A practical implementation of:

(Re-)Imag(in)ing Price Trends*

Assignment 4

Aidan van Niekerk

CID: 02363561

Preliminary Note

It is advised to first read the README file in the accompanying GitHub repository. In the interest of time, some of the required code can run while the reader reads the report.

**A publication by Jiang, Kelly and Xiu (2020).*

Introduction

Many industries have benefited from the advent of deep learning as a tool for image processing and analysis. Investment management is, however, not necessarily the first industry that comes to mind when one explores applications of image processing. Traditionally the applications of machine learning in investment management deal with regular tabular data. There is one piece of investment management that has always relied on a strong visual aspect: technical analysis. Simply put, technical analysis is a method used by traders and investors in which stock price charts are studied for recurring patterns to forecast future stock returns. Technical analysis is not short of controversy owing to its contradiction of fundamental financial hypotheses (such as the random walk hypothesis and efficient market hypothesis) and the mixed literature examining its profitability. As Lo, Mamaysky and Wang (2000) puts it:

"Nevertheless, technical analysis has survived through the years, perhaps because its visual mode of analysis is more conducive to human cognition, and because pattern recognition is one of the few repetitive activities for which computers do not have an absolute advantage (yet)."

Figure 1: An example of a technical analysis pattern indicating a "buy" signal on www.questrade.com/learning/investment-concepts/data-analysis/technical-analysis

To this end Jiang, Kelly and Xiu (2020) present an interesting piece of research, "(Re-)Imag(in)ing Price Trends" whereby they investigate the ability of convolutional neural networks to learn patterns in present in US stock price charts to predict future returns. CNNs were trained to predict the probability that a stock price will go up or down in a 5, 20, 60-day period. In other words, one uses a CNN to perform its own technical analysis on US stocks in lieu of a human trader. This assignment is an implementation of their model on the South African stock market to assess how well the methods outline by Jiang, Kelly and Xiu generalize to smaller stock markets.

Data Selection

The primary South African stock market index is the Top 40 Index which comprises the 40 biggest listed companies on the Johanessburg Stock Exchange (JSE). The data period over which the analysis is conducted is split into two periods, (i) **training/validation** and (ii) **testing**. The training/validation period runs from 2000/01/01 to 2014/12/31. The testing period runs from 2015/01/01 to 2023/11/30. The "**training/validation**" period refers to the data sample used to train the CNN; the "**testing**" period refers to the period whereby the model buys stocks based on its predictions. This already marks a departure from (Re-)Imag(in)ing Price Trends which used a dataset of thousands of US stocks, a training/validation period from 1993-2000, and a testing period from 2000 to 2020.

Of course, technical analysis using CNNs necessitates the need for visual charts in the form of images. Given that this is a module on unstructured data, it is apt that this analysis creates all its own images from scratch! Jiang, Kelly and Xiu (2020) present a method to create price charts using only stock price data (which is freely available). The relevant data for a stock price chart is typically the *open, high, low,* and *close* prices for several days – in addition to the daily volume traded (see Output 1). This data is typically visualised using a OHLC (open, high, low close) chart – see figure 2. The left arm of a bar represents the day's opening price, the right arm represents the day's closing price, the top of the bar represents the day's highest price, and the bottom of the bar represents the day's lowest price. The volume bars are presented at the bottom of the chart.

Date	Open	Close	High	Low	Volume	Ticker
2000-01-04	13800.0	13600.0	13960.0	13600.0	300476	NED
2000-01-05	13500.0	13120.0	13500.0	13000.0	270635	NED
2000-01-06	13400.0	12780.0	13400.0	12780.0	110879	NED
2000-01-07	12800.0	12800.0	13800.0	12800.0	260708	NED
2000-01-10	13500.0	14600.0	14900.0	13500.0	419092	NED

Output 1: Price Data for Nedbank (NED)

Figure 2: OHLC chart of Nedbank from finance.yahoo.com

Stock price charts also regularly include indicators such as a moving average. Part I of the Jupyter notebook downloads the prices from Yahoo Finance. Part II of the notebook creates daily OHLC charts (without volume) each spanning a 5-day and 20-day period (hence two sets of charts). Further, more daily OHLC charts with a 5-day and 20-day moving average (respectively are created). Hence, the four sets of OHLC charts created are:

Table 1: Characteristics of created image sets

Period	Moving average	Volume	Height (Pixels)	Filename
5 days	-	-	32	Data/Images/Top 40/i5.csv
20 days	-	-	64	Data/Images/Top 40/i20.csv
5 days	5 days	Yes	32	Data/Images/Top 40/i5_vma.csv
20 days	20 days	Yes	64	Data/Images/Top 40/i20_vma.csv

The charts are constructed so that each daily bar is coloured white and 3 pixels wide – the background of the chart is black. The left pixel denotes an opening price, the middle pixels denote the high-low line, the right pixel denotes a days closing price. Hence each chart is 3n pixels wide (whereby n is the number of days the chart spans). The prices for the stock period are arranged so that the highest price in the period reaches the top pixel of the chart and the lowest price reaches the bottom pixel. In the case of charts with volumes, the volume bars occupy the bottom 20% of the image and a single row of blank pixels separate the OHLC bars from the volume bars. Moving averages are overlayed on the OHLC bars (where appropriate) and are drawn using Bresenham's line algorithm.

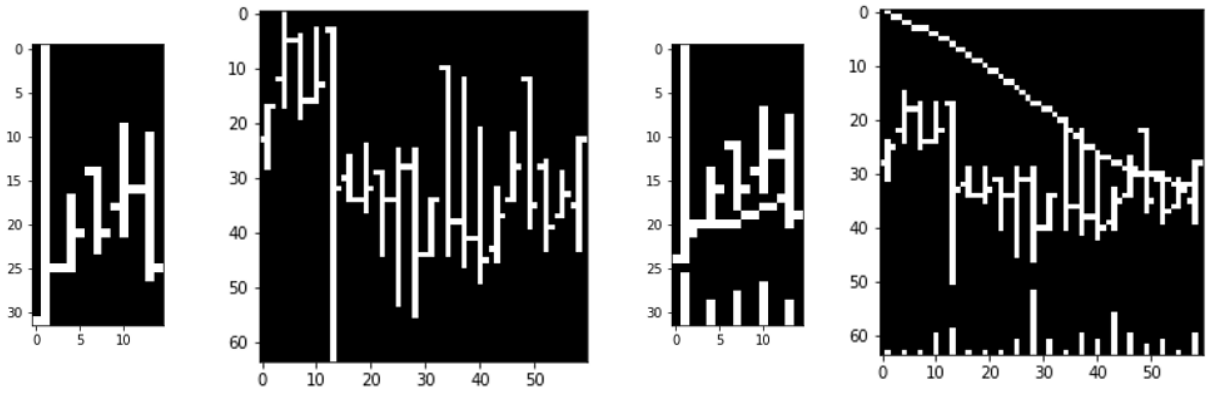

Figure 3: Examples of created OHLC charts (as ordered in Table 1)

The saved files each include the variables: start date of the chart ('start_date') , end date of the chart ('end_date'), chart image data in vector form ('img_data'), return over the next 5 days ('Fwd_Ret_5'), returns over the next 20 days ('Fwd_Ret_20'), stock ticker ('ticker'). A casual note is that in this context a "day" refers to a trading day (not a calendar day). The data are now in a convenient form for later analysis.

	start_date	end_date	img_data	Fwd_Ret_5	Fwd_Ret_20	ticker
0	2016-02-12	2016-03-10	[0, 255, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...	-0.006258	0.021582	ANH

Output 2: Example of the dataset contents.

Problem Statement

Now that we are armed with a comprehensive chart image library, we have the data we need to perform technical analysis. In theory, a technical trader could look at the charts (such as in figure 3) and hypothesise whether to make buy or sell decisions for their investment portfolio. However, instead we seek to train a machine learning model to learn patterns in the image dataset and predict up or down directions in stock price (i.e. perform a machine learning version of technical analysis) and build an investment portfolio based upon its predictions.

Specifically, we seek to develop a convolutional neural network to predict the direction (up or down) of the future stock price accurately and reliably. The desired output of the model should be a probability whereby 100% denotes a predicted upward direction and 0% predicted downward direction. Furthermore, we aim to use this probability score to extract the signal from the very noisy financial data. The intent is to leverage the model predictions to not only create a profitable investment portfolio, but to create an investment portfolio that beats a naïve benchmark. Comparing an investment portfolio to a predefined benchmark is a hallmark of performance comparisons in the investment space and is usually used to determine whether an investment manager has added value to a portfolio. The basic metrics to compare investments are:

- **Annualised return**: The average return an investment generates per year. A higher return is desired by an investor.
- **Annualised standard deviation (risk)**: The standard deviation per year of the investment. A lower risk number is more beneficial.
- **Sharpe ratio**: The "risk-return" ratio of an investment. Calculated by dividing annualized return by standard deviation. The higher this number the more investors are compensated per unit of risk they take on.

Jiang, Kelly and Xiu (2020) suggested a series of convolutional neural networks to process the code-generated charts and make predictions. Their analysis was based on 5, 20, and 60 day data. This analysis produces both 5 and 20 day data, but in the sake of processing times only uses 5 day data in the training of CNNs. The details of the method are described in the next section. The authors already proved that their methodology (based on thousands of US stocks) has merit by producing profitable investment portfolios based on CNN predictions with higher returns, lower risk, and higher Sharpe ratios compared to benchmarks. As mentioned, this analysis uses a much smaller dataset for training, which leads to the natural research questions:

"Given a much more restricted South African dataset, can the CNN-based method suggested by Jiang, Kelly and Xiu (2020) accurately predict stock price direction over 5 days? Can we use these predictions to create a portfolio that beats a benchmark on the basis of return, risk, and Sharpe ratio?"

Jiang, Kelly and Xiu (2020) use 5-, 20-, and 60-day charts to predict 5-, 20-, and 60-day returns. Specifically, table 2 highlights the decile performance of a CNN that predicts 5-day return using 5-day charts. The "decile Q" refers to in which decile a stock probability is ranked, with a stock in decile 10 having the highest probability of going up. The performance of each decile is calculated as the average return of the stocks in that quantile over the subsequent 5 days. That is, it is the realised return that an investor achieves over the subsequent 5 days by owning the stocks of decile Q in equal proportion. For a historical simulation, it is assumed that the investor repeats this process for every 5-day period in their test sample (2000-2020). If the CNN could accurately predict stock price direction, one would expect the high decile (10) to achieve the highest return (and Sharpe) and the low decile (1) to achieve the lowest return (and Sharpe). Table 2 illustrates that it is indeed the case – providing strong evidence that CNNs can identify patterns in stock prices and make accurate predictions.

Table 2: Summary of results from (Re-)Imag(in)ing Price Trends showing profitable performance for 5-day portfolios

Decile	Average 5-Day Return (%)	Annualised Sharpe Ratio
Low	-0.28	-1.92
2	-0.04	-0.27
3	0.03	0.15
4	0.08	0.41
5	0.09	0.48
6	0.14	0.70
7	0.17	0.84
8	0.22	1.06
9	0.30	1.48
High	0.54	2.89

Description and Justification of Methods and Analysis

The general methodology described below conforms to the headings in the notebook. Part I deals with data collection; Part II describes image creation; Part III details model creation and training; and Part IV analyses investment performance.

Part I: Data Collection

All stocks in the JSE Top 40 contain a unique code (ticker). A csv file ('Data/stock_tickers_40.csv') containing the tickers for the Top 40 constituents is loaded. Open, high, low, close and volume data for all stocks for the period 2000/01/01 - 2023/11/30 are fetched from Yahoo Finance (using the 'yfinance' library). The resulting data is stored in a dictionary 'stock_data'. Each entry is of the form shown in ouput 1 above.

Part II: Image Creation

Images were created for the 5-day and 20-day charts, although only 5-day charts were used in the analysis. The image creation is described in the Data Selection section above, but for clarity the 5-day process is outlined:

1. A stock's pricing data is subset to a 5-day period.
2. A blank black image of height 32 and width 15 pixels is created, i.e. a 32x15 matrix with 0 entries.
3. A OHLC bar is drawn in white (a value of 255) with each day's price data. Each bar is 3 pixels wide: the left single pixel is the open arm; the right single pixel is the closing arm; and the vertical bar is the high-low line. The process is repeated for all 5 days.
4. The prices are scaled so that the highest price and lowest price over the 5 days touch the top and bottom pixel rows of the image, respectively.
5. In the case of volume bars and moving averages:
    a. The picture is adjusted so that volume bars span the bottom 20% of the image with a blank pixel row separating it from the OHLC bars.
    b. The 5-day moving average is plotted over the existing OHLC bars.
6. The process is repeated per ticker for every rolling 5-day period. Hence assuming 250 trading days per year and 23 years of data and 40 stocks, one can expect roughly 23*40*250 = 230 000 images per set. The details of set characteristics are in table 1.

To complement the images, a dataset is created with start and end date are provided in addition to a subsequent 5-day and 20-day return and a stock ticker label. Output 2 provides an example of an entry in such a dataset. The process is lengthy and runs for approximately 30 minutes.

Part III: Model Creation and Training

Training Data

The dataset used to train and validate the model runs for the period 2000/01/01 – 2014/12/31. The data created in Part II is read into the notebook arranged into a convenient form. Data is rudimentarily cleaned: entries with 5-day returns more than 100% or less than -90% are removed. Stocks with a positive 5-day return are labeled as 1, whereas negative returns are labeled as 0.

The training/validation set is randomly split into a 70%/30% chunks. The random split allows for a more equal class label distribution in the training set (to prevent the model from training on data that is exclusive to one period's economic conditions). The loss on the validation set is used to determine at which epoch to stop training (details below).

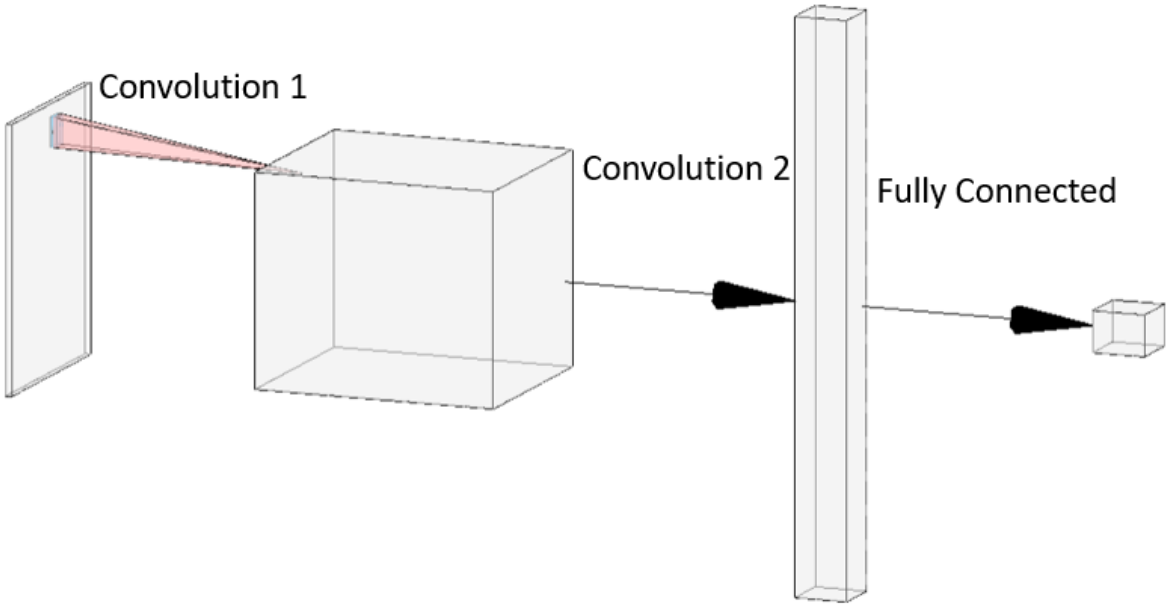

Figure 4: A representation of the model architecture

Model Architecture

To predict future 5-day returns using 5-day charts, Jiang, Kelly and Xiu (2020) propose a CNN with an input layer, 2 convolutional layers, and 1 fully connected layer. A 5x3 convolution is applied in both convolutional layers. The size specification allows the model convolution to consider 1 entire OHLC bar at a time (since it is 3 pixels wide). The model is specified in the notebook under the "model specification" heading and designated as class 'CNNi5'.

- **Input layer:** 32x15 image
- **Convolution 1:** 5x3 convolution; LReLU activation, 2x1 max pooling, 2x1 padding, stride 1, 64 batch normalization, output 64x16x15
- **Convolution 2:** 5x3 convolution; LReLU activation, 2x1 max pooling, 2x1 padding, stride 1, 128 batch normalization, output 128x8x15
- **Fully Connected Layer:** 50% drop out, Softmax activation function, output size 2 - designating probability of down (0) and up (1), respectively.
- Xavier initialiser is applied to each layer; initial biases are set to 0.01, learning rate is set to 0.00001, Adam algorithm optimiser is used, binary cross-entropy as loss function, batches are sized as 128.
- The model stops training when the loss for the validation set does not improve for 2 consecutive epochs (or runs for a maximum of 20 epochs).

```
i5CNN(
  (layer1): Sequential(
    (0): Conv2d(1, 64, kernel_size=(5, 3), stride=(1, 1), padding=(2, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=(2, 1), stride=(2, 1), padding=0, dilation=1, ceil_mode=False)
  )
  (layer2): Sequential(
    (0): Conv2d(64, 128, kernel_size=(5, 3), stride=(1, 1), padding=(2, 1))
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=(2, 1), stride=(2, 1), padding=0, dilation=1, ceil_mode=False)
  )
  (DropOut): Dropout(p=0.5, inplace=False)
  (FC): Linear(in_features=15360, out_features=2, bias=True)
  (Softmax): Softmax(dim=1)
)
```

Output 3: CNN implementation using PyTorch

CNN implementation was done using PyTorch and training/validation splitting was done using sklearn.

Part IV: Prediction and Investment Performance

Once the model is trained, it can be used to make predictions on stock price movement and build investment portfolios. Each week, the model will assign a probability to each stock. A higher probability corresponds to a higher likelihood of a good investment performance (in theory). It is assumed that the investor starts with an investment of R100 (100 South African Rand). The investor will buy/sell stocks weekly to realise a profit or loss. The investor splits their capital equally among all stocks selected, i.e. if the investor buys n stocks then 1/n% of their capital will be allocated to each stock. To create performance comparisons with the model's suggested stocks and a benchmark, we create 3 different portfolios:

- **Portfolio 1 ("Rich Portfolio")**
  In this portfolio the investor buys the top 8 of the stocks recommended by the model at each 5 day interval.
- **Portfolio 2 ("Poor Portfolio")**
  In this portfolio the investor buys the bottom 8 of the stocks recommended by the model at each 5 day interval.
- **Portfolio 3 ("Benchmark Portfolio")**
  The investor ignores the model and naively allocates their capital to all 40 stocks equally at each 5 day interval. This is in essence a random control.

To compare the value of the investment insight of the 3 models, the investment strategies outlined above are calculated for the period 2015-2023. This allows for performance comparison in terms of annualised return, annualised risk, Sharpe ratio, and total capital growth. **It is important to note that if the model is good at predicting which stock prices will go up, then portfolio 1 will outperform 2 and 3. If the model is good at predicting which stock prices will go down, then portfolio 2 should perform worse than 1 and 3.**

Interpretation and Reflection on Output

Some of the code used in implementation was adapted from assignment 1 (van Niekerk 2023).

Model Training

In implementation, the CNN trained for a total of 9 epochs achieving a training and validation loss of 0.6595 and 0.6927 (see # Output 4 in notebook). I was unable to ensure exact consistency in training and validation loss, but the model used to evaluate investment performance in Part IV is saved as "Models/save/i5r5_original.pth". All implementation was done using a laptop and the entire training time ran for approximately 15 minutes.

Model Accuracy on Validation Set

We assess the model's accuracy on the validation set for interest. Owing to the noise in financial markets and the small stock dataset, it is unlikely that the model will achieve high accuracy. Since the investment portfolio will only own 8 stocks at a time, overall accuracy is not of primary importance. Instead, we are interested in whether the model can select 8 stocks better than a naïve benchmark portfolio. Nonetheless, the model achieves accuracy slightly better than chance.

Table 3: Performance Metrics of CNN

Accuracy	53%
Precision	54%
Recall	62%
F1-Score	58%

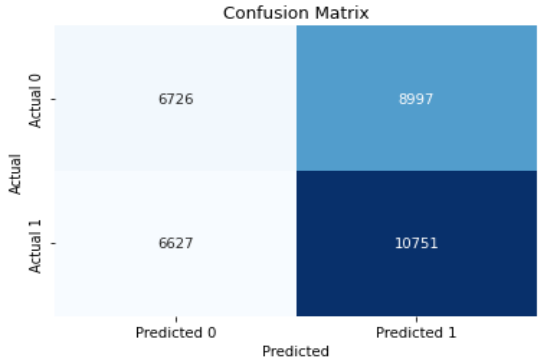

Figure 5: Confusion Matrix of classifier

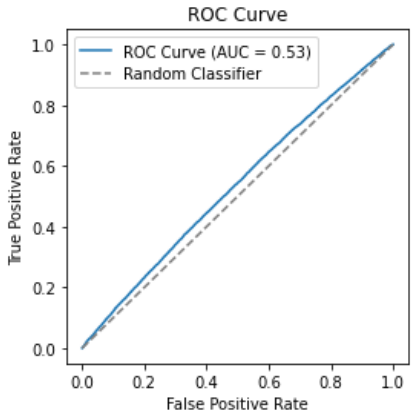

Figure 6: ROC curve of classifier

Overall model accuracy is slightly better than chance (not surprising given the complexity of financial data). The ROC curve does suggest that the model works as intended. Interestingly, the precision and recall metrics greater than 50% indicates that the model may very well be better at chance at distinguishing which stock prices will rise. The model appears less adept at predicting which stocks prices will go down. It remains to test the model's predictions on the historical sample from 2015-2023.

Historical Portfolio Performance

Portfolios 1 ("Rich"), 2 ("Poor"), and 3 ("Benchmark") were constructed according to the descriptions above. Table 4 details the performance characteristics of each portfolio over the sample 2015-2023. The Rich portfolio exhibits much better performance in terms of return and Sharpe ratio – it has almost double the average annual return of the benchmark (22.79% vs 11.76%). Its Sharpe ratio (measure of risk-reward) is also substantially higher than benchmark (0.92 vs 0.59). Its risk is slightly higher, but it is not unexpected as the Rich portfolio holds only 8 stocks at a time compared to the benchmark's 40 stocks (hence it is not as diversified and theoretically more volatile). **The outperformance of the rich portfolio compared to the benchmark suggests that the model is very adept at selecting stocks which have a higher probability of increasing in value.**

Table 4: Portfolio Performance 2015-2023

	Portfolio 1 (Rich)	Portfolio 2 (Poor)	Portfolio 3 (Benchmark)
Annual Return	22.79%	-12.63%	11.76%
Annual Risk	24.66%	23.96%	20.04%
Sharpe Ratio	0.92	-0.53	0.59
Growth of R100	R457.48	R23.92	R220.83

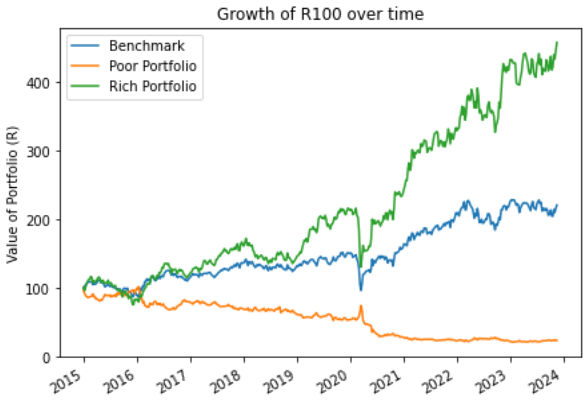

Figure 7: Cumulative Growth of Portfolios

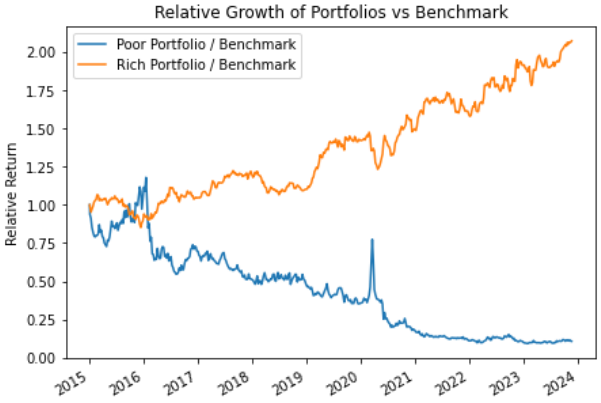

Figure 8: Relative Growth of Portfolios

The Poor portfolio performs particularly poorly, indicating that the model is able to identify stocks that have a high likelihood of depreciating in value. The cumulative graph (figure 7) shows the extent by which the Rich portfolio outperforms and the Poor portfolio underperforms. In Rand terms, the Rich portfolio accumulated double the value of the benchmark over the same period (R457 vs R220).

The relative performance graph (figure 8) tracks the performance of the Rich and Poor portfolio over time relative to the benchmark portfolio. A value greater than 1 indicates the portfolio performs better than the benchmark. It is encouraging to note that the Rich portfolio consistently slopes in an upward direction while the Poor portfolio consistently slopes downwards. This indicates that the model can consistently identify "buy" and "sell" signals from the market. It is worth noting that the model was only trained once and was not updated during the testing period.

The profitable performance of the Rich portfolio and the poor performance of the Poor portfolio allows us to state – in terms of the above research questions – that:

There is evidence to suggest that the methodology outlined by Jiang, Kelly and Xiu (2020) generalise well to the South African market. The CNN can – within an acceptable degree of accuracy – predict stock price movement. With these predictions, it is possible to create an investment portfolio that outperforms a naïve benchmark.

Conclusion

Jiang, Kelly and Xiu (2020) presented a unique approach to investment management and stock selection. Their results clearly provide evidence that their methodology is profitable. The analysis above suggests that their method generalises well to the South African market. There is great practical application and novelty in investment management in using machine learning technology for stock selection. The results above – despite a limited dataset – suggest that even relatively simple CNNs are able to extract complex signals in noisy financial data.

References

Jiang, Jingwen and Kelly, Bryan T. and Xiu, Dacheng, (Re-)Imag(in)ing Price Trends (December 1, 2020). Chicago Booth Research Paper No. 21-01, Available at SSRN: https://ssrn.com/abstract=3756587 or http://dx.doi.org/10.2139/ssrn.3756587

Lo, A.W., Mamaysky, H. and Wang, J. (2000). Foundations of Technical Analysis: Computational Algorithms, Statistical Inference, and Empirical Implementation. *The Journal of Finance*, [online] 55(4), pp.1705–1765. doi:https://doi.org/10.1111/0022-1082.00265.

van Niekerk, A. (2023). 'Assignment 1', Unstructured Data Analysis, MLDS, Imperial College London. Unpublished assignment.