reputation of the College is not damaged.
- Objectivity: Biases in the results and research will be avoided by using a wide range of literature that offer differing opinions and perspectives.
- Respect for Intellectual Property: References will be used to acknowledge the work of others. Care will be taken to ask for permission when using unpublished results, methods or tools so as to avoid any chance of unintentional plagiarism.

Additionally, some legal issues that may arise were accounted for:

- Patent Infringement: Since there is a potential for the project to become commercialised, care must be taken to attempt to avoid infringing patents. If any issues arise, the supervisor will help advise the situation. A logbook will be kept to document the timeline of the project.
- Privacy: Given the sensitive nature of the data, consent will be requested from the provider before any formal publications are made.

Finally, below details a list of potential safety issues with plans of action to mitigate the potential of them occurring:

- Appliance Safety: It is not expected for portable appliances to be used as part of the project. However, if this were to occur, laboratory staff will be contacted for guidance and equipment to be PAT tested for safety.
- Biological Safety: No biologically hazardous materials are used as part of the project.
- Chemical Safety: No poisonous, irritant or allergenic materials are used as part of the project.
- Data Infrastructure Safety: The likelihood of viruses will be reduced through careful file management and software usage. Cloud storage will be used in place of USB flash drives to prevent the opportunity of physical tampering. Preventing excess use of computation resources is achieved externally by the Google Colaboratory environment, which offers up to 12 hours of continuous access and 13GB of usage per period.

- Electrical Safety: Care will be taken when plugging in desktops or laptop chargers whilst working on the project. Precautions will be made so as not to tamper with the wiring of desktops in the laboratory and to ensure that the casing of personal laptop chargers are well insulated and not damaged.
- Fire Safety: Although there are few sources from within the project itself that could cause fire, care will be taken to observe evacuation routes in shared environments so as to prevent any harm or injuries.
- Physical Safety: Due to the software centric nature of the project, there are no interactions with large or fast moving physical objects that could potentially cause harm.
- Study Participant Safety: Tests and surveys will not be used as part of this project.

6. Bibliography

References

[1] *Jupyter Notebook Documentation*. Project Jupyter, 2018.

[2] H. Ahmadi. *Testability of the Arbitrage Pricing Theory by Neural Network*. Proceedings of the International Conference on Neural Networks, 1990.

[3] J. Alberg and Z. C. Lipton. *Improving Factor-Based Quantitative Investing by Forecasting Company Fundamentals*. arXiv, 2018.

[4] C. Bergmeir, R. J. Hyndman, and B. Koo. *A Note on the Validity of Cross-Validation for Evaluating Autoregressive Time Series Prediction*. Elsevier, 2017.

[5] V. DeMiguel, L. Garlappi, and R. Uppal. *Optimal Versus Naive Diversification: How Inefficient is the 1/N Portfolio Strategy?* Oxford University Press, 2007.

[6] C. S. Eun, W. Huang, and S. Lai. *International Diversification with Large- and Small-Cap Stocks*. Journal of Financial and Quantitative Analysis, 2008.

[7] E. F. Fama and K. R. French. *Common risk factors in the returns on stocks and bonds*. Journal of Financial Economics, 1993.

[8] B. Graham and D. Dodd. *Security Analysis*. Whittlesey House, 1934.

[9] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra. *DRAW: A Recurrent Neural Network For Image Generation*. arXiv, 2015.

[10] B. M. Henrique, V. A. Sobreiro, and H. Kimura. *Literature review: Machine learning techniques applied to financial market prediction*. Expert Systems With Applications, 2019.

[11] S. Hockreiter and J. Schmidhuber. *Long Short Term Memory*. MIT Press, 1997.

[12] K. jae Kim. *Financial time series forecasting using support vector machines*. Neurocomputing, 2003.

[13] T. Merz and P. Janus. *Low-Volatility Investing: Empirical evidence of the defensive properties of low volatility enhanced portfolios*. UBS, 2016.

[14] C. Olah. *Understanding LSTM Networks*. colah.github.io., 2015.

[15] J. Racine. *Consistent cross-validatory model-selection for dependent data: hv-block cross-validation*. Journal of Econometrics, 2000.

[16] G. Robertson. *Machine Learning in Investment Management*. Man AHL, 2018.

[17] S. A. Ross. *The Arbitrage Theory of Capital Asset Pricing*. Journal of Economic Theory, 1976.

[18] G. Rouwenhorst and W. N. Goetzmann. *The Origins of Value: The Financial Innovations that Created Modern Capital Markets*. Oxford University Press, 2005.

[19] D. E. Rumelhart, P. Smolensky, J. L. McClelland, and G. E. Hinton. *Schemata and Sequential Thought Processes in PDP Models*. Morgan Kaufmann Publishers, 1986.

[20] W. F. Sharpe. *Mutual Fund Performance*. Journal of Business, 1966.

[21] A. Shaw. Getting started with testing in python. https://realpython.com/python-testing/.

[22] J. L. Treynor. *Market Value, Time, and Risk*. 1961.