Assignment 2 - Group: 131

Tutors: Canh Dinh, Yixuam Zhang, Thomas Selvaraj

Group members: Hari Nath Bingi (490619580), Samuel Curtis (430505410), Dijun Ng (460425843)

1.0 Abstract

This paper outlines a comparative analysis of machine learning and deep learning models with the overall objective of predicting next day close prices for a subset of New York Stock Exchange (NYSE) stocks. We investigated the predictive ability of linear regression, extreme gradient boosting (XGBoost), and a long short-term memory neural network (LSTM) model. The objective of our research was to determine a model that generalises across the entire NYSE as determined by the sectors to which each stock belongs. Through the processes of feature engineering and hyperparameter optimisation we found that the LSTM model exhibited the strongest performance with an average mean squared error (MSE) of 0.692 over a one year testing period in 2016.

2.0 Introduction

Advancements in the fields of machine and deep learning have had significant implications across a wide variety of sectors. The uptake of these methods to enhance predictive analytics in financial markets has been aggressive as competing firms seek to outperform one another. In this paper our group seeks to create a predictive algorithm for forecasting next-day market close prices of 11 major stocks listed across the New York Stock Exchange (NYSE) each spanning a different market sector. We have tasked this regression to three algorithms that vary in their approach; linear regression, extreme gradient boosting (XGBoost), and a long short-term memory neural network (LSTM). Each algorithm is trained with the objective function of minimising mean squared error (MSE) of daily log-returns and evaluated by MSE of next day close prices as will be discussed in Section 4 (methodology).

The dataset we are working with contains two files with data collected from Yahoo! Finance, NASDAQ Financial-100, and EDGAR SEC databases. The first being NYSE trading data regarding all S&P 500 stocks at a daily granularity spanning from 2010-2016. More specifically, this includes the daily open, high, low, close prices as well as total traded volumes. In this work we have used the adjusted prices data rather than the raw prices data. Adjusted prices simply means that the price throughout time has been standardised in instances where companies have undergone a stock split (increased the total number of issued shares) or issued dividends.[1] The Standard & Poor's 500 (S&P 500) is an index of roughly the largest 500 companies listed on the NYSE as determined by market capitalisation. The second file contains company fundamental data at an annual level (sometimes bi-annual for a small subset). This data

contains information such as company earnings (before and after tax), total accounts payable, value of fixed assets, amongst many other factors and can be used to determine the general 'health' of a company's financial status. Because our group has elected to forecast prices at a daily granularity we have elected to mostly discard company financial data due to the fact that it is reported annually except for where it is used for further feature engineering as will be discussed in the methodology component of this paper.

In order to be able to truly test the ability of our models we decided to evaluate our models on stocks across the 11 different sectors within the S&P 500, these being consumer discretionary, consumer staples, energy, financials, health care, industrials, information technology, materials, real estate, telecommunications services, and utilities. We determined the market capitalisation of each company (*Eq 2.1*) and sought to forecast the largest company per sector. By doing so we remove biases that may exist between highly correlated stocks within the same sector. For example, if our group sought to only forecast Google and Apple stock prices, a model that predicts only that the price will go up the next day is likely to achieve good results due to the strength of the technology industry in those years. Furthermore, Google and Apple share prices are very highly correlated and we cannot infer that the model is a good predictive model by simply evaluating these two stocks. By introducing numerous sectors we mostly remove this correlation and can more accurately determine if our model is able to generalise over different prevailing market conditions.

$$market\ capitalisation = total\ shares \times share\ price$$

Eq 2.1 *Market Capitalisation*

This paper will be outlined below this point as follows. Section 3 will provide a brief summary of related works as found in academic literature providing us with some established benchmarks for comparison. Section 4 will outline the methodology focussing on data cleaning and preprocessing, feature engineering, and outlining the implementation of each algorithm. Section 5 will demonstrate the experiments we have undertaken in hyperparameter optimisation and a discussion of results. Section 6 will conclude the paper with our overall thoughts regarding the performance of each of the models and ideas for further work.

3.0 Related Work

3.1 Linear Regression

Siew and Nordin[2] compared various regression techniques to predict stock price and concluded that linear regression did the best producing a small root MSE of 18.59 in comparison with Additive Regression, Regression by Discretization, Sequential Minimal Optimization (SMO) Regression. This research is quite old and there have been quite some advancements in the field of machine learning after 2012, as part of our work we are experimenting linear regression with our combination of features and comparing it with other regression methods.

3.2 XGBoost

Basak et al[3] compared XGBoost classifier with other tree based techniques to predict stock price of 2 S&P stocks i.e. facebook and apple and concluded that XGBoost achieved an accuracy of 94% on facebook stock and 77% on apple stock. These results were close to results obtained by Random Forest method i.e. 94% on facebook stock and 93% on apple stock, which outperformed all methods. The trends observed regarding the change in the accuracy with change in window-width for XGBoost is similar to the change in accuracy with random forest. Window-width here refers to the number of previous days of data used for prediction. This paper predominantly converted the stock prediction into a classification problem and used XGBoost Classifier, we will be using XGBoost regression in our method as building a model to predict continuous variables will be more robust as an approach. Further to this we will also include additional features mostly technical indicators.

3.3 Long Short-Term Memory Neural Network

Fischer et al[4] compared different deep learning methods such as deep neural networks (NN), gradient-boosted-trees and random forests. In a single-feature setting, the daily returns with respect to the closing prices of the S&P 500 from December 1992 until October 2015 are provided to forecast one-day-ahead for every stock the probability of outperforming the market. LSTM always outperformed random forests, traditional NNs and logistic regression. LSTM had an accuracy of 54.3% while random forest and NN around 53% and logistic regression around 52%.

Ghosh and Neufeld[5] used LSTM networks (more precisely CuDNNLSTM) as training methodologies to analyze effectiveness in forecasting out-of-sample directional movements of constituent stocks of the S&P 500 for intraday trading. The idea is to buy the 10 stocks with the highest probability and sell short the 10 stocks with the lowest probability to outperform the market in terms of intraday returns on each day. Dataset is created with non-overlapping testing period i.e. by dividing the dataset consisting of 29 years starting from January 1990 till December 2018, using a 4-year window and 1-year stride, where each study period is divided into a training period of approximately 756 days (≈ 3 years) and a trading period of approximately 252 days (≈ 1 year). Features for the model are determined to have 240 timesteps and 3 features, and train it to predict the direction of the 241st intraday return. Targets are chosen by dividing each stock at time t into 2 classes of equal size, based on their intraday returns. Class 0 is realized if for stock s is smaller than the cross-sectional median intraday return of all stocks at time t, whereas Class 1 is realized if for stock s is bigger than the cross-sectional median intraday return of all stocks at time t. The model is made up of 25 cells of CuDNNLSTM, followed by a dropout layer of 0.1 and then a dense layer of 2 output nodes with softmax activation function. The results indicate an average return of 3.84 per annum.

These work around LSTM have been an inspiration for us to choose LSTM as one of the techniques, we will combine a couple of dense layers along with some good features in our methodology.

The above mentioned works mostly look at historical data of all stocks to determine which stocks are gonna outperform others. In our method we will only focus on a few stocks and based on its previous history and some significant information determined by technical indicators and its correlated stocks, we will aim to predict the future price of a given stock.

4.0 Methodology

To determine the stocks on which we want to regress the existing dataset was subsetted into a dictionary by the sector to which they belong. Working with the data that is classified only by their respective sectors allowed for easier manipulation and feature engineering as will be outlined below.[6 7]

4.1 Feature Engineering

4.1.1 Technical Indicators

A range of technical indicators that are commonly found in financial literature have been tested with the following ones being selected for use in model inputs. These technical indicators were generated in conjunction with the Python TA-Lib package.

4.1.2 Simple Moving Average

The simple moving average (SMA) takes the average of the last n days of log-returns to use as a feature, where we have set n=20. The use of this indicator allows us to capture price trend through time defined as follows:

$$SMA = \frac{x_1+x_2+...+x_n}{n}$$

Eq 4.2

4.1.3 Exponential Moving Average

Exponential moving average (EMA) is a moving average that places more weight on more recent dates, it is commonly used in trading to generate buy or sell signals based on when there is a crossover between EMA and the actual price. The EMA takes two input parameters; a smoothing parameter k and number of days n. We have used n=20 days and k=2 as the parameter choices:

$$EMA_t = (logret_t \times \frac{k}{1+n}) + EMA_{t-1}(1 - \frac{k}{1+n})$$

Eq 4.3

4.1.4 Triple Exponential Moving Average

Triple exponential moving average (TEMA) was developed as a trend indicator to counter two key issues faced by other moving average equations; the issue of lag from oscillation studies when trying to make trading decisions as well as the volatility impact of minor price fluctuations. This is done by taking numerous EMAs of the original EMA and then subtracting a portion of the lag:

$$TEMA = (3 \times EMA_1) - (3 \times EMA_2) + EMA_3$$

Eq 4.4

4.1.5 Moving Average Convergence Divergence (MACD)

Moving Average Convergence Divergence (MACD) is a commonly used trend indicator that is created by subtracting a 26-period EMA from a 12-period EMA (*Eq 4.5*). It is commonly used by traders in financial markets in financial markets to determine if the market is strengthening (bullish) or weakening (bearish) and can be used by our model to infer information regarding the current behavioural characteristics of the time series.

$$MACD = 12Period\ EMA - 26Period\ EMA$$

Eq 4.5

4.1.6 Relative Strength Index

Relative strength index (RSI) is a metric within the range of [0,100] that is used to determine if markets are overbought or oversold and therefore likely to swing back the other way. It does this by taking into consideration the magnitude of recent price changes relative to historical price volatility.

$$RSI = 100 - \left(\frac{100}{1+\frac{Average\ gain}{Average\ loss}}\right)$$

Eq 4.6

4.1.7 Bollinger Bands

Similarly to the RSI bollinger bands are commonly used technical indicators to determine if the current market price is overbought or oversold and therefore likely to swing back the other way. Bollinger bands differ from RSI by placing upper and lower bands two standard deviations above the typical price (TP) from the last n days and creating a middle indicator of an n day SMA. If the SMA passes above the upper band it is considered overbought and an opportunity to

sell and vice-versa if it crosses the lower band. We have decided to use a 20 day time-period to assess the last n days standard deviation.

$$BB_{upper} = SMA(TP_n) + 2\sigma(TP_n)$$

Eq 4.7 *where TP can be calculated using Eq 4.9*

$$BB_{lower} = SMA(TP_n) - 2\sigma(TP_n)$$

Eq 4.8 *where TP can be calculated using Eq 4.9*

$$TP = \frac{high+low+close}{3}$$

Eq 4.9

4.1.8 Williams' %R

Williams' %R (WILLR) is again a metric used to determine if the market is overbought or oversold within a given timespan n. By contrasting the current price with highest and lowest prices in the allotted timespan it returns a number spanning [0,-100] where values above -20 are typically considered overbought and values below -80 are typically considered oversold. We have used 20 days as the timeframe over which we have evaluated this metric:

$$WILLR = \frac{High_{1:n} - Close_t}{High_{1:n} - Low_{1:n}}$$

Eq 4.10

4.1.9 Market Capitalisation and Binary Classification

In order to determine the stocks on which we seek to regress, market capitalisation of each stock was determined as previously defined. We chose to use the stock that has the largest market capitalisation in each sector as the dependent variables as these are the most established stocks within their field and less likely to be impacted by random volatility in prices. Market capitalisation was taken for the year prior to the commencement of the test set (2015). The chosen stocks by each sector were shown in *Table 4.1*.

Table 4.1 *Stocks with the largest market capitalization in each sector*

Sector	Company	NYSE Ticker
Consumer Discretionary	Walt Disney Co	DIS
Consumer Staples	Conagra Brands	CAG
Energy	Helmerich & Payne	HP
Financials	H & R Block Inc	HRB
Health Care	Cardinal Health Inc	CAH
Industrials	Deere & Company	DE
Information Technology	Microsoft	MSFT
Materials	Air Products & Chemicals, Inc	APD
Real Estate	Realty Income Corp	O
Telecommunications Services	AT&T Inc.	T
Utilities	Exelon Corporation	EXC

4.1.10 Rolling Correlations

When predicting the next day price of a company it is often useful to look outside the realm of data relating specifically to that company itself. Companies that operate within the same sector as the one in question are often highly correlated in their daily price fluctuations (*Fig 4.1*). For each stock we sought to forecast we used the previously generated market capitalisations to also take the next three largest stocks within the same sector and used their rolling 30 day correlation on daily log-returns as an input to the model. The intuition behind this is that the largest companies operating in the same sector are directly influenced by their relative performance and can be a useful source of information.

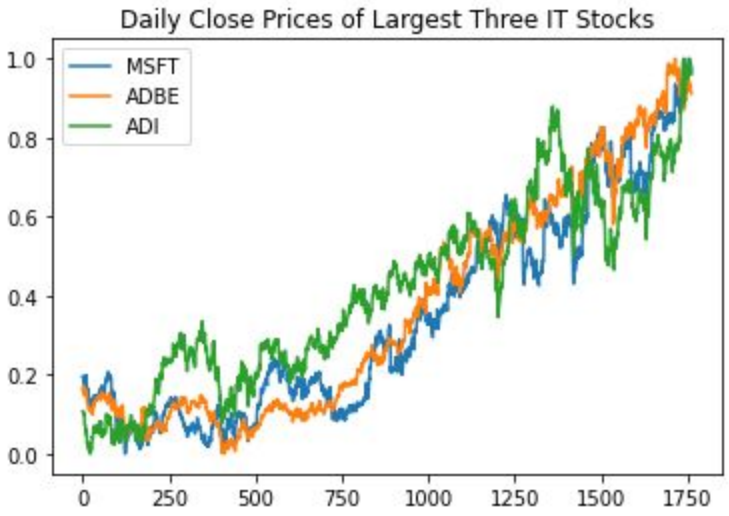

Fig 4.1 *Demonstration of correlation between stocks within the same sector*

4.1.11 Lagged Components

Because of the nature of time-series data on which this model is trained and tested, it is important to be mindful of the mechanisms of cross-validation that can both cause leakages in data across time as well as ruin the historical nature of the progression of variables across time in training from which information is obtained. In order to circumvent these issues and still be able to evaluate the chosen algorithms via 10-fold cross-validation, we implemented lagged components as new independent variables. This means that for every independent variable on which the model learns (excluding technical indicators which already account for behavioural characteristics through time as well as binary indicator variables), *n* new variables are generated to show the value of the previous *n* days.

Across each of the models, different values of *n* are evaluated for performance as outlined in section 5. For the linear regression model we determined 10 lags to be the optimal value, for the XGBoost we determined 4 lags to be the optimal value, and for the LSTM model 7 lags was optimal.

4.1.12 Log-Returns

It is known in financial markets that stock prices tend to follow a log-normal distribution.[8] Converting prices from their raw form to log-returns (*Eq 4.11*) has numerous underlying benefits for the purposes of analysis.

$$log(1 + r_t) = log(\frac{p_t}{p_{t-1}}) = log(p_t) - log(p_{t-1})$$

Eq 4.11 *where: p = price, r = return*

Based on the assumption of log-normally distributed prices, it naturally follows that $log(1+r_t)$ is normally distributed (where r_t is the return on the t'th day) which has much nicer properties for statistical analysis.

We also know from statistics that the product of normally distributed variables are non-normal which leads to many issues when considering factors such as compounding returns through time. Log transformation allows us to calculate compound returns via summation which maintains the normally distributed properties of the underlying data.

4.2 Data Pre-Processing

4.2.1 Data Normalisation:

All variables used for analysis other than binary variables and log-returns have been scaled via min-max normalisation according to the following equation:

$$x_{norm} = \frac{x_i - min(x)}{max(x) - min(x)}$$

Eq 4.12

This normalisation means that values are now scaled within the range of [0,1], reducing issues that can be caused by numerous variables that exist on different scales. Considering the implementation of cross-validation in this project, this scaling is done on a per-fold basis separately on both the training and test sets to prevent data leakages.

4.2.2 Principal Component Analysis (PCA)

PCA is a technique that projects data onto a space of reduced dimensionality, also known as the principal subspace, such that the variance of the projected data is maximized. The overarching objective of this transformation is to reduce data dimensionality, and consequently algorithmic complexity, whilst trying to maintain as much information from the input data as possible. This is achieved by computing the principal components of the dataset, which are a set of linear combinations of variables in the principal subspace such that the sum of squares between the origin and new subspace is minimized. The first principal component is the eigenvector corresponding to the largest eigenvalue and thus preserves the most variation from the original data:

$$Explained\ variance\ (\%) = \frac{Sum\ of\ explained\ variance\ in\ first\ i^{th}\ components}{Total\ sum\ of\ explained\ variance} \times 100$$

Eq 4.13

Thus far, a great number of 164 features were prepared for our regression models. We used PCA to transform our training data for every fold into a frame such that >99% of variance in our features were retained. We then transformed our test set by applying the same PCA of training data set, this is again to avoid data leakage.

4.3 Cross Validation & Mean Squared Error

To evaluate the performance of each model, our dataset was shuffled and split into 10 folds, with each fold serving as the testing set and the remaining nine folds forming the training set. MSE, which is the average squared difference between the predicted and observed data points, will be calculated for all 10 folds:

$$MSE = \frac{1}{N} \sum_{n=1}^{N} (y_n - t_n)^2$$

Eq 4.14

where N is the size of the testing set and y and t are the predicted and observed prices on the n^{th} data point, respectively. The average MSE across all folds were then compared between different regression models.

4.4 Models

4.4.1 Linear Regression

The first supervised algorithm we used to predict our closing price was linear regression, which involves a linear combination of input variables:

$$y = w_0 + w_1 x_1 + \ldots + w_D x_D$$

Eq 4.15

where $x_1, x_2, \ldots, x_D$ are the input features in D number of dimension and $w_0, w_1, \ldots, w_D$ are the coefficients of our linear function.[9] The coefficients are identified by fitting the function to the training data such that the residual sum of squares between the predicted and observed training data points are minimized.

4.4.2 XGBoost

The second supervised algorithm we explored to predict our closing price was XGBoost, which uses general principles of gradient boosting for regression.[10] Briefly, gradient boosting involves the creation of multiple additional decision trees, a supervised learning method that predicts variables according to decision rules inferred from the training set, based on the residual sum of squares calculated from previous trees. This correction of the residuals sum of squares process is repeated until the residual is fully minimized.

There exist benefits of XGBoost over gradient boosting. For instance, XGBoost creates trees in parallel as opposed to building trees sequentially, thereby speeding up tree construction. Additionally, XGBoost has built in cross-validation and regularization functions, removing the need for specifying boosting iterations, reducing overfitting, and allows a better control of model complexity. Furthermore, XGBoost utilizes the weighted quantile sketch algorithm, allowing

efficient handling of weighted and sparse data as well as optimizing external memory via careful disk space utilization.

4.4.3 Long Short-Term Memory Neural Network

NNs have been demonstrated to achieve significant results in the field of predictive analytics. A key drawback of a NN in time series forecasting is that the network does not retain a memory across time meaning that at each timestep the model re-evaluates given the current input data to produce a new forecast. A development in this field was the introduction of the Recurrent NN (RNN) which is a modified architecture of the traditional NN where information is passed through hidden layers (*Fig 4.2* and *4.3*).[11]

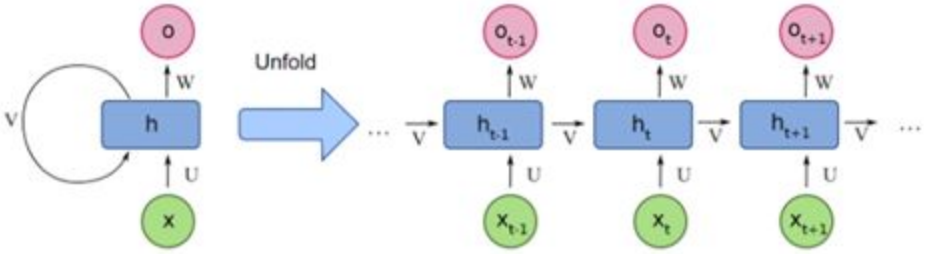

Fig 4.2 *Generic RNN architecture*

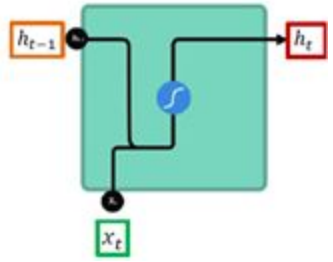

Fig 4.3 *RNN Hidden Layer*

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

Eq 4.16

Each hidden layer takes the input value(s) at the given timestep x_t as well as information from the prior hidden layer h_{t-1}. Parameters are applied over each of these inputs and a tanh activation function is applied to them before the are concatenated to form the current hidden layer value h_t which will serve as an input into the next hidden layer h_{t+1}. RNN models have been demonstrated to significantly enhance predictive performance compared to NNs, notably in the fields of NLP and Financial Market Predictions. There is a major drawback of RNN algorithms however, being the long-term dependency problem. Information that happened many timesteps ago becomes lost when repeatedly passed through many hidden layers. This is what led to the development of the LSTM algorithm.

The LSTM model is very similar to the vanilla RNN model however it contains a more complex hidden layer.[12] [13] The critical component of the LSTM cell is the constant horizontal line observed in *Fig 4.4* which is the memory pipeline. Additionally, the LSTM cell has the ability to add or remove information to this memory pipeline through a series of gates.

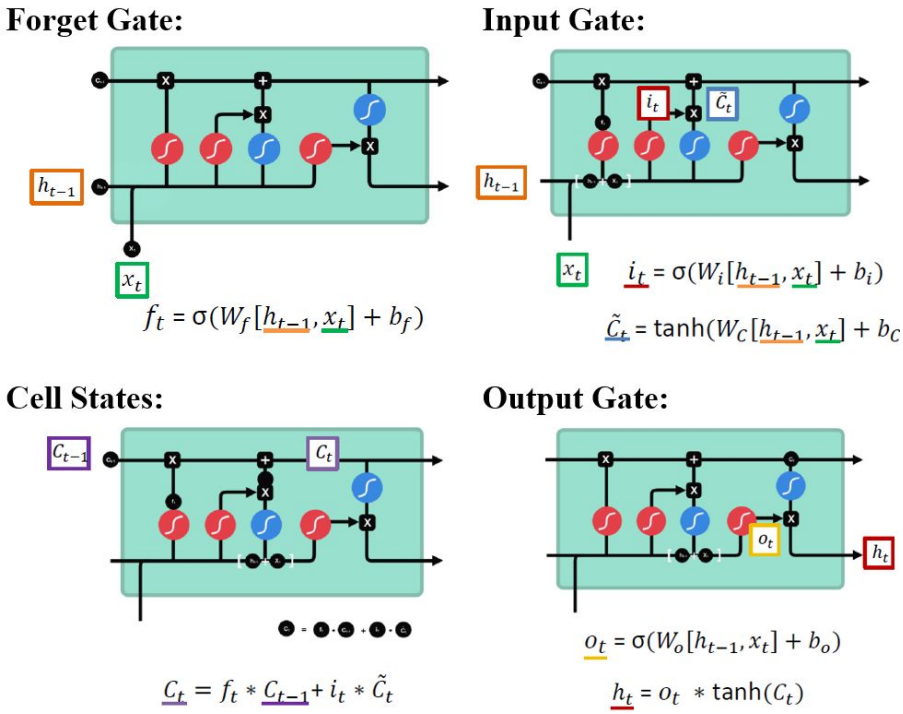

Fig 4.4 LSTM Hidden Layers

We have also added a dropout layer to our LSTM model to assist with prevention of overfitting.

5.0 Experiments and Discussion

We identified 1 large stock from each of the 11 sectors to perform our experiments. We have also added a 30 days rolling correlations of 3 stocks from each sector as summarized in *Table 5.1*. We have conducted all our experiments using a 10-fold cross validation approach on training data. We have kept 20% of the data as test data aside to measure the performance of our final model with best hyper parameters.

Table 5.1 *Stocks with the largest market capitalization in each sector as well as three stocks with the highest 30 day rolling correlations. All stocks are represented using its NYSE ticker*

Sector	Largest Stock	Correlated stocks		
Consumer Discretionary	DIS	AMZN	AN	AZO
Consumer Staples	CAG	CAG	CHD	CL
Energy	HP	APC	BHI	CHK
Financials	HRB	AIG	AIZ	AJG
Health Care	CAH	ABC	ABT	ALXN
Industrials	DE	ALK	ALLE	AME
Information Technology	MSFT	ADBE	ADI	ADS
Materials	APD	APD	AVY	BLL
Real Estate	O	AMT	AVB	BXP
Telecommunications Services	T	FTR	LVLT	T
Utilities	EXC	AEP	AWK	CMS

Our models took in inputs the features we have created (mentioned in section 4.2) and also historical data i.e. log-returns of open, high, low, close of previous days. We experimented with historical data of days 1-7 and 10. Historical data older than 10 days exhibited significantly poorer price forecast results (as indicated by greater MSE) across all models, and thus were excluded from our design. This meant that factors contributed to price vary over time.

5.1 Dimensionality reduction

One of our experiments also included dimensionality reduction using PCA. We have noticed that linear regression and XGBoost have slightly improved average MSEs of our selected 11 stocks on the PCA-transformed dataset following 10 fold cross validation (*Fig 5.1*). This transformed dataset explained around 99% of variance from the initial dataset, with principal components ranging between 16 and 17 among the 10 validation folds. However, price predictions from our LSTM model were significantly poorer on this transformed dataset. Thus,

the transformed dataset was used only for the linear regression and XGBoost models as it resulted in more precise price forecasts while also improving computation time due to reduced training features. The initial dataset without PCA was used for the LSTM model for the rest of our experiments. This experiment ensures that the neural network is capable of extracting hidden features when compared to other models.

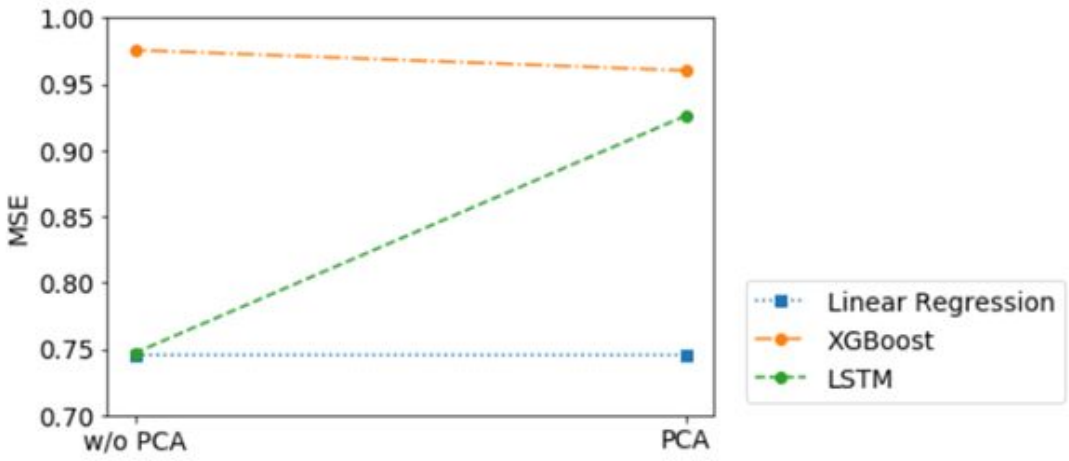

Fig 5.1 *Scatter plot comparing MSEs of our various models with and without PCA on data.*

5.2 Historical data

Among the historical data of days 1-7 and 10 used for our models, linear regression, XGBoost and LSTM performed best with 4, 10, and 7 days of history, respectively (*Fig 5.2*). LSTM exhibited the lowest average MSEs of 0.66, followed by linear regression and XGBoost at 0.74 and 0.94, respectively. We referred to this as *seq_len* in our code.

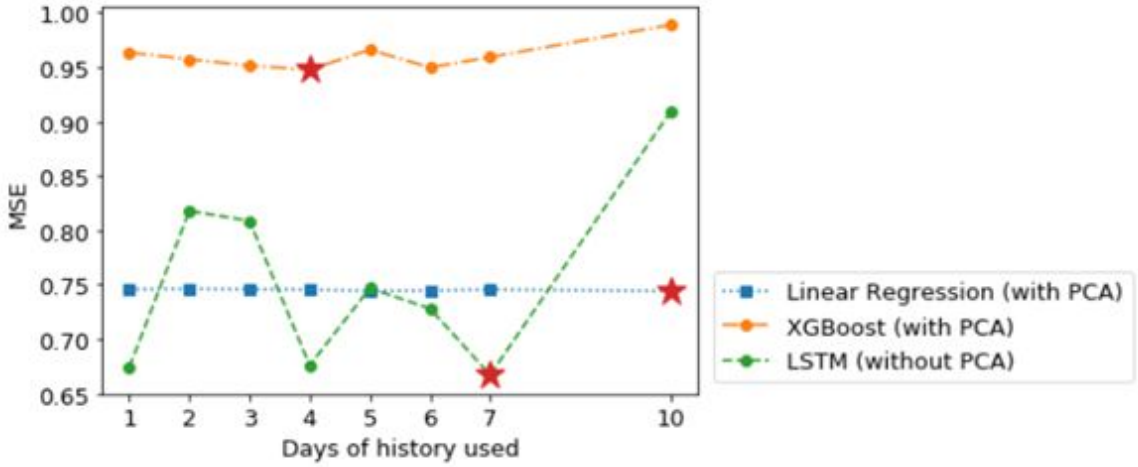

Fig 5.2 *Comparison of MSEs vs historical data for various models. Red stars indicate the lowest MSE for each model.*

5.3 Optimizing XGBoost

To identify the optimal hyperparameters that minimize MSEs for XGBoost regressor, we tested learning rates 0.5 and 1. A lower learning rate was found to increase robustness to overfitting although it may require more boosting rounds during training due to the more marginal improvements in each round. Additionally, it was also found that a maximum decision tree depth of 50 gave a low MSEs. While a deeper tree with more decision nodes may increase the complexity of the model, too many nodes may result in overfitting due to the reduced significance in tree splits and poorer robustness to noise. This was evident in our scenario as a maximum tree depth of >100 led to worse price predictions. Boosting rounds of 10 was identified to provide the lowest MSEs for XGBoost model with the updated optimal learning rate and maximum tree depth hyperparameters. These experiments were conducted with 4 days of historical data.

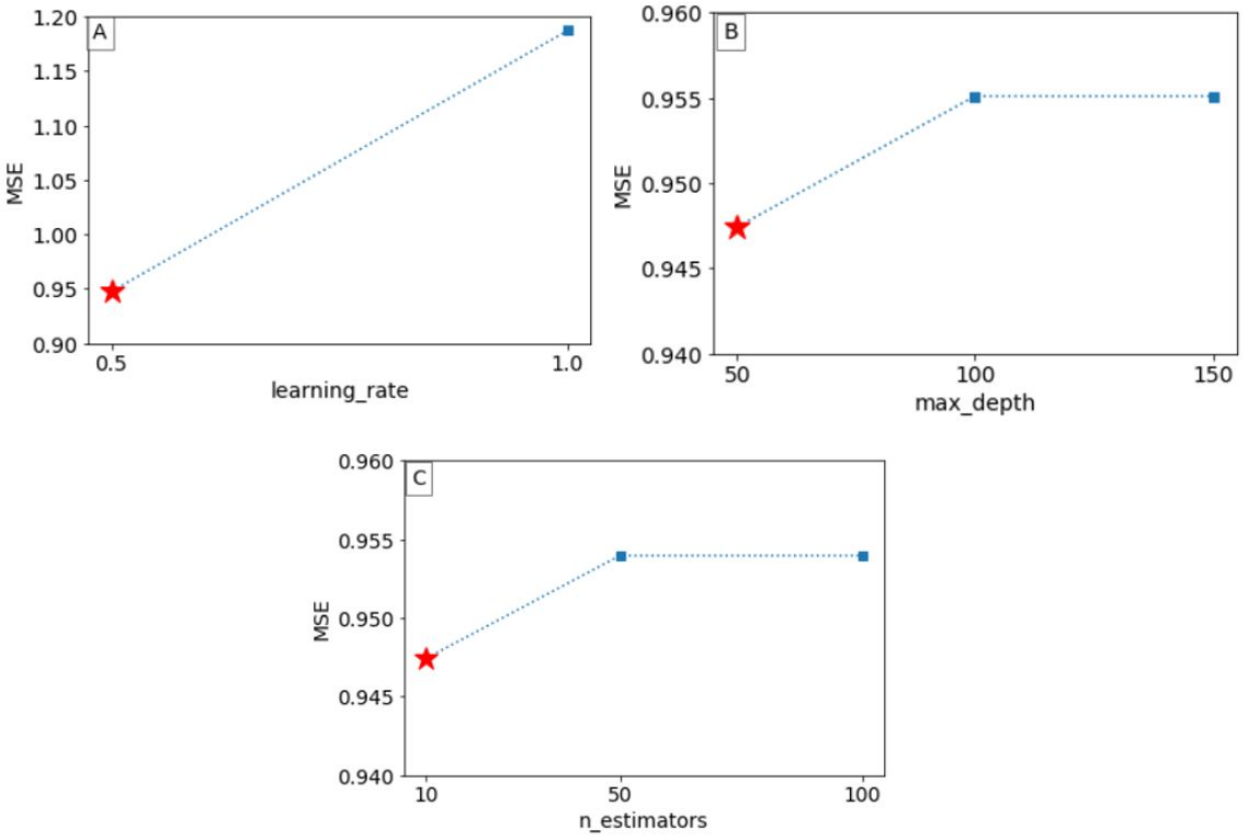

Fig 5.3 *Comparison of XGBoost performance over various (A) learning rates, (B) maximum decision tree depths, and (C) number of boosting rounds. Red stars indicate the lowest MSE.*

5.4 Optimizing LSTM

As expected, allowing 200 epochs in the LSTM model led to improved MSEs compared to that of 100 and 50 as this gave the model more iterations to fine tune weights (*Fig 5.4*).

Additionally, similar to XGBoost, a lower learning rate of 0.0001 also increased robustness to overfitting and provided reduced average MSEs in the LSTM model.

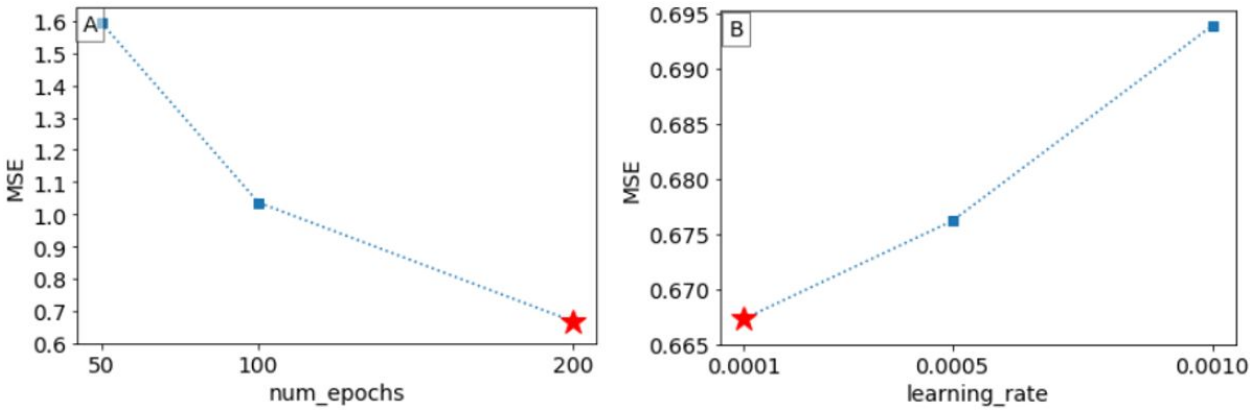

Fig 5.4 *Comparison of LSTM performance over various (A) epochs and (B) learning rates. Red stars indicate the lowest MSE*

We tuned other hyperparameters i.e. dropout rate and batch size to improve average MSE. Dropout is the random exclusion of several hidden units in the network during certain training iterations. These units will not be activated during feed-forward predictions and their weights will not be tuned during backpropagation. This prevents hidden units to be over reliant on one another, which may cause the model to be "fragile" and too specialized to the training dataset. Our 10 fold cross validations revealed a dropout rate of 0.2 had better performance than that of 0.05 and 0.1 *(Fig 5.5)*. It is clear from these experiments that regularisation methods help in improving NN performance.

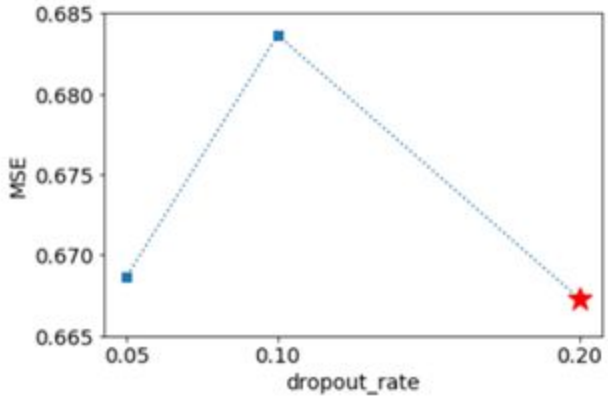

Fig 5.5 *Comparison of LSTM performance over various drop out rates. Red stars indicate the lowest MSE*

5.5 Code Upgrade

In an attempt to reduce the training time, we introduced batch size. Batch size is the number of observations used to train the model at one time. For instance, a batch size of 64 allows the model to train with the only 64 observations, then followed by another 64 observations, and so on. Using batches of data reduces computation resulting in less time and less

memory. Our experiments also revealed the same i.e. using a batch size of 256 in comparison with no batch reduced the training time from 473 seconds to 59 seconds i.e. speeding up the training process by 7 times as indicated in *Fig 5.6.* This result encouraged us to experiment with different batch sizes.

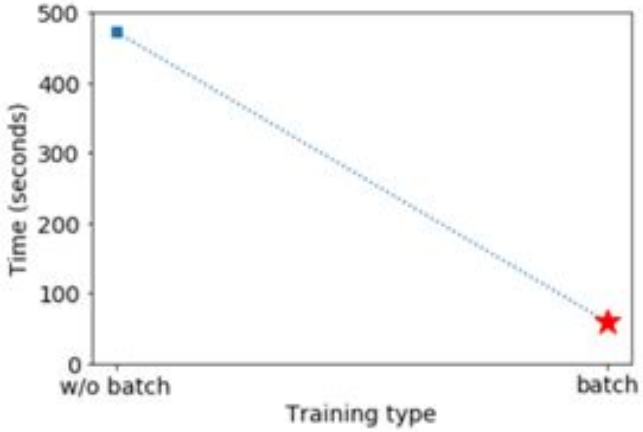

Fig 5.6 *Comparison of LSTM performance with and without batch. Red stars indicate the lowest MSE*

Lastly, our experiments with different batch size revealed that smaller batch sizes do not always lead to better predictions, as our LSTM model with a batch size of 256 produced the best prediction results (as indicated in *Fig 5.7*) compared to other batch sizes i.e. 128 and 64.[14]

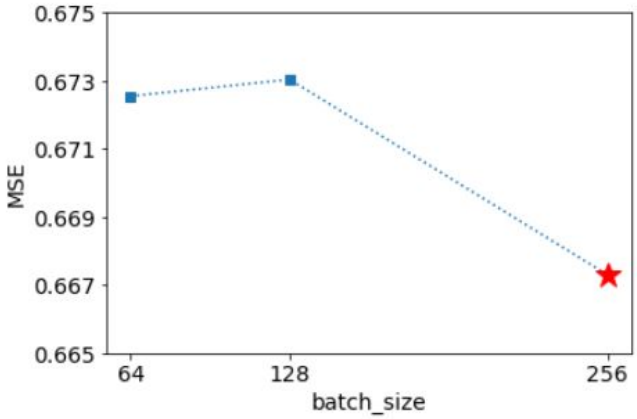

Fig 5.7 *Comparison of LSTM performance with different batch size. Red stars indicate the lowest MSE.*

These experiments revealed that LSTM without dimensionality reduction with 7 days of historical data and a bath size of 256 with a drop out of 0.2 and a learning rate of 0.0001 trained over 200 epochs produces best results. Our final model trained on 80% of the data produces an MSE of 0.692 on the test data.

6.0 Conclusion

The proposed method in this paper describes good machine learning techniques with a combination of advanced features (i.e. technical indicators) which generalises well over different stocks from various sectors which is not described in many research papers. We experimented and understood that a good selection of features along with modifications to the NN i.e. adding multiple layers, including drop out rate, and increasing neurons in the several layers will result in a good model. Further to this our understanding of core concepts has been strengthened for instance NN is good in identifying hidden features and hence reducing data dimensionality will not help in making the NN a better model as dimensionality reduction would remove features from the data. Our model was also able to generalise well with stock of different sectors, not many papers experimented with multiple stocks. This method can also be generalised to other markets like forex trading.

We also see a potential for future work by optimizing and expanding feature space by leveraging other python finance packages. We also see merits in evaluating other variants of LSTM such as gated recurrent units (GRU). We can also supplement our dataset with candlestick charts and then use a convolutional NN (CNN) in combination with LSTM, as this would help in identifying additional features not available in technical indicators.

References

1. Bischoff B. Adjusted closing price vs. closing price. Zacks Investment Research. 2019 Mar 31. Available from: https://finance.zacks.com/adjusted-closing-price-vs-closing-price-9991.html
2. Siew HL, Nordin MJ. Regression techniques for the prediction of stock price trend. IEEE. 2012 Dec 31
3. Basak S, Kar S, Saha S, Khaidem L, Dey R. Predicting the direction of stock market prices using tree-based classifiers. Journal of Scientometric Research. 2019 Jan; 47:552-67
4. Fischer T, Krauss, C. Deep learning with long short-term memory networks for financial market predictions. Journal of Economic Dynamics and Control. 2018 Oct 16; 270(2):654-669
5. Ghosh P, Neufeld A, Sahoo JK. Forecasting directional movements of stock prices for intraday trading using LSTM and random forests. 2020
6. Benediktsson J. Ta-lib. GitHub repository. 2020. Available from: https://github.com/mrjbq7/ta-lib
7. Sharma H. Technical analysis of stocks using ta-lib. Towards Data Science. 2020 Sep 5. Available from: https://towardsdatascience.com/technical-analysis-of-stocks-using-ta-lib-305614165051
8. Sharpe M. Lognormal model for stock prices. Available from: http://www.math.ucsd.edu/~msharpe/stockgrowth.pdf
9. Kapse AD. Get smarter in eda+ml modelling+stock prediction, version 7. Kaggle. 2020. Available from:

https://www.kaggle.com/akhileshdkapse/get-smarter-in-eda-ml-modelling-stock-prediction
10. Hallows S. Using xgboost with scikit-learn, version 1. Kaggle. 2018. Available from: https://www.kaggle.com/stuarthallows/using-xgboost-with-scikit-learn
11. Williams RJ, Hinton GE, Rumelhart, DE. Learning representations by back-propagating errors. Nature. 1986 Oct; 323(6088):533-36
12. Castilla P. Predict stock prices with lstm, version 10. Kaggle. 2016. Available from: https://www.kaggle.com/pablocastilla/predict-stock-prices-with-lstm?scriptVersionId=1008769
13. Malm, R. Ny stock price prediction rnn lstm gru, version 4. Kaggle. 2017. Available from: https://www.kaggle.com/raoulma/ny-stock-price-prediction-rnn-lstm-gru
14. Kandel I, Castelli M. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. ICT Express. 2020 May 5