Enhanced Equity Ratings: A Deep Dive into LSTM and Random Forest Classifications

Emanuel Cristian Ciudin
University College London (UCL)

Abstract

Equity ratings, which provide Buy/Hold/Sell classifications, are computed based on subjective valuation methods which promote bias and inconsistent metrics across financial institutions. In this paper we propose a standardised way of labelling and computing equity ratings for mid-frequency investment decisions via the use of an LSTM algorithm. This algorithm aims to be a foundational step towards a software used to support traditional finance professionals in portfolio selection. By utilising both stock and commodity data, we seek to explore the potential in augmenting current research in the field. The LSTM algorithm is benchmarked against an ensemble of Random Forests and reveals superior F1 score and other key metrics. This confirms the efficacy of the LSTM model in providing an objective and consistent approach to generating equity ratings.

1 Introduction

Finance has been an ever-changing field since its inception thousands of years ago, with the rise of the first advanced civilisations. From the barter systems and primitive concepts of futures and options contracts in Ancient Mesopotamia (Code of Hammurabi [16]) to the Amsterdam Stock Exchange creation and depiction by Joseph de la Vega in his book *Confusion of Confusions* [23], the evolution of the financial markets reflects the growing complexities of economic activities and human interactions.

In the past two centuries, financial theory has undergone significant development and structuring, splitting into two distinct, but intercorrelated areas: traditional finance and quantitative finance. Quantitative finance, which employs mathematical models to analyse the financial markets, has seen a substantial surge in the past 3 decades, fueled by the added computational power which led to the advent of algorithmic trading. Machine learning (ML) and deep learning (DL) methods have also been used extensively in recent years, showing promising results in mid-frequency and long-frequency investment horizons [20, 6]. Currently, fundamental finance concepts and quantitative finance models are both employed when taking investment decisions by large buy-side and sell-side institutions, yet rarely are these methodologies integrated and used in conjunction.

This paper aims to offer a quantitative perspective of a traditional finance concept, equity ratings, by generating a more robust allocation of Buy/Hold/Sell classifications via a DL algorithm. Sell-side equity research reports (buy-side reports are confidential), generally employ a valuation model, such as the Discounted Cash Flow (DCF) and Earnings Multiplier, together with a market analysis to produce a rating for the security, considering its price progression in the next 12 months [8]. Although valuable in making investment decisions, sell-side reports have shown to be biased. Data indicates significant cultural bias in analysts: they are more inclined to offer a positive stock recommendation if the company is headquartered in their home country [17]. Optimistic bias has also shown to be present, albeit less in large investment banks [4]. Additionally, the different scales of defining buy, hold and sell criteria promotes rating inconsistency across institutions. In addressing the aforementioned issues related to equity research, this study explores a data driven approach for a standardised short-term rating generation mechanism which aims to ensure consistent and objective stock ratings.

1.1 Literature Review

ML methods have become increasingly popular in quantitative finance, with the number of published papers per year increasing threefold in the last decade only [22]. Being widely used in signal generation, risk management and

portfolio allocation, ML has not only augmented but also refined the precision of traditional financial and statistical methodologies. For example, DL algorithms, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have shown as high as a 22.6% increase in accuracy when forecasting next-day time series data, compared to the Seasonal Autoregressive Integrated Moving Average with Exogenous Factors (SARIMAX) benchmark [14]. Another famous class of RNN's, Long-Short Term Memory (LSTM) Networks, was applied on a financial time series prediction problem and reduced the Root Mean Squared Error (RMSE) by approximately 85% when compared to the established ARIMA model [21]. Recent years have revealed extensive publishing of time series classification tasks addressed by the use of AI applications. In terms of DL architectures, algorithms that are particularly designed to harness the temporal aspects of time series, such as RNNs, have shown a higher performance in cryptocurrency price trend classification when compared to Gradient Boosting Decision Trees (GBDT), a powerful ML classification algorithm [13]. Random Forest (RF), as part of the established classical machine learning algorithms, has also proven to be a powerful benchmark, outclassing KNN (K-Nearest Neighbour), Naive Bayes and singular decision trees in classification of intraday S&P 500 returns [3].

Moving further to establish a clear setting of our issue at hand, predicting whether a security should be bought, sold or held in the near future can be expressed as a classification problem with 3 classes, based on a multivariate time series input. This is essentially a Markov Chain in which the current state (time series of historical asset price and other trade characteristics) encapsulates all relevant information to make a decision. Each state would represent a distinct market condition, and the objective is to infer the transition probability to subsequent states and therefore compute a predicted class label. A similar scenario has been tackled before by Borovkova et al, which used an ensemble LSTM model for intraday market classification of stocks on "buy" and "sell" classes, proving increased efficiency compared to regularised logistic regression classifiers [2]. The algorithm was trained on 1-year of trading data organised in 5-minute interval samples for 20 large cap US stocks. The multivariate time series input considered features such as closing price and volume as well as computed ones like Sharpe Ratio, Bollinger Bands and volatility of returns. 12 LSTM networks were selected to form the ensemble approach, and the resulting probability of the output to belong to the buy class was taken as average of the singular outputs. The model was evaluated via the Area Under the Receiving Operating Characteristic (AUROC) to the best accuracy of 0.56. This results provide evidence of predictability for intraday patterns, but leaves room for further enhancement of the algorithm, both in terms of architecture and feature generation. Another 2014 paper by Rechenthin, which proposed an ensemble of thousand of ML and DL models, capable of storing past performances and adapting to current market conditions, achieved a general ROC of 0.58 [18]. This is remarkable considering the singular models are classical classification algorithms, such as Support Vector Machines and Decision Trees, and the use of only stock-related features.

1.2 Objectives and Research Scope

In this study we endeavour to take the aforementioned research further, investigating the existence of medium frequency classification patterns and extending the feature horizon to incorporate data from other financial instruments beyond the equities' realm. Therefore, the primary objective of this paper is to build a classification algorithm which would assess whether a specific stock should be bought, held or sold, considering the upcoming 30-day period. To achieve this, we build an LSTM network capable of processing a multivariate time series input, composed of both stock data and external data, such as commodities data. Additionally, we compare the performance of the LSTM model against an ensemble of RF classifiers, each applied to the individual univariate time series data. The output of the RF algorithms are combined to form a Logistic Regression input which aims to maximise the individual models. As this is a multiclass task, the results will be compared via one-to-many AUROC considering the Ensemble Random Forrest as a benchmark result. The insights of this report would prove valuable in closing the gap between traditional finance and quantitative models, as it looks to provide a standardised and unbiased approach to generating short-term equity ratings. By leveraging both ML and DL methodologies and incorporating external data, this paper seeks to optimise industry results and create a benchmark model for aiding in reliable, objective, investment decisions.

The rest of the report is organised as follows. In the Methodology section, we first review the mathematical principles underlying the LSTM network and the Random Forest null model, alongside their application in this context and the deployed architectures. Then we discuss the data collection, detailing the exploratory analysis and preprocessing steps which were undertaken. The Results section follows to compare the 2 approaches in both qualitative and quantitative fashions, also presenting the validation scheme employed. Finally, the Discussion section aims to synthesise the findings and contemplate on the limitations, validity and potential implications of this study, while also acknowledging the potential for future work in the field.

2 Methodology

2.1 RNN Background

We will briefly detail how an RNN functions and what added complexity is added to its structure to arrive at the LSTM network. RNNs are a family of artificial neural networks which incorporate recurrent feedback connections. This is particularly useful when the input is sequential as the network is able to consider multiple input values at once in order to make a prediction [19]. In the vanilla RNN, the architecture is fully-connected and contains the input layer, hidden layers and an output layer:

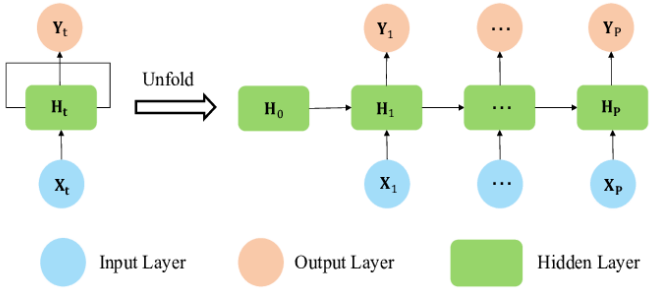

Figure 1: Unfolded and Folded view of the Vanilla RNN [9]

In Figure 1, we can see the way the hidden states are dynamically changed as all the input is fed sequentially to the network. Let us provide a more formal definition of the dynamic hidden states:

$$h_t = f_W(h_{t-1}, x_t) \tag{1}$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \tag{2}$$

$$y_t = W_{hy}h_t \tag{3}$$

In equations (1),(2) and (3), we have the hidden state at time t, h_t, which is computed by the non-linear activation function ($f_W = \tanh$ in this case). Consequently, h_t is the output of the previous state h_{t-1} and the input x_t with the respective weight matrices, W_{hh} and W_{xh}. Multiple layer RNNs are similarly designed, but this time h_t becomes h_t^l, as each state is computed sequentially considering the outputs from the previous layer and from the previous time step of the current layer:

$$h_t = \tanh(W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}) \tag{4}$$

The limitations of this network design arise when considering the backpropagation algorithm. In backpropagation, we reuse the same weight matrix at each time step, multiplying by its transpose once for every cell that we pass. Therefore, when we try to compute the gradient for the first hidden state h_0, we are forced to backpropagate through every cell, effectively multiplying by the weight matrix W_{hh}^T a large number of times. If the weight matrix includes numbers larger than 1 this is called the exploding gradient problem, where gradient values can rapidly increase to excessively large magnitudes, leading to numerical instability. The exploding gradient problem can be solved by weight clipping in practice, where gradients are scaled down if they exceed a certain threshold. For weights smaller than 1, this becomes the vanishing gradient problem, where gradients shrink exponentially and become unable to reach a local minimum.

2.2 LSTM Network

The Long Short Term Memory Network was invented by Hochreiter and Schmidhuber in 1997, as a solution to the vanishing gradient problem discussed above [7].

Let us define the forget gate (f), the input gate (i), the candidate gate (g) and the output gate (o) which are pictured below, in Figure 2:

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \tag{5}$$

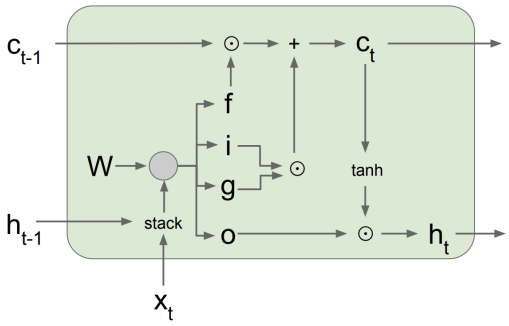

Figure 2: LSTM cell [5]

Now we can define the cell state c_t and the hidden state h_t in terms of the above as:

$$c_t = f \odot c_{t-1} + i \odot g \tag{6}$$

$$h_t = o \odot \tanh c_t \tag{7}$$

Where $\odot$ denotes the element-wise operation and σ the sigmoid function. The use of sigmoid functions, compared to the RNN's $\tanh$, provides for positive fractions (as the codomain of the function is [0,1]) of the input, forget and output gates. The candidate gate (g), as per the name suggests, transforms the previous hidden state value, h_{t-1} and the input, x_t, to form the basis of the new state. The input gate weighs the values provided by the candidate gate to determine the new information which should be added to form the new state c_t. The forget gate quantifies how much information should be forgotten from the previous cell state c_{t-1}. The output gate has a similar usage to the input gate, as it aims to filter the current calculated value for the new cell state, c_t, to arrive at a final value for the new hidden state. Now looking at equations (6) this becomes evident, as c_t is formed from the addition of the past state and the new information. In equation (7), the new hidden state is the element-wise product of the output gate and c_t, showcasing how the LSTM modulates the cell state through a non-linear transformation.

2.2.1 LSTM Architecture

As expressed in the literature, the stacked LSTM (multiple LSTM layers) architecture provides a simple approach to extracting higher level features and achieving higher performance [10] [24].

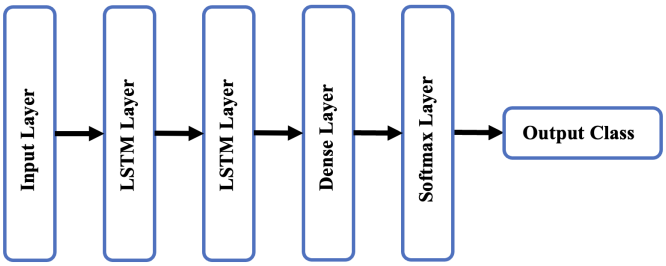

Figure 3: LSTM Architecture of the proposed algorithm [5]

Our proposed architecture presented in Figure 3 takes advantage of this. The input layer initialises the network with the input shape, which is detailed in the Data section. The 2 stacked LSTM layers extract the temporal features at a hierarchical level of abstraction. Their output is then passed to a Dense layer with fully-connected neurons that maps the learned features to an output vector formed by the number of classes. As there are 3 classes in this instance, Buy/Hold/Sell, the output will be a 3-element vector. The Softmax function, present in the Cross-Entropy loss, transforms the elements into class probabilities. The final output will be selected as the highest predicted probability in the output vector and therefore belong to the respective class.

2.3 Random Forest

The null model is formed from a RF ensemble. RF in itself is an ensemble model, which unites Decision Tree algorithms to minimise the variance of its singular units and, therefore, increase the generalisation of the model.

Decision Trees are used for both regression and classification tasks in a similar fashion and provide the foundation of modelling non-linear relationships. In our case, a classification model is needed to predict the buy/hold/sell classes and we will briefly detail the inner workings of a decision tree classifier.

As with all ML algorithms, Decision Trees aim to minimise the error function. In this case the algorithm used to minimise it is called Recursive Binary Splitting. Recursive Binary Splitting searches across the input space, J, and splits the data at the point which provides the highest reduction in error into separate regions. The most common ways to assess the quality of the split used in Decision Trees classifiers are the Gini Index and the cross-entropy function [1]:

$$G = \sum_{i=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{8}$$

$$D = -\sum_{i=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk} \tag{9}$$

Consequently, the regions $R_i \in J$ are selected to minimise node impurity or entropy. The output of a Decision Tree is the same as the majority of the training outcomes in that region. In the end, the output of the RF algorithm is the majority of the votes of the individual trees.

Our algorithm employs 7 RFs (all using the Gini Index as the splitting criterion), one for each individual feature. This was needed as the RF algorithm requires a feature vector and cannot comprehend multiple time series. The 7 feature time series are discussed below in the Data Section. The outputs of the 7 RFs are concatenated to form a $1X7$ vector. This vector is then fed to a Logistic Regression model to produce the final output, therefore combining all the 7 features.

2.4 Data

The data was extracted using the Bloomberg Terminal for 10 stocks from the Technology Sector, all listed on NASDAQ, besides Visa which is listed on NYSE, to ensure consistent behaviour. The tickers of the 10 stocks are: AAPL, META, TSLA, AMZN, NVDA, GOOGL, ADBE, V, NFLX, MSFT. Data for all 10 stocks are used in training, validation and testing to promote generalisation on the tech sector for universal investment decisions. Alongside these 10 stocks, 2 commodities were selected to allow for more external information as discussed in the Objectives and Research Scope section. The commodities are gold and copper, with the considered assets being their continuous futures contract. Copper and gold were considered as they are strongly tied to the tech industry in terms of hardware manufacturing, with many of the above companies (Apple, Microsoft, Nvidia etc.) relying heavily on them for production. The aim is to classify stock data into buy/hold/sell, consequently, the commodities will only be used as features. All the data was downloaded from $1st$ of January 2016 to $21st$ of February 2024.

2.4.1 Data Exploration

The Closing Price and the Volume of each stock and commodity were downloaded for the period mentioned above. The data was cleaned to exclude NaN values and make sure all the dates match in all instruments. We are left with 2047 data points for every stock and commodity.

The log returns of the closing prices are also calculated and added, as they are stationary and reduce the fluctuations found in the closing prices. The Simple Moving Averages of 30 days of the prices are also computed for noise reduction and a smoother view of the trend. All 10 stock prices, volumes and the 2 commodities are tested for stationarity by employing the Augmented Dickey Fuller (ADF) test. The ADF test considers that the variable is an autoregressive process of order p ($AR(p)$) and tests for the presence of a unit root which would infer non-stationarity. It is found that all prices, for both commodities and stocks, are non-stationary, while the volumes present a stationary behaviour. For reference, the closing prices and the aggregated log returns of the 10 stocks are plotted in Figure 4.

It can already be observed from the closing prices, that most stocks are highly correlated, at different magnitudes of course. Each input will contain a single stock price to avoid such correlations between and therefore reduce multicollinearity. Now, considering we are looking to run the algorithm which will output an investment decision (Buy/Hold/Sell) for a 30-day horizon, we would like to compute the returns of the current price, p_t compared to an average future price p_{t+30}. More about this will be explained in *Experiment Setup*. Let us also show the aggregated (all 10 stocks included) future returns distribution in Figure 5.

From Figure 5 we can see that the 30-day average returns are positively skewed with a slightly negative mean. The mean was calculated at -0.02 with a standard deviation of 0.13. Consequently, there are many small negative

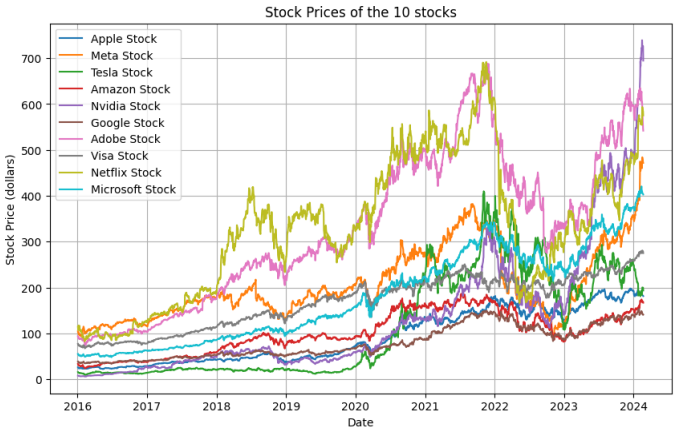

Figure 4: Stock Prices of the 10 stocks

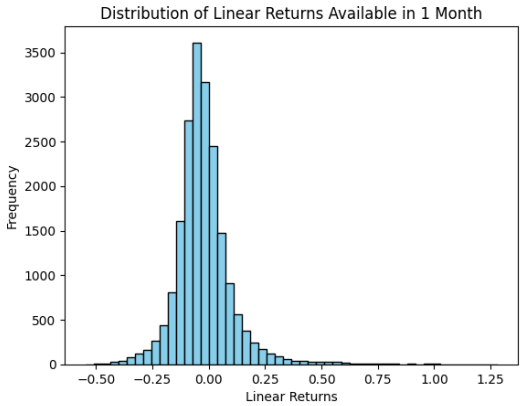

Figure 5: Aggregated Linear Future Returns Distribution

returns, but we can see in both Figure 4 and Figure 5, that the right tail of the distribution is fat, with quite a few high returns.

2.4.2 Experiment Setup

Taking into account the points discussed above, non-stationary data was excluded according to the ADF test. It is standard to perform analysis on stationary series to ensure consistency between training and testing datasets. Therefore, the first difference or log returns were considered to arrive at the following shape of the input:

$$X_{ti} = \begin{pmatrix} rlog_{t-29,i} & V_{t-29,i} & SMAD_{t-29,i} & Crlog_{t-29} & CV_{t-29} & Grlog_{t-29} & GV_{t-29} \\ rlog_{t-28,i} & V_{t-28,i} & SMAD_{t-28,i} & Crlog_{t-28} & CV_{t-28} & Grlog_{t-28} & GV_{t-28} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ rlog_{t,i} & V_{t,i} & SMAD_{t,i} & Crlog_{t-28,i} & CV_t & Grlog_t & GV_t \end{pmatrix} \tag{10}$$

Matrix X_{ti} represents the input at time t and stock i. The 7 columns of X_{ti} are explained below, in order:

$$\begin{cases} rlog_{t,i} = \text{The log return of stock } i \text{ at time } t. \\ V_{t,i} = \text{The daily trading volume of stock } i \text{ at time } t. \\ SMAD_{t,i} = \text{The difference of the 30-day SMA of stock } i \text{ at time } t. \\ Crlog_t = \text{The log return of copper futures at time } t. \\ CV_t = \text{The daily trading volume of copper futures at time } t. \\ Grlog_t = \text{The log return of gold futures at time } t. \\ GV_t = \text{The daily trading volume of gold futures at time } t. \end{cases} \tag{11}$$

As mentioned before the copper and gold data are shared between different stocks as they are separate features used to add information. At every time step, 10 samples would be present, all with different Y values, depending on the performance of the respective stock in the subsequent 30 days period. As we are interested to assess future 30-day label prediction, it is fair to consider the same period in the past, prompting every input to be of shape $(30, 7)$. To make sure no colinearity exists we have computed the correlation matrix of the 7 columns in which no extreme values are found, enabling us to continue with this setup:

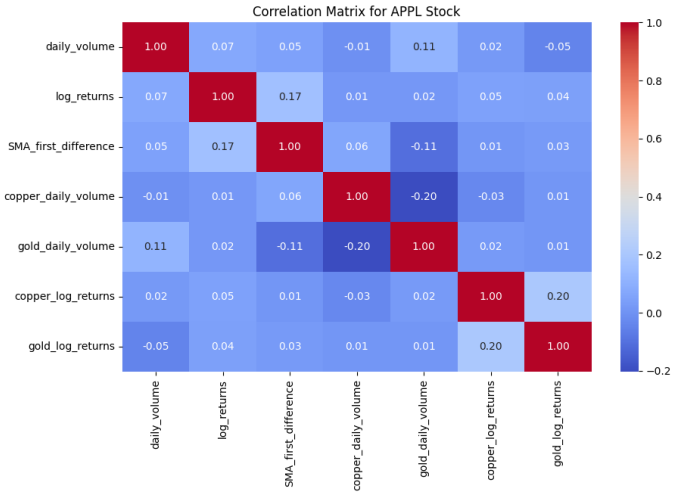

Figure 6: Correlation Matrix for Apple Stock samples

2.4.3 Labelling

We have followed the standard labelling convention and associated the following with the 3 classes:

Class Description	*Label*
Buy	2
Hold	1
Sell	0

Figure 7: Class Labels

Let us now define the criteria used for the Buy/Hold/Sell classes. As we are targeting an investment decision on a 30-day horizon it is sensible to calculate a return with the closing prices at the time t and time $t + 30$. This might prove to be a problem as the market is highly volatile, registering many small losses as was pointed out in Figure 5. Consequently, instead of comparing the closing price at time t to the one at time $t + 30$, we compare it with an average closing price around day 30:

$$R_t = \frac{\bar{P}_{t+26:t+35} - P_t}{P_t} \tag{12}$$

$$\textit{Where: } \bar{P}_{t+26:t+35} = \frac{1}{10} \sum_{i=0}^{9} P_{t+26+i} \tag{13}$$

This encourages the network to search for a rising/declining trend in the stock, which aligns with the trend-following investment strategies. To provide a standardised way of defining the thresholds for the labels that promotes flexibility to the investment decision, we have opted for the scheme in Equation (14).

Let y_{sample} be defined as follows, where R_t is defined in Equation (12), R_f is the *risk-free rate*, and R_{min} is the *minimum buy return*:

$$y_{\text{sample}} = \begin{cases} 2 & \text{if } R_t > R_{\text{min}} \quad \text{(Buy)} \\ 1 & \text{if } -R_f < R_t \leq R_{\text{min}} \quad \text{(Hold)} \\ 0 & \text{if } R_t \leq -R_f \quad \text{(Sell)} \end{cases} \tag{14}$$

To enhance adaptability, the minimum return should be calculated by considering the required Sharpe Ratio of the strategy. In our case we have opted for a Sharpe Ratio of 0.5 as this is a monthly "strategy". The risk-free rate was approximated during the period to be roughly 2%.

$$R_{min} = R_f + SharpeRatio \cdot \sigma \approx 0.02 + 0.5 \cdot 0.125 = 0.0825 \tag{15}$$

Considering the high volatility and negative mean recorded in Figure 5, the class distribution shows a majority Sell class and a minority Buy class:

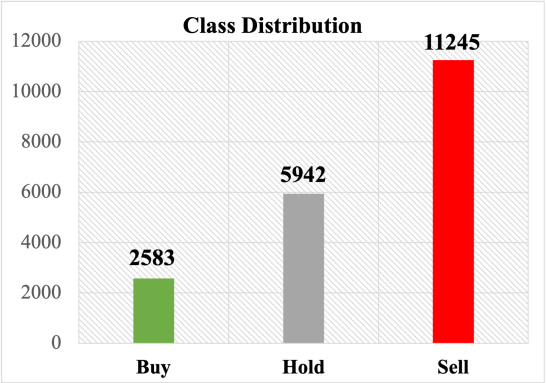

Figure 8: Class Imbalance

2.5 Scaling and Handling Class Imbalance

Class Imbalance is a frequent classification problem discussed by many academics [12]. To mitigate this in our dataset, we have opted for Random Oversampling of the Buy and Hold Classes. As we are dealing with time series samples showing different patterns across time, Synthetic Minority Oversampling Technique (SMOTE), was deemed inappropriate. That is because SMOTE uses the KNN algorithm to interpolate new synthetic instances between existing similar samples, and this may not capture the full complexity of the markets. As we have a multivariate time series, variables such as the log returns and volume are interrelated and the synthetic samples will not capture the full dependency which is non-linear in nature. Random Undersampling of the minority class was also not employed, as the number of samples is small and it promotes loss of information.

Scaling is crucial to the adaptability and interpretation of the neural network. In this context it was done separately on each of the 7 time series present in Equation (10), using the MinMax function:

$$X_{\text{std}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \tag{16}$$

$$X_{\text{scaled}} = X_{\text{std}} \cdot (X_{\text{max}} - X_{\text{min}}) + X_{\text{min}} \tag{17}$$

Minmax doesn't reduce the effect of outliers, although it normalises the data within the $(0, 1)$ range. As all the included time series are stationary, it is appropriate to calculate a standard deviation of the listed values. When splitting data in training and testing a random stratified split was utilised to ensure the same distribution in both datasets and therefore avoid potential scaling issues. As we have employed, K-Fold Cross Validation, a scaler is first fitted to the training data (without scaling it) to be used for the testing dataset. The training data is then scaled when the folds are created in the validation loop. This process ensures no data leakage.

2.6 K-Fold Cross Validation and Hyperparameter Optimisation

The advantage of using k-fold cross validation over a single validation dataset stems from the generalisation in hyperparameter tuning. As the LSTM algorithm is continuously trained on k-1 folds and validated on the last fold, this fights against overfitting and makes better use of the data. As mentioned before, the random oversampling was done inside the folds to ensure minimal data leakage.

In terms of hyperparameters, the number of epochs, learning rate, batch size and hidden size of the LSTM layers were optimised through the exhaustive Grid Search method. Finally, the selected parameters were: *learning rate* $= 0.001$, *hidden size* $= 50$, *batch size* $= 32$ and *epochs* $= 30$.

2.7 Optimiser and Loss Function

The loss function utilised in the LSTM implementation is the Cross-Entropy Loss. This is particularly useful and a standard for multiclass classification because it measures the distance between the predicted probability distribution and the actual distribution of the labels (it is one of the two terms of the Kullback-Leibler Divergence). The Cross-Entropy Loss function in PyTorch is the equivalent of using a Log Loss after a Softmax function is applied. The Softmax function transforms the raw outputs of the network (a 1x3 vector in this case, one value per class) to a vector of probabilities. Consequently, the Softmax function for a 1x3 vector is defined on $\sigma : \mathbb{R}^3 \to (0,1)^3$:

$$\sigma(\mathbf{z})_i = \frac{e^{(z)_i}}{\sum_{j=1}^{3} e^{(z)_i}} \tag{18}$$

The cross-entropy loss is the log-likelihood of the conditional probability:

$$Loss = \sum_{j=1}^{3} y_i * log(\hat{y_i}) \cdot \tag{19}$$

Where $y_i = 1$ if the index i is the true label and $y_i = 0$ otherwise. $\hat{y_i} = \sigma(\mathbf{z})_i$ is the predicted probability from the Softmax function.

The optimiser employed is the Adam optimiser. Much research has been done in the past decade regarding novel optimisation algorithms, with Adam being introduced by Kingma and Ba in 2014 [11]. Adam, or adaptive moment estimation, combines AdaGrad and RMSprop into a single algorithm. Adam considers the exponential moving averages of the first moment (mean) and second moment (variance) of the gradient. Consequently, Adam has a dynamic learning rate α which is updated in terms of the last computed exponential average parameters β_1 and β_2. This minimises the risk of overshooting, while ensuring efficiency and stability.

2.8 Performance Metrics

Besides the accuracy, which may be biased due to class imbalances, an accurate estimation of a classification algorithm's performance is the Area Under the Receiver Operating Characteristic (AUROC). The ROC, reports the True Positive Rate (TPR) and False Positive Rate (FPR) of the algorithm at different thresholds:

$$\begin{cases} \text{TPR} = \dfrac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \\ \text{FPR} = \dfrac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}} \end{cases} \tag{20}$$

Therefore, the AUROC measures the area under the FPR and TPR. AUROC is a valuable tool in binary classification, so considering we have a multiclass problem at hand, we will utlise this method in one-to-many class comparisons. The F1 score is a similar metric used to understand real results without the bias of class imbalance. This will also be applied in one-to-many label comparisons:

$$\begin{cases} \text{Precision} = \dfrac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \\ \text{Recall (TPR, Sensitivity)} = \dfrac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \\ \text{F1 Score} = 2 \cdot \dfrac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \end{cases} \tag{21}$$

3 Results

The AUROC was computed and plotted below in the one-to-many case. Therefore each class is considered the positive class, while the other 2 form the zero class to compute the specific label ROC.

The confusion matrix for the 2 algorithms is also presented in Figure 11. Finally, the overall statistics for the 2 algorithms are attached in the table in Figure 12.

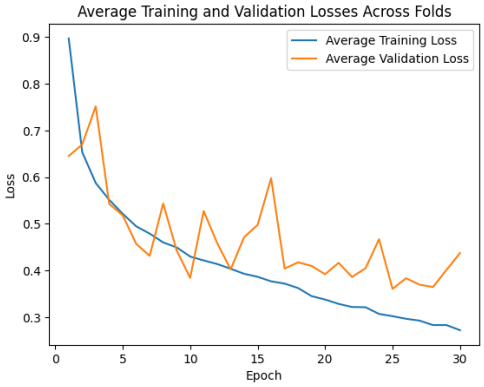

Figure 9: Training and Validation Loss across epochs

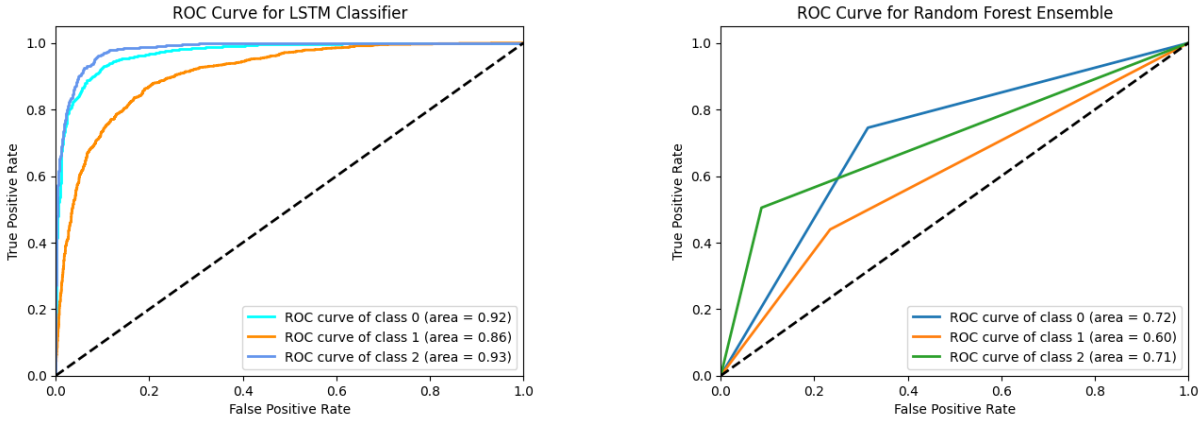

Figure 10: ROC one-to-many format for LSTM (left) and Random Forest Ensemble (right)

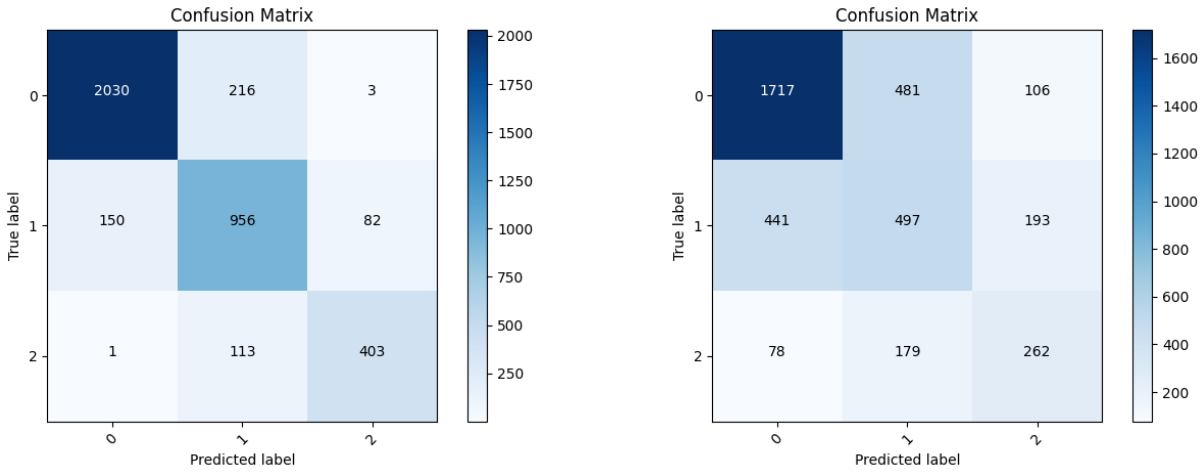

Figure 11: Confusion Matrix for LSTM (left) and Random Forest Ensemble (right)

	Random Forest Classifier	***LSTM Model***
Accuracy (Testing)	0.63	0.85
AUROC Class 0	0.92	0.72
AUROC Class 1	0.86	0.60
AUROC Class 2	0.93	0.71
Precision Class 0	0.77	0.93
Precision Class 1	0.43	0.74
Precision Class 2	0.47	0.83
Recall Class 0	0.75	0.90
Recall Class 1	0.44	0.80
Recall Class 2	0.50	0.78
F1 Score Class 0	0.76	0.92
F1 Score Class 1	0.43	0.77
F1 Score Class 2	0.49	0.80

Figure 12: One-to-Many Evaluation Matrix

4 Discussion

Plotted in Figure 9, we have the training and validation loss values computed by the Cross-Entropy Loss function which was discussed in chapter 2.7. As the LSTM was built for k-fold cross validation with 3 folds, the plotted values are the average for the 3 folds. All the evaluation metrics (besides the confusion matrix) are computed in the one-to-many case to ensure an accurate view of the Buy/Hold/Sell classification. We can see from Figures 11 and 12 that all the results clearly favour the LSTM algorithm. The Confusion Matrices show that, although the RF performed reasonably well in classifying sells (label 0) and buys (label 2), it couldn't really decide on the hold class. It should be noted that the RF ensemble's performance matches that of the highest performing singular RFs (which used Simple Moving Average stock data). Consequently, the ensemble model didn't extract any additional insights from the other, lower performing, datasets in classification. The hold class seems to be learned poorly by the RF algorithm, as evidenced by its lower precision, recall, and F1 score compared to the other two classes. Class 1 also posed some challenges for the LSTM algorithm, with a lower one-to-many ROC and F1 Score registered. The main advantage of the LSTM is that it easily differentiates between the buy and sell labels, with only 3 samples classified as a buy, when the true label was a sell, and a single sample classified as a sell, when the label was a buy. This greatly enhances the algorithm's potential to be used in aiding investment decisions. While the high AUROC values of the LSTM model theoretically indicate exceptional performance, interpreting multiclass ROCs is a complex task and the results should be taken with a grain of salt. A much fairer measure is the F1 score included in Figure 12, where we can see that the results underscore LSTM's superiority in classifying all three labels. Again we can see that Class 1 exhibits the poorest prediction, but its performance is close to that of the Buy class. One direct solution to this misclassification of the Hold class is switching from 3 to 5 classes, adding strong buy/strong sell classes. The investment professional would then only use strong buy and strong sell classifications to be more confident in their decisions. A slight predisposition to predict class 0, when the true label is 1, is also present in the LSTM results, indicating that the oversampling function employed, SMOTE, may have not fixed the bias completely. Again, it is worth mentioning that the evaluation metrics are also highly dependant on the way the data is labeled. As the label structure we provided in Chapter 2.4.3 essentially predicts a trend in the next 30 days, it might be opportune to explore tighter future averages to "localise" the future price prediction.

4.1 Future Work

Considering the LSTM architecture employed, we've achieved substantial improvement in terms of accuracy, AUROC and F1 Score. There are a number of methods that can be implemented to optimise these results further and advance the algorithm to a suitable level which can be utilised in the investment industry. One of them could be the integration of alternative data sources. Media and news articles have already proven useful in financial forecasting, with sentiment analysis being one of the most researched topics in the field in recent years [15]. Providing features extracted from financial news might bolster the algorithm's performance. Data augmentation should also be sought after in other asset classes in a similar fashion to how we extracted gold and copper futures contracts for use in our feature vector. For this to work efficiently, a combinatorial search algorithm could be developed that would search through the feature space (many listed financial instruments from different asset classes) and provide the optimal subset. Finally, other architectures also suited for time series analysis could improve the current results. For example, adding an attention layer which elevates LSTM's capabilities of processing the input sequence or directly utilising a transformer for its self-attention mechanism, could capture higher level patterns of the time series.

4.1.1 Conclusion

In this research, we've effectively demonstrated that medium frequency patterns in stock equity data can be used to offer a different approach to computing equity ratings. Based on the high-frequency research done by Borovkova et al., we have expanded the scope to also include data from the commodities sector, exceeding the original study's results [2]. In order to achieve this, we have built an LSTM model capable of processing a multivariate time series input for a multiclass classification, and an ML ensemble benchmark, formed from 7 Random Forest classifiers. Oversampling techniques and cross-validation-imposed hyperparameter optimisation were also used to enhance the capabilities of both networks. Our results, validated by the use of robust evaluation metrics such as AUROC and the F1 Score, prove that such a model has the potential to be integrated in an industry setting.

References

[1] James Gareth et al. *An introduction to statistical learning: With applications in R.* Springer, 2013.

[2] Tsiamas I Borovkova S. "An ensemble of LSTM neural networks for high-frequency stock market classification". In: *Journal of Forecasting* 38 (2019), pp. 600–619. DOI: 10.1002/for.2585.

[3] Pasi Luukka Christoph Lohrmann. "Classification of intraday SP500 returns with a Random Forest". In: *International Journal of Forecasting* (2019), pp. 390–407. DOI: 10.1016/j.ijforecast.2018.08.004.

[4] Paola De Vincentiis. "Accuracy and Bias of Equity Analysts in an Environment Characterized by Higher Disclosure: Empirical Evidence from the Italian Market". In: (Feb. 2010).

[5] Danfei Xu Fei-Fei Li Ranjay Krishna. "Recurrent Neural Networks". In: (2021).

[6] Jan Grudniewicz and Robert Ślepaczuk. "Application of machine learning in algorithmic investment strategies on global stock markets". In: *Research in International Business and Finance* (2023). DOI: 10.1016/j.ribaf.2023.102052.

[7] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[8] Shahed Imam, Jacky Chan, and Syed Zulfiqar Ali Shah. "Equity valuation models and target price accuracy in Europe: Evidence from equity reports". In: *International Review of Financial Analysis* (2013). DOI: 10.1016/j.irfa.2013.02.008.

[9] Ye Jiexia et al. "How to Build a Graph-Based Deep Learning Architecture in Traffic Domain: A Survey". In: (May 2020).

[10] Bo Peng Jin Wang and Xuejie Zhang. "Using a stacked residual LSTM model for sentiment intensity prediction". In: *Neurocomputing* 322 (2018). DOI: 10.1016/j.neucom.2018.09.049.

[11] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: (2017). DOI: 10.48550/arXiv.1412.6980. arXiv: 1412.6980 [cs.LG].

[12] B Krawczyk. "Learning from imbalanced data: open challenges and future directions". In: *Progress in Artificial Intelligence* 5 (2016). DOI: 10.1007/s13748-016-0094-0.

[13] et al. Kwon Do-Hyung. "Time Series Classification of Cryptocurrency Price TrendBased on a Recurrent LSTM Neural Network". In: *Journal of Information Processing Systems* 15.3 (June 2019), pp. 694–706. DOI: 10.3745/JIPS.03.0120.

[14] Manisa Pipattanasomporn Mengmeng Cai and Saifur Rahman. "Day-ahead building-level load forecasts using deep learning vs. traditional time-series techniques". In: *Applied Energy* (2019), pp. 1078–1088. DOI: 10.1016/j.apenergy.2018.12.042.

[15] Kostadin Mishev et al. "Evaluation of Sentiment Analysis in Finance: From Lexicons to Transformers". In: *IEEE Access* 8 (2020), pp. 131662–131682. DOI: 10.1109/ACCESS.2020.3009626.

[16] Christian Pauletto and Steve Kummer. "The History of Derivatives: A Few Milestones". In: (May 2012). DOI: 10.13140/RG.2.2.13901.15844.

[17] Vesa Pursiainen. "Cultural Biases in Equity Analysis". In: *The Journal of Finance* (2022). DOI: 10.1111/jofi.13095.

[18] Michael D. Rechenthin. "Machine-learning classification techniques for the analysis and prediction of high-frequency stock direction". In: *ProQuest Dissertations and Theses* (2014), p. 291.

[19] Hojjat Salehinejad et al. "Recent Advances in Recurrent Neural Networks". In: (2018). DOI: 10.48550/arXiv.1801.01078.

[20] Abhijit Sharang and Chetan Rao. "Using machine learning for medium frequency derivative portfolio trading". In: (2015). DOI: 10.48550/arXiv.1512.06228. arXiv: 1512.06228 [q-fin.TR].

[21] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. "A Comparison of ARIMA and LSTM in Forecasting Time Series". In: (2018), pp. 1394–1401. DOI: 10.1109/ICMLA.2018.00227.

[22] Warin T. and Stojkov A. "Machine Learning in Finance: A Metadata-Based Systematic Review of the Literature". In: *Journal of Risk and Financial Management* (2021). DOI: 10.3390/jrfm14070302.

[23] Jose de la Vega. *Confusion de Confusiones*. Martino Publishing, 1688. ISBN: 1614274517.

[24] Changhua Hu Yong Yu Xiaosheng Si and Jianxun Zhang. "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures". In: *Neural Computing and Applications* 31 (2019). DOI: 10.1162/neco_a_01199.