Title: Ubiquant Stock Market Predication (Kaggle Competition)

G050 (s1720422, s2201031, s2195929)

Abstract

This paper aims to perform an analysis of methods used for tackling the issue of stock price prediction by using deep learning techniques deemed appropriate for the task. Stock price prediction is a difficult task that can't be generalized solely by linear classifiers. Given that machine learning has evolved and that there are deep learning solutions for time-series forecasting problems in domains such as voice recognition, it is appropriate to apply such a technique within the financial domain. Since this is a time-forecasting problem, RNNs are to be explored within this analysis. We will be analysing its effectiveness of a linear regression solution as a baseline and comparing it to a LSTM deep learning model. The aim of which is to see if there is any benefit.

Given a data set containing 300 obfuscated features representing econometric indicators, we investigate if an LSTM's ability to capture temporal features within data works well in this context where we don't know the true meaning behind each feature in the data set. To aid this comparison, we use a linear regression model as the baseline which uses the time ID column to separate the data into 5 folds. We compare it to a LSTM model which used the data-set that was separated and shuffled into a random 5 folds to find that the LSTM model generalises more consistently compared to the linear regression baseline which had heavily biased results.

1. Introduction

With the advancements in the technological realm in the last decade, from processing power to memory optimizations to enhancements in network communication capabilities, not only data scientists and machine learning engineers are exploring new frontiers, but it has also empowered incumbents and startups across industries to tackle existing problems and challenges using cutting edge, state of the art technologies.

Within the financial ecosystem, in particular, this phenomenon coupled with fierce competition has made existing banks, investment firms and hedge funds rethink their traditional stock trading strategies to stay ahead in the financial markets and maximize the returns for their financial investors. According to World Bank (Bank, 2020), the global capitalization of stock markets, which is a vital component of a country's economy, surpassed $93 trillion in 2020.

Stock Market Prediction (SMP) is an example of time-series forecasting of market data that promptly examines previous data and estimates future data values. However, SMP is not a simple task due to its non-linear, dynamic, stochastic, and unreliable nature (Rouf et al., 2021). Adding to this, the evolving market size and high velocity of trade execution, the once popular mechanism of SMP using investors' personal experience and familiarity with market patterns has gone obsolete. Furthermore, studies (Yeh & Hsu, 2014) suggest that SMP conducted using historical data may not be applicable to make meaningful returns because the stock price follows a random pattern.

Despite the use of many models and algorithms such as Regression Algorithms (RA), Artificial Neural Networks (ANN), Support Vector Machine (SVM), and Naïve Bayes (NB) (Rouf et al., 2021) for financial market trading in the past decade, companies and data scientists are looking at new approaches to improve the quantitative researchers' ability to forecast an investment's return. In section 5, we review published work around these techniques within the SMP domain to establish a better understanding of our work in a wider context.

Intrigued by this, we are participating in a stock market prediction competition hosted by Ubiquant Investment (Beijing) Co., Ltd (a leading domestic quantitative hedge fund based in China) in collaboration with Kaggle. The aim of undertaking this competition is to solve a real-world data science problem, building a model that forecasts an investment's return rate with as much accuracy as possible.

1.1. Research questions and objectives

Some of the research questions we have tried to address are as follows:

Does training the model on more recent time-series data result in better forecasting accuracy than training on as much time-series data as possible? We look into selecting periods of time-series data for training versus all time-series data), whilst making sure that the model generalizes well across all time periods.

Are there any features that are correlated with each other or are correlated with the target variable?

Does an LSTM model lead to better forecasting accuracy due to the model making use of future data to aid in predicting past targets during training?

Will using a feature extraction library such as tsfresh, followed by a basic MLP yield better results than feeding the existing features into a RNN model such as LSTMs or GRU?

Due to time constraints, the latter research questions can't be investigated but is of value in further research to determine if statistical features generated from a data set are of value for predicting stock prices.

2. Data set and task

2.1. Data

Ubiquant, the company hosting the competition in collaboration with Kaggle has provided the Ubiquant Market Prediction Dataset which contains features derived from real historic data from thousands of investments. We will be using it to predict the value of an obfuscated (concealed) metric pertinent for making trading decisions.

The dataset is made accessible in the form of a comma-separated-values (.csv) file called **train.csv**. It includes a unique identifier for each row **row_id**, the ID code for the time the data was gathered **time_id**, the ID code for an investment **investment_id**, 300 anonymized features generated from market data **f_0:f_299** and the obfuscated (concealed) metric pertinent for making trading decisions **target**. It is important to note that the **time_id** attribute which represents the ID code for the time the data was gathered is ordered, however, the real time between the time IDs is not constant and as mentioned by Ubiquant, it will likely be shorter for the final private test set than in the training set. Similarly, for the **investment_id** attribute, not all investments have data in all time IDs.

The rules of the Kaggle competition mandate using a python time-series API provided by Ubiquant, which ensures that models do not peek forward in time. The ubiquant/ API serves the test set data (one million rows) in batches, with all of the rows for a single time **time_id** per batch. The API uses less than 15 minutes of runtime for loading and serving the data and requires 0.25 GB of memory after initialization. The initialization step **env.iter_test()** requires more memory than that and hence, it is recommended that the model is loaded after making the API call. To run the example test set through the API offline without errors, we need Python 3.7 and a Linux environment.

An **example_test.csv** file is provided which includes random data to demonstrate what shape and format of data the API will deliver to our notebook when we make a submission which essentially has all the columns present in the **train.csv** file excluding the **target** attribute. An **example_sample_submission.csv** file is also provided so the publicly accessible copy of the API provides the correct data shape and format.

2.2. Data Preprocessing

Given that the data set of anonymised features and targets is approximately 18GB in size as a CSV file, reading it into a kaggle notebook isn't feasible due to the likelihood of the notebook crashing. To mitigate the issue, a columnar-based file storage format, Parquet, is being used to efficiently compress the data whilst still allowing the full data set to be queried without any space or memory issues. The result is a new compressed data set of 5.5 GB in size containing the same data types and accessibility as the original data.

Initially K-fold cross-validation was used to separate the data set into 5 folds where 1 fold was used for validation and the remaining 4 folds were used for training. This process was iterated and the folds were created based on their time IDs for the baseline model.

For the LSTM model, the same K-fold cross validation methods was used however instead of using Time Id to separate the ordered data set, it was randomly shuffled and a random seed was used to separate the data into 5 folds.

2.3. Task

The task of the competition is to estimate the correlation between the obfuscated econometric indicator features and the target variable. The targets are also obfuscated but is likely representative of price volatility of an asset. The aim of the competition is to predict the correlation between the features and target to a high degree. This is being evaluated on Kaggle public leaderboards using Pearson's Correlation Co-efficient between the actual target values and the predicted targets obtained by our model.

2.4. Evaluation

As part of the competition rules, our submissions will be evaluated based on the mean of the **Pearson Correlation Coefficient** for each time ID.

Given that the submission is made using the provided python time-series API, when we submit our notebook, it will be rerun on an unseen test data. We must meet the following criteria, in order to make a successful submission:

- CPU Notebook <= 9 hours run-time
- GPU Notebook <= 9 hours run-time
- Internet access disabled
- Freely & publicly available external data is allowed, including pre-trained models

This is also a forecasting competition, where the final private leaderboard will be determined using data gathered after the training period closes, which means that the public and private leaderboards will have zero overlap.

2.5. Exploratory Data Analysis

We performed Exploratory Data Analysis (EDA) in order to gain key insights from the data before building our ML models. This gave us an idea of how the data was structured amongst the large number of features and groups present within the data set.

2.5.1. Observations, Time Steps and Assets

- Number of observations: 3,141,410
- Number of time steps: 1,211
- Number of assets: 3,579 (range from 0 to 3773

We find that the range of assets (investments IDs) goes beyond the number of assets themselves. According to the competition host, this holds true, since the assets will change in part in the test set and the **investment_id** may exceed this range.

2.5.2. Target Analysis

In 1, Target by Asset Distribution graph, we first plot a histogram and are able to observe that there are some discontinuities present in the first section of the time. Upon further exploring the data set, we learn that the assets are distributed in a different way. There are some assets that are more frequently observed than the others. Hence, this gave us a glimpse into utilizing a good cross validation and modelling strategy.

In 2, Mean of Target Distribution graph, we are able to observe outliers in the average of mean target by asset where some assets have quite negative average target (-0.4 area) and whilst some have quite positive (+0.8 area). Overall the average mean target by asset is slightly negative (-0.0231) and shows a bell-shaped distribution.

In 3, Standard Deviation of Target Distribution graph, we plotted the average of mean standard deviation (std) by assets. We are able to observe interesting patterns since the graph is skewed towards the right, with some assets having more std (up to 2.5), whilst on the other side, there are some assets with std almost at zero.

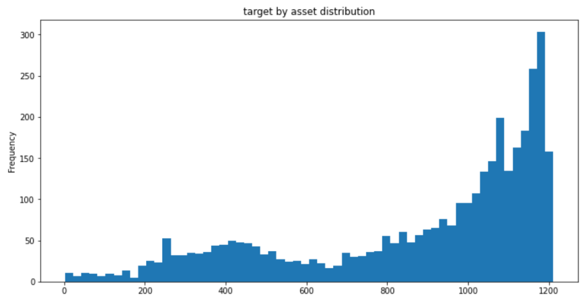
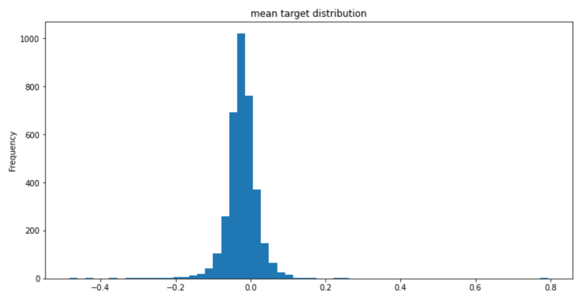
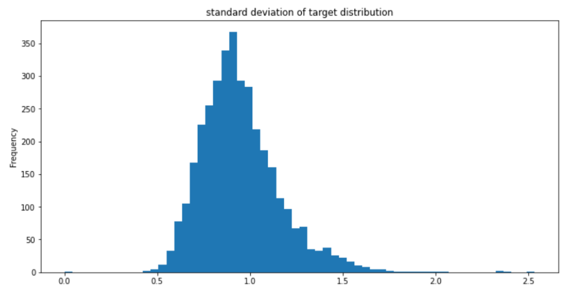


Figure 1. Target by Asset Distribution



```
Mean of mean target: -0.02310
```

Figure 2. Mean of Target Distribution



```
Mean of std target:  0.94790
```

Figure 3. Standard Deviation of Target Distribution

In 4, Number of Unique Assets, Average Target and Std of Target by Time graph, we tried looking at the number of unique assets by time first. We observe randomness in terms of the number of the assets present at each time step. Another insight which we were able to glean was that as the time step increases, the number of assets also grow by almost one third. We further plotted the number of assets by time alongside the average target by time, and it became clearer that when there are less assets, the target oscillates more with prevalently higher targets.

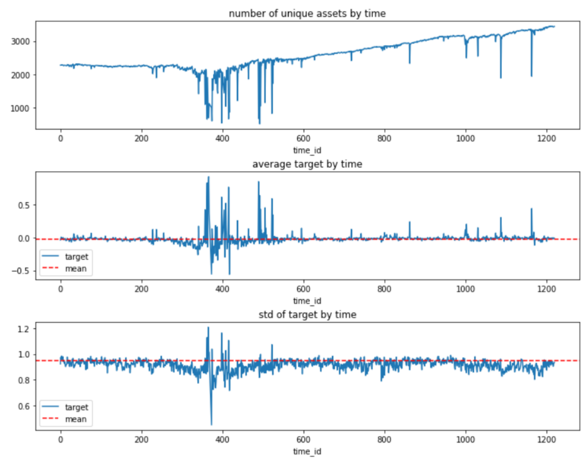


Figure 4. Number of Unique Assets, Average Target and Std of Target by Time

In 5, Target Mean and Standard Deviation by Time graph, we aimed to explore more closely the relationship between

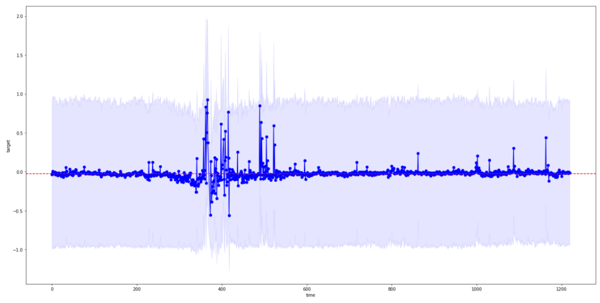


Figure 5. Target Mean and Standard Deviation by Time

the target variable and time feature. From the graph, we have been able to observe how the target is forced to mean zero and unit standard deviation. This is representative of our hypothesis that the target variable represents the market volatility of an asset as it is shown to be mean reverting. This analysis may be useful to help determine which features are likely to correlate with the target values spiking.

3. Methodology

3.1. Baseline Model - Linear Regression

A linear regression model was chosen as the baseline due to it's ability to predict target values and fit to numerical data. Linear regression works by fitting a line of best fit to the data-set's features. It is an ideal baseline as a stock price's graph is not linear and doesn't only go up, so this model will serve as an accurate comparison to see if the model we develop is performing well. A model using linear regression can be characterised by the following equation:

$$\hat{y} = b_0 + b_1x_1 + b_2x_2 + ...b_ix_i \tag{1}$$

Where $\hat{y}$ is the target variable being predicted, b_0 is the y-intercept of the line of regression and b_ix_i is the slope of each line of regression multiplied with the value of the corresponding independent variable (300 features of the data set) .

The performance of the baseline will be measured using the Mean-squared error between the predicted values and actual target label. It is characterized by the following equation:

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

Where n is the total number of samples in the data-set, Y_i is the actual target label and $\hat{Y}_i$ is the predicted target label. This metric was chosen as the loss function as it is the de-fato standard for training deep-learning algorithms on. Despite the competition's evaluation metric being the Pearson's Correlation Co-efficient, using MSE yields better results as the loss function. The reason for this is because MSE can be optimised easily and will have a single olution. Pcc on the other hand has an infinite amount of solutions so it is likely the gradient would be undtable when used during training. Furthermore, each batch's bias and scale

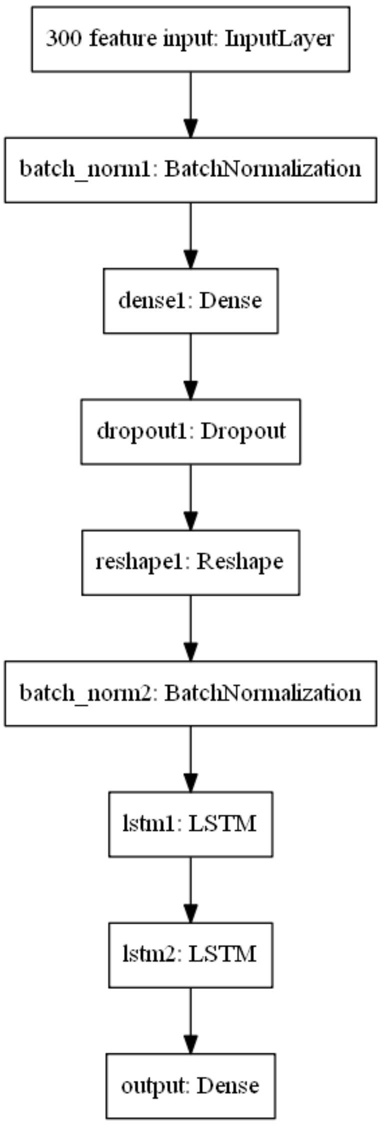

300 feature input: InputLayer
batch_norm1: BatchNormalization
dense1: Dense
dropout1: Dropout
reshape1: Reshape
batch_norm2: BatchNormalization
lstm1: LSTM
lstm2: LSTM
output: Dense

Figure 6. Training Pipeline

mau be different which would further cause the gradeitn to become unstable if PCC was used as the loss function. Hence MSE was chosen instead.

In order to obtain a reliable estimate of the model's performance, K-fold cross validation was used to evaluate the MSE by training and testing the model across 5 folds of the data-set.For the baseline model, the folds wer created based on the ordered investment IDs of the data, which are also time-ordered. This was done as we were investigating whether training our model on different portions of the time-series data has any effect on the prediction ability for more recent data versus training the model on all of the time-series data.

3.2. Model Selection - LSTM

An LSTM model was chosen to develop upon the baseline model due to it's solution towards the probelm of DNNs having short-term memory. DNNs have great ability when it comes to learning pattern present between features and targets, however for complex models, the vanishing gradient problem becomes an issue. LSTMs are designed to counter this problem by using a forget state, input state and cell state to preserver important features between layers within the RNN. This is especially useful when it comes to a time-forecasting problem such a stock price prediction as there may be patterns in the past that are important for predicting future price movements (Hochreiter & Schmidhuber, 1997).

Given that not all investment IDs are within the test set, the investment ID was not considered as a feature for the time being. To tackle the problem, an LSTM Deep-Learning model was chosen initially due to its ability to capture temporal features within sequential data. Since the time IDs aren't at constant intervals between each other, this wasn't selected as an input for the model. The inputs of the model consists of the 300 obfuscated features that are representative of econometric indicators. The dense layer of the model helps to create 256 new features which work as sequential inputs to the LSTM model, hence capturing the temporal qualities present within the 300 features.

The training pipeline illustrated in 6 represents the 300 different features being used as the inputs of a 9-layer NN (1 input, 7 hidden and 1 output layer). The Batch Normalization input layers are being used to standardize the inputs to each layer to allow for a stable learning process. The dropout layers are being used to prevent over-fitting of the model. Both of these techniques are used to reduce the likelihood of the vanishing gradient problem. The LSTM layers are being implemented to capture the temporal qualities preserved within the time series data to identify useful features from it. The model is being trained using learning rate of 0.001 and a dropout rate of 0.1 as this yielded stable performance during training.

To evaluate the performance of the model, MSE was also used as the loss function and K-fold cross-validation was used to ensure the results were reliable. The data-set for cross-validation was nit split based on the investment IDs, this is because we wanted to see how the model performs when the data is shuffled into random folds irrespective of that feature.

4. Experiments

4.1. Motivation & Baseline

To compare how effective deep learning models are in the context of stock price prediction, we implemented a baseline model using linear-regression for the data set. For our experiments, we used the mean squared error (MSE) as the evaluation metric since the mean-squared error as the error function is ideal due to the nature of the problem being regression based. The task at hand is to predict as close to a target value as possible, hence MSE serves as an ideal metric for doing so.

4.2. Linear Regression Model

4.2.1. Description

To create the baseline linear regression model, the training and test data was split into 5 folds based on their Investment IDs. For each fold, the linear regression model was trained on the rest of the data and tested on the fold.

4.2.2. Results

The results for the MSE values of the Baseline Linear Regression model are displayed in 1. When taking the average of the 5 folds, we obtain an average validation MSE of **0.8319**. This baseline result will be compared later on within this paper once the results for the LSTM model have been obtained.

Fold #	Final Validation MSE
1	0.9804
2	0.8400
3	0.8108
4	0.7642
5	0.7639

Table 1. Table showing MSE values for the Baseline Linear Regression model

4.2.3. Interpretation and Discussion

From the results in 1, we can see that for the later folds, the mean squared error drops significantly. This is indicative that the predictive accuracy is greater when the model is tested on more recent investments with a higher time ID. Therefore, in order to help the model generalize better than this baseline, for the LSTM model, the investment ID will not be used as a feature or for determining how the folds are split. This will ensure that the model generalizes well.

4.3. LSTM Model

4.3.1. Description

For the second experiment, an LSTM RNN was constructed which contained 7 hidden layers. These hidden layers comprised of Batch Normalization and dropout layers which were implemented to keep the model stable to avoid the vanishing gradient problem. Furthermore, it consisted of LSTM layers which were being used to capture the temporal qualities within the data which would help the model learn the features better. The learning rate was set to 0.001 and dropout rate set to 0.1. These values were chosen as these parameter values allowed the model to remain stable.

To ensure the results obtained have integrity, 5-fold cross validation was used across the training and validation set. In this version, the folds were selected based on a random seed instead of using the time-ordered investment ID. This allowed us to get a better idea if the model is generalizing accurately or not. For all experiments, 10 epochs and a batch size of 128 were used during training due to the limited amount of time available.

4.3.2. Results

For the LSTM model, the average MSE obtained across all 5 folds on the validation sets was **0.8606**. This is slightly higher compared to the average from the baseline obtained for linear regression which was 0.8319.

Fold #	Final Training MSE	Final Validation MSE
1	0.8159	0.8349
2	0.8176	0.8305
3	0.8188	0.8269
4	0.8183	0.8274
5	0.8169	0.8330

Table 2. Table showing MSE values across 5 folds for the implemented LSTM model.

4.3.3. Interpretation and Discussion

From an initial glance at the results in 2 it may look like the baseline logistic regression performs better but that is not the case. The reason the LSTM model has scored a higher MSE is due to the method for K-fold cross validation being different. In this experiment, the folds were split and shuffled randomly, the time ID and investment ID were not taken into account. The result of this was consistent validation errors across all folds unlike the baseline model which had largely varying validation errors for each fold.

5. Related work

5.1. Literature Review

There are many machine learning models used in previous literature for stock market prediction (SMP). Four of the most commonly used techniques for SMP are Artificial Neural Networks (ANNs), Support Vector Machine (SVM), Genetic Algorithms (GAs) and Naïve Bayes (NBs). In this section, we review published work around these techniques within the SMP domain to establish a better understanding of our work in a wider context.

5.1.1. Artificial Neural Networks (ANNs)

One of the most recognized approaches that primarily focuses on stock market prediction is using Artificial Neural Networks (ANNs), which are biological inspired computational models (Sharma et al., 2021). In the network, a series of neurons (nodes) are connected into layers starting with an input layer and ending with an output layer. The network acquires knowledge about the problem and makes predictions by feeding various example models into the network. During the learning process, the network gradually improves its performance via adjusting weights between connected nodes.

In 2004, Jasic and Wood (Jasic & Wood, 2004) proposed a model to predict daily index returns using several global stock markets with the focus on supporting profitable tradings. They introduced a new method based on univariate neural networks using untransformed data inputs to provide short-term predictions of the stock market indices returns. Their study uses the daily closing values of S&P 500, the German DAX Index, the Japanese TOPIX index, and London's Financial Times Stock Exchange Index (FTSE) over the period 1965-1999. While the results have demonstrated the feasibility, the statistical significance and the potential profitability of SMP using their model, the performance on non-normalized data or rescaled data was still unclear.

Another stochastic time effective neural network model to disclose the predictive relationships of numerous financial and economic variables was introduced by Liao and Wang in 2010 (Liao & Wang, 2010). In their model, historical data was assigned with weights to indicate how close they are to the present. The nearer the historical data is to the present, the stronger the impact the data have on the model. The performance of their model is tested from each trading day for 18 years from 1990 to 2008 over several stock markets including the Shanghai and Shenzhen Stock Exchange Stock A Index (SAI), Stock B Index (SBI), and the Hang Seng (HIS), Dow Jones Industrial Average (DJIA), NASDAQ Composite (IXIC) and S&P500. However, their experiments were mainly focused on the Chinese stock market.

Three years later, the ANN model is further refined by Chavan and Patil (Chavan & Patil, 2013). They attempted to find the most vital input parameters that produce optimal prediction accuracy. They conclude that technical variables were widely used in machine learning models, while microeconomic variables were more commonly seen in predicting stock market index values. An analysis of deep learning networks for SMP was conducted by Chong et.al. in 2017 (Chong et al., 2017). Deep learning networks extract features from a large set of unprocessed data without relying on prior knowledge and thus have the advantage of high frequency. In addition, they were also able to extract further information from the residuals of the autoregressive model and improve prediction performance.

5.1.2. Support Vector Machine (SVMs)

Support Vector Machine (SVM) offers an alternative method to ANNs for improving SMP accuracy. This technique mainly involves supervised learning, limits errors and augments geometric margins. Training examples are considered as being part of one category or another. An SVM model represents the examples as nodes in a space to create a gap between the categories that are as wide as possible. New examples are classified based on the category to which they are most likely to belong.

A unique study led by Schumaker and Chen (Schumaker & Chen, 2009) in 2009 used an SVM model combined with textual analysis exploring the impact of news articles on stock prices. They developed a predictive machine learning approach for financial news article analysis using several different textual representations: Bag of Words, Noun Phrases, and Named Entities. In their study, they found that the model containing both article terms and stock price when the article was released provided the closest estimate to the actual future stock price and the highest returns. While their findings are promising, they also acknowledged that their dataset might be relatively small and larger datasets may lead to different results.

Two years later, Yeh et. al. (Yeh et al., 2011) further addressed the problem with kernel function hyperparameters. Traditionally, a hyperparameter is a parameter whose value is fixed before the start of a learning process. They develop a two-stage multiple-kernel learning algorithm by incorporating sequential minimal optimization and the gradient projection method. Experimental results using data from the Taiwan Capitalization Weighted Stock Index proved that the modified method performs better than other methods. The daily stock closing price datasets used for training, validating, and testing the model were from October 2002 through March 2005.

Das and Padhy (Das & Padhy, 2012) investigated the performance between SVM and backpropagation (BP). Their experiments were run on datasets from the National Stock Exchange (NSE) of India Limited for the period from 2007 to 2010. The implementation is completed using MATLAB and SVM Tools (LS-SVM Toolbox). In their study, they managed to prove that in terms of SMP, the SVM model is superior to the BP technique

5.1.3. Genetic Algorithms (GAs)

As illustrated above, SMP using ML algorithms such as ANNs or SVMs have had some success but, over time, there appears to be an increasing interest in trying to further improve results using multi-technique approaches. One alternative machine learning method that has the potential to accomplish this is to incorporate genetic algorithms (GAs) with either ANNs or SVMs to reduce single technique limitations.

GAs are heuristic approaches to problem-solving that mimic the natural evolution process. The algorithms apply the concept of natural selection to select the optimal possible solution. The evolution process starts with a randomly generated set of solutions. In each iteration, the fitness of each solution is measured by an objective function. Solutions with high fitness are retained and combined with other high fitness solutions to create a new generation of solutions that retains some of the characteristics of the two original solutions. This process continues until a certain threshold has been reached, or the solutions have reached a desired level of fitness.

Numerous studies have shown the enhancement GAs have to SMP accuracies. For example, Kim et.al. (Kim et al., 2017) in 2017 developed an intelligent decision support system for stock trading. Their study employed rough sets and GA for non-linear and complex stock data to find the characteristics that can be used to generate the optimal trading rules. These rules, in return, are applied to generate optimal buying or selling strategies. However, their approach requires several data reduction techniques, such as object (i.e., instance) and attribute selection methods beforehand because their model does not have optimal performance on larger and more complex datasets.

5.1.4. Naïve Bayes (NBs)

Naïve Bayes (NBs) is a classification method that classifies the data points based on the Bayesian Theorem of probability. This classification method is of extremely high speed and can be scaled over large datasets. This classification approach has been used widely for SMP. For instance, Yu et. al. (Yu et al., 2013) employed the Naïve Bayes algorithm for the sentiment analysis of textual data from multiple sources in 2013. The authors compared the effect of conventional and social media data sources on different companies and their interrelatedness. However, the success of their model highly relies on the quality of information availability and information processing.

Two years later, Patel et. al. (Patel et al., 2015) compared the performance of ANNs, SVMs, random forest and NBs with two different approaches to process input data in forecasting the direction of movement of the stock and stock price index. The first approach is to compute ten technical indicators from the stock trading data (open, high, low and close prices) and the second approach is to represent the technical indicators as trend deterministic data.

In the study, they discovered that the first approach of random forest outperforms the other three prediction models on overall performance. Additionally, the performance of all the prediction models is improved when technical indicators are represented as trend deterministic data. However, they failed to show whether their proposed algorithm can be deployed in other markets as well as with different financial instruments.

In 2020 Bhandare et. al. (Bhandare et al., 2020) used the Naive Bayes classifier to provide analyse and quantify the performance of stock market analysts by providing ratings. The results indicated that the performance of the system is optimal when Gaussian Naive Bayes Classifier was used. However, their rating system only incorporates five values, which may limit its applicability.

6. Conclusions

From the experiments, we can see that a RNN-based deep learning approach towards stock price prediction is definitely feasible given that the model was only trained for 10 epochs yet the final validation accuracy was still coming down. With further hyperparamater tuning and if trained for longer, the model would generalise better as well as outperform traditional ML methods.

It could also be observed that there is some bias present when training on datasets that involve time-series forecasting. This is because the model would perform better at predicting more recent targets compared to older ones. Using an LSTM model helped to eliminate the likelihood of this occurring due to its ability to capture temporal qualities within the data.

For future research, further experiments can be performed such as investigating whether using a hybrid combination of a linear classifier followed by a non- linear classifier would yield better performance. Another point that would be valuable to investigate would be whether the investment and time ID have a correlation with the target variable as the LSTM model used did not consider them as features. Ideally, a higher number of epochs should be used in the comparison between models but due to time constraints, this wasn't possible.

References

Bank, World. Market capitalization of listed domestic companies, 2020. URL https://data.worldbank.org/indicator/CM.MKT.LCAP.CD.

Bhandare, Yash, Bharsawade, Sumit, Nayyar, Dhurv, Phadtare, Omkar, and Gore, Deipali. Smart: Stock market analyst rating technique using naive bayes classifier. In *2020 International Conference for Emerging Technology (INCET)*, pp. 1–4. IEEE, 2020.

Chavan, Prashant S and Patil, Shrishail T. Parameters for stock market prediction. *International Journal of Computer Technology and Applications*, 4(2):337, 2013.

Chong, Eunsuk, Han, Chulwoo, and Park, Frank C. Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications*, 83:187–205, 2017.

Das, Shom Prasad and Padhy, Sudarsan. Support vector machines for prediction of futures prices in indian stock market. *International Journal of Computer Applications*, 41(3), 2012.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

Jasic, Teo and Wood, Douglas. The profitability of daily stock market indices trades based on neural network predictions: Case study for the s&p 500, the dax, the topix and the ftse in the period 1965–1999. *Applied Financial Economics*, 14(4):285–297, 2004.

Kim, Youngmin, Ahn, Wonbin, Oh, Kyong Joo, and Enke, David. An intelligent hybrid trading system for discovering trading rules for the futures market using rough sets and genetic algorithms. *Applied Soft Computing*, 55: 127–140, 2017.

Liao, Zhe and Wang, Jun. Forecasting model of global stock index by stochastic time effective neural network. *Expert Systems with Applications*, 37(1):834–841, 2010.

Patel, Jigar, Shah, Sahil, Thakkar, Priyank, and Kotecha, Ketan. Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert systems with applications*, 42(1):259–268, 2015.

Rouf, Nusrat, Malik, Majid Bashir, Arif, Tasleem, Sharma, Sparsh, Singh, Saurabh, Aich, Satyabrata, and Kim, Hee-Cheol. Stock market prediction using machine learning techniques: A decade survey on methodologies, recent developments, and future directions. *Electronics*, 10(21), 2021. ISSN 2079-9292. doi: 10.3390/electronics10212717. URL https://www.mdpi.com/2079-9292/10/21/2717.

Schumaker, Robert P and Chen, Hsinchun. Textual analysis of stock market prediction using breaking financial news: The azfin text system. *ACM Transactions on Information Systems (TOIS)*, 27(2):1–19, 2009.

Sharma, Sparsh, Ahmed, Suhaib, Naseem, Mohd, Alnumay, Waleed S, Singh, Saurabh, and Cho, Gi Hwan. A survey on applications of artificial intelligence for pre-parametric project cost and soil shear-strength estimation in construction and geotechnical engineering. *Sensors*, 21(2):463, 2021.

Yeh, Chi-Yuan, Huang, Chi-Wei, and Lee, Shie-Jue. A multiple-kernel support vector regression approach for stock market price forecasting. *Expert Systems with Applications*, 38(3):2177–2186, 2011.

Yeh, I-Cheng and Hsu, Tzu-Kuang. Exploring the dynamic model of the returns from value stocks and growth stocks using time series mining. *Expert Systems with Applications*, 41(17):7730–7743, 2014.

Yu, Yang, Duan, Wenjing, and Cao, Qing. The impact of social and conventional media on firm equity value: A sentiment analysis approach. *Decision support systems*, 55(4):919–926, 2013.