Recurrent Neural Networks on Stock Price Prediction

Table of Contents

Executive Summary

The report investigates five different academic articles that revolve around predicting the future stock prices of industries, companies, and markets through utilizing Recurrent Neural Networks (RNNs). Through the analysis of each article, it was evident that the LSTM model was typically the most accurate at predicting the future stock price.

The five academic articles were analyzed through investigating the data and timeframe utilized, metrics of success, final model performance, and the structure of the RNN model. From these five articles, the second article was chosen as a basis to conduct research on whether or not the RNN structure is viable to use on a different dataset or would the RNN structure be completely hyper tuned for that specific dataset.

Through investigating the five articles, it was evident that the multi-layer perceptron (MLP) was able to outperform the LSTM within the second article. This could be a result of poorly constructing the RNN models as the other articles the LSTM was the most accurate model predictor.

Furthermore, the second article was utilized as a basis for constructing three different RNN models, of which were, RNN, LSTM, and GRU. The models built according to the second article were not able to accurately predict the stock price of the AMP company stocks. This led to building three specifically tuned models for the AMP stock price prediction. To which the LSTM was the most accurate through investigating the MSE and graphs.

This report shines light on how LSTM models can be extremely powerful predictor models to predict the future stock prices of industries, companies and markets.

Academic Article 1: Pearson Correlation Coefficient-based performance enhancement of Broad Learning System for stock price prediction
Article authors: Li, Guanzhi; Zhang, Aining; Zhang, Qizhi; Wu, D
Year published: 2022-03-16

Data and Timeframe Used

The research paper by the authors mentioned above revolved around the Shenzhen and Shanghai stock exchange market, where data points of approximately 2500 days were recorded. The entire dataset was spilt into two parts, where the first part consisted of 80% for the training data and the remaining 20% were used for the test data. The dataset consisted of stock prices from September 2010 to September 2021. The dataset was utilized as research data

From the knowledge gathered from the research data, a second dataset was collected. The second dataset points were collected from multiple sources in order to improve the forecasting ability. Furthermore, the historical data and financial indicators of each stock were retrieved from BaoStock's API interface. Moreover, the datapoints collected in regards of the Shenzhen Securities Component index (SZI) and Shanghai Securities Composite Index (SSEC) were collected from the NetEase website.

Metric of Success

The research paper consists of ten different machine learning methods, of which were, Support Vector Regression (SVR), Adaptive Boosting (Adaboost), Bootstrap aggregating (Bagging), Random Forest (RF), Gradient Boosting Decision Tree (GBDT), Multi-layer Perceptron (MLP), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and Broad Learning System (BLS).

As the research paper consists of ten different machine learning methods, only the LTSM, best and worst models will be inspected in regards of the metrics of success. This is due to the fact that analyzing all ten different models would create a lot of noise that is not needed.

Furthermore, forecasting the performance of the stocks below has been done through five different evaluation criteria, where the mean square error (MSE), mean absolute error (MAE), mean absolute percentage error (MAPE), coefficient of determination (R^2), and finally adjusted coefficient of determination (R^2_{adj}).

The tables below will reflect the result of the models and the evaluation criteria on the following listed companies, GF Securities (sz.000776), Unicom (sh.60050), and BAOSTEEL (60019).

GF Securities (sz.000776)					
Model	**MSE**	**MAE**	**MAPE**	**R^2**	**R^2_{adj}**
PCC-BLS	0.166	0.285	1.864	0.950	0.947
LSTM	0.350	0.389	2.506	0.895	0.887
MLP	1.080	0.673	4.201	0.677	0.652

Unicom (sh.600050)					
Model	**MSE**	**MAE**	**MAPE**	**R^2**	**R^2_{adj}**
PCC-BLS	0.006	0.054	1.092	0.982	0.981
LSTM	0.012	0.080	1.627	0.962	0.959
MLP	0.122	0.287	5.709	0.659	0.633

BAOSTEEL (sh.600019)					
Model	**MSE**	**MAE**	**MAPE**	**R^2**	**R^2_{adj}**
PCC-BLS	0.028	0.109	1.724	0.987	0.986
LSTM	0.071	0.165	2.493	0.969	0.967
MLP	0.397	0.406	6.264	0.820	0.806

From the tables above, it is evident that the best performing machine learning method was the PCC-BLS, while the worst performing model was the MLP. The LTSM model could be considered the middle ground between the best and worst model.

Final Model Performance

It is evident that the LSTM was not the best predictor of future stock prices, as the PCC-BLS was the most accurate. The Pearson Correlation Coefficient (PCC) was utilized to select the 35 features for input, where the original stock price, technical indicators, and financial indicators were chosen by the PCC. Moreover, the features selected were utilized for rapid information feature extraction and training a Broad Learning System (BCC). The PCC-BLS model was able to provide a more accurate forecasting of future values.

RNN Structure

The RNN structure of the LSTM model used within the academic research paper was not mentioned or explained. However, the LSTM model was utilized to predict the S&P 500 trend from the up-down signals. Furthermore, the LSTM was combined with the Naïve Bayes classifier to predict the opening price of the listed companies within China.

Academic Article 2: Jointly modeling transfer learning of industrial chain information and deep learning for stock prediction
Article authors: Dingming Wu, Xiaolong Wang, Shaocong Wu
Year published: 2022-04-01

Data and Timeframe Used

The research paper by the authors mentioned above revolved around 379 stock market indices by industry in China, which were accessed on the 7th of December 2020. Where the Diebold Mariano test (DM test) was utilized on the prediction results of the validation data. Furthermore, the local interpretable model agnostic explanations (LIME) were used as the 'explainer' in the model results.

Furthermore, the authors employed transfer deep learning of industrial chain information and deep learning models for stock market prediction. Where the four deep neural networks, Multi-Layer Perceptron (MLP), Recurrent Neural Network (RNN), Long Short Term Memory (LSTM), and Gated Recurrent Unit (GRU) were trained through studying the characteristics of upstream enterprises stock prices. Following the training, a second phase of training begins, where the networks was transferred into the corresponding downstream company stock data.

When the networks are fully trained, they are capable of partially memorizing stock price characteristics of upstream enterprises. Moreover, the networks are able to highly realize accurate stock market predictions of downstream enterprises. The analysis was undertaken based on the results of transfer learning for 100 pairs of datasets.

Upstream industries such as planting, mining, fishing, processing, breeding of raw materials, mining of iron ore, oil, coal mine, natural gas, forest, grain, cotton. While, middle stream industries can be considered as, reprocessing of raw materials such as steel, chemical industry, pharmaceutical biology, textile, nonferrous metals, papermaking, oil refining, and food processing. Downstream industries can be considered as, transportation, automobile, electrical equipment, logistics, shopping malls and aerospace.

Metric of Success

The metric of success was measured through representing the mean value of the final ten epoch results for each corresponding model.

Index	MLP		RNN		LSTM		GRU	
Code	Loss	Acc	Loss	Acc	Loss	Acc	Loss	Acc
801001	0.63	0.66	0.68	0.62	0.74	0.55	0.77	0.50
801002	0.63	0.65	0.67	0.62	0.69	0.61	0.68	0.61
801003	0.57	0.70	0.62	0.63	0.65	0.61	0.63	0.62

The results from the table above reveal that the MLP has outperformed the RNN, LSTM, and GRU models. It is essential to identify that the conclusions are not part of a single set of parameters, as the parameters on the RNN, LSTM, and GRU were run multiple times. Where the hidden layers were experimented with as they ranged from 11 to 300, while the LSTM and GRU models had layers ranging from 1 to 2 with a learning rate of 0.0005 to 1. Furthermore, the RNN, LSTM, and

GRU models were retrained according to these parameters, however, the results were not satisfactory and were worse than the results of the MLP.

Final Model Performance

Through the investigation into deep machine learning and the 379 stock market indices by industry in China, it was evident that the MLP model is the most appropriate for data prediction than the RNN, LSTM, and GRU models.

The MLP model was outperformed the RNN models as it was apparent that the time series data such as stocks, the long term memory ability of the RNN models can sometimes reduce the prediction accuracy.

Furthermore, there were 100 constructed datasets for transfer learning process, where the results were analyzed and combined with the balanced proportions of the samples. It was evident through the analysis that the proposed transfer learning method based on the industrial chain information is able to improve the predication performance of the MLP model.

RNN Structure

The RNN models within this academic paper consisted of RNN, LSTM and GRU models all of which followed the following aspects:

- ⇒ Input size = 11
- ⇒ Hidden size = 11
- ⇒ Output size = 2
- ⇒ Number of layers = 1
- ⇒ Optimization function = ‘adam’
- ⇒ Parameter learning rate = 0.002
- ⇒ Betas = (0.09, 0.999)

It is relevant to identify the model parameters of the MLP model as it outperformed the RNN models. The MLP model was constructed as such:

- ⇒ Input size = 11
- ⇒ Hidden size = 100
- ⇒ Output size = 2
- ⇒ Number of hidden layers = 4
- ⇒ Dropout = 0.3
- ⇒ Activation function ‘Relu’
- ⇒ Optimization function = ‘adam’
- ⇒ Parameter learning rate = 0.001
- ⇒ Betas = (0.9, 0.999)

Academic Article 3: Copper price forecasted by hybrid neural network with Bayesian Optimization and wavelet transform
Article authors: Kailei Liua, Jinhua Chenga, Jiahui Yia,
Year published: 2022-04-01

Data and Timeframe Used

The research paper by the authors mentioned above revolved around the international copper future price during the period from January 1st, 2005, to December 31st, 2019. The dataset consisted of the open, close, high, and low prices. Furthermore, the average open and close prices of each trade data are calculated and normalized through the following equation.

$$Normalized\ Price = \frac{Real\ price - Minimum\ Price}{Maximum\ Price - Minimum\ Price}$$

Furthermore, the authors have included the wavelet transform to denoise the data, where then the denoised data was more stable and appropriate to be learned through deep machine learning. Moreover, the best hyperparameters were chosen with the Bayesian Optimization method, as it decreases the time for searching hyperparameters than the traditional methods.

Metric of Success

The metric of success was measured through the mean standard error (MSE) and the root mean squared error (RMSE) of the training time consumption and the prediction accuracy for each of the LSTM and GRU models.

The loss value and time consumption of LSTM and GRU models

Value	LSTM			GRU		
	MSE	RSME	Time	MSE	RMSE	Time
Training	0.00207	0.0455	365.83	0.00141	0.0375	317.102
Prediction	0.00103	0.0321	/	0.00080	0.0282	/

The table above indicates that the LSTM loss within the training and prediction was less than the GRU. This stipulates that the prediction results from the LSTM model should be more accurate. However, due to the LSTM's convoluted structure, the LSTM model did consume more time than the GRU model.

The training and prediction loss value of the GRU model over three different time periods

Period (days)	Training MSE	Prediction MSE	Training RMSE	Prediction RMSE
100	0.00141	0.00080	0.0375	0.0282
150	0.00152	0.00090	0.0389	0.0300

200	0.00155	0.00093	0.0393	0.0305

The table above identifies that the model performed well within the three time periods, with a small loss value, and reliable prediction accuracy. However, it is evident that the MSE from 100 to 200 days has increased. This increase is due to the fact that more copper prices are going into the prediction dataset instead of the training set. This results with the machine learning less information from the training set. Hence, more data is required to predict.

Final Model Performance

Both the LSTM and GRU models were sufficient predictors of the copper price, as the models were combined with wavelet transform and Bayesian Optimization. It is essential to have a large enough training set if the goal of the model is to predict long term predictions.

In regards of time series, it is evident that the LSTM model was able to run more accurately than the GRU model. However, the LSTM model did require more GPU power and consumed more time to run than the GRU model. Hence if the goal of the model is to create an accurate prediction at the fastest time possible than the GRU model is the model to choose. However, if computing power and time are not an issue, the LSTM models are to choose as they are lightly more accurate than the GRU models.

RNN Structure

The LSTM and GRU models both had the same structure where the structure can be seen below:

- ⇒ Number of layers = 2
- ⇒ Initial neuron numbers in first layer = 17
- ⇒ Initial neuron numbers in the second layer = 37
- ⇒ Dense layers = 2
- ⇒ Neurons number in first dense layer = 32
- ⇒ Neurons number in the second dense layer = 1
- ⇒ Dropout rate = 0.3
- ⇒ Epochs = 100
- ⇒ Batch size = 20
- ⇒ Learning rate = 0.001
- ⇒ Initial decay value of Adam optimizer = 0.08

Academic Article 4: An LSTM and GRU based trading strategy adapted to the Moroccan market
Article authors: Yassine Touzani and Khadija Douzi
Year published: 2021-09-24

Data and Timeframe Used

The research paper by the authors mentioned above revolved around three different stock market datasets. The authors have utilized the US and French stock market data as the training datasets for the LSTM and GRU models, while the Moroccan stock market data has been utilized as the validation data.

The authors have utilized the American and French stock market data to escape the discontinuity issues within the Moroccan stock market. The stocks chosen from the American stock market were all the stock from SP500, except BRK.B and BF.B as these stock prices were not loading. Furthermore, the stocks from the French stock market have been extracted from the CAC40 index utilizing investing.com investpy library. The historical prices from these two markets were collected from January 2010 to January 2019. While the Moroccan stock market prices have been chosen by selecting only the stocks with a high average volume traded in 2019 and 2020, hence from the 76 stocks available in Morocco, only 32 have been chose. The Moroccan stock prices have been extracted with the same methodology used to extract the CAC40 index data.

The main objective of the research paper is to forecast short and medium term price pattern with a high accuracy to create a trading strategy, where the moving average will be the main concern. Moreover, the LSTM models will be utilized to predict the short term stock prices, while the GRU model will be utilized to predict the medium to long term stock prices.

This is due to the fact that the LSTM model are difficult to train to handle long term dependency, due to the gradient vanishing problem. Meanwhile, the GRU model will be trained to handle the medium to long term stock price predictions due to the fact the vanishing gradient problem has been solved in 2014 by Cho et al.

Metric of Success

The quality evaluation for the prediction accuracy provided by the LSTM and GRU model will be assessed through the mean absolute percentage error (MAPE), mean squared error (MSE) and root mean squared error (RMSE).

The table below identifies the performance of each model regarding the Colorado (COL) AND Med Paper (MDP) stocks as their theta parameters are 1.01 and 1.03 respectively.

Model	Stock	MAPE (%)	MSE	RMSE	Min Price	Max Price
LSTM	COL	1.96	2.25	1.5	44.26	66.9
LSTM	MDP	3.14	0.57	0.75	12	28.92
GRU	COL	2.12	2.81	1.67	44.26	66.9
GRU	MDP	4.30	0.84	0.91	12	28.92

The table above indicates that the LSTM and GRU models predicted the same prices for the minimum and maximum stock price. While their MAPE, MSE, and RMSE are different, the LSTM model had an MAPE of 1.9% and 3.1% respectively, while the GRU model had an MAPE of 2.1% and 4.3% respectively.

Due to the fact that these results do not reflect the closing price of a stock, but act as an indication of the moving average for the next five days, the results are excellent. The results showcase a positive prediction as when the models are to predict the closing price, both models should have a small difference.

Final Model Performance

This research paper proposes a new trading strategy for the Moroccan market, where an LSTM and GRU models have been trained on the US and French stock market. It is evident that both the LSTM and GRU models provide an accurate prediction for the closing price, either short or long term.

The proposed trading strategy has indicated positive results as the strategy should return an annualized return of 27.13% and a monthly return of 2.02% within the Moroccan stock market.

RNN Structure

The LSTM model has been built to predict the short term closing future prices of the stock market, while the GRU model was built to predict the long term closing future prices of the stock market.

Both models were built through utilizing the Keras API. Where the model's number of layers was determined after a random search, while the weights were initialized with the 'He' initializers and optimized using the adaptive moment estimation 'adam' algorithm. Furthermore, the dropout was utilized as the regularization technique and Gaussian noise was added to increase the generalization within.

After the prediction models have been completed, they are then run over a simulation from the period of 2010 to 2015, utilizing a values of theta parameter for all the stocks of the dataset.

Academic Article 5: A stock price prediction method based on meta-learning and variational mode decomposition
Article authors: Tengteng Liu, Xiang Ma, Shuo Li, Xuemei Li, Caiming Zhang
Year published: 8 October 2021

Data and Timeframe Used

The research paper by the authors mentioned above revolved around creating a new stock price prediction model named, 'VML' is proposed. The VML model is able to slice the stock price series to obtain multiple window series and utilizes the variational mode decomposition (VMD) to decompose the window series to obtain multiple series.

The data required to create the new proposed model of VML is collected from the Chinese and American stock market. Where the SSE dataset consists of stocks from the SSE50 index of the Chinese market, while the DIJA dataset consists of stocks obtained from the Dow Jones Industrial Average of the American market. The data was collected from the 2nd of January 2018 to the 31st of December 2020, where stocks with a missing value were removed. Furthermore, the SSE dataset was obtained through Tushare, while the DIJA dataset was obtained from Yahoo Finance.

The proposed model predicts the daily closing price, where the dataset was split into three categories, training, test, and validation with a 4:1:1 split. Furthermore, the three sets were segmentate with a sliding window and decomposed with VMD respectively, while noise within the dataset was normalized with the formula below.

$$\mu'_{kt} = \frac{\mu_{kt} - min}{max - min}$$

Where the min and max denote the minimum and maximum values of the sample.

Metric of Success

The stock price prediction shall be compared to the LSTM, VML, VMD-LSTM, and MAML-LSTM models. Where the VMD-LSTM model is a that decomposes the entire stock price series, and results with a look-ahead bias.

While the MAML-LSTM is a model that utilizes pre-trained well generalized initial parameters for the model source tasks. That allows the model to initialize and adapt to the target task with few gradient steps.

Model	SSE			DIJA		
	MSE	MAE	MAPE (%)	MSE	MAE	MAPE (%)
LSTM	0.29	0.24	1.20	11.34	2.02	1.36
VML	0.18	0.18	0.91	7.05	1.55	1.05
VMD-LSTM	0.20	0.19	0.96	7.24	1.60	1.08
MAML-LSTM	0.27	0.23	1.18	10.03	1.93	1.32

The table above evaluates the prediction accuracy of the models with the mean squared error (MSE), mean absolute error (MAE), and mean absolute percentage error (MAPE). The table indicates that the VMD-LSTM model has the smaller MSE, MAE, and MAPE than the normal LSTM. This is also true for the VML when compared to the MAML-LSTM. These results indicate that the subseries can be modeled more effectively.

It is evident that the MAML-LSTM also has smaller MSE, MAE, and MAPE than the normal LSTM, this indicates that the MAML could improve prediction performance.

Final Model Performance

Through the experimental tests, it was evident that the VML model that utilizes meta learning and decomposition to predict stock prices is highly reliable. This is due to the fact that this model utilizes window series instead of whole series while also eliminating the look ahead bias. Furthermore, a meta learning framework MAML was utilized to alleviate the impact of concept drift on the stock price prediction.

The results yielded from the experiential tests showcase the effectiveness of MAML in regards of performance of the decomposition based model.

RNN Structure

The models have been built in regards of the VMD parameter setting, where the moderate bandwidth constraint is 2000, the noise tolerance is 0, tolerance of convergence criterion is 1e-7. While the number of sample pairs for each task is 6, where 5 samples constitute the support set, and a set sample pair constitutes the query set. Furthermore, the hidden size is 8, and the learning rate is 0.01/ While the MAML learning rate is 0.01 and the number of gradient steps is 3.

Introduction

Analyzing the future stock market prices can be viewed as an interesting topic. From the academic papers reviewed, there are multiple strategies and methods for utilizing Recurrent Neural Networks (RNNs) to predict the future stock price.

Recurrent Neural Networks are artificial connections between nodes that are formed from undirected or directed graphs along a temporal sequence. This sequence allows for temporal dynamic behavior. Within the area of RNNs, there are three different models that can be used to predict future stock prices, simple RNN, Long Short Term Memory (LSTM), and Gated Recurrent Unit (GRU).

The three different models serve different purposes, where the simple RNN operates through saving outputs of a particular layer and then feeding the layer back into the input to predict the output of the layer. While the LSTM operates through feedback connections, as the model is able to process entire sequences of data. Furthermore, the GRU operates through controlling the flow of information and have a simpler architecture.

The report will focus on training a simple RNN, LSTM and GRU model to predict the future closing stock price of the company AMP. The AMP stock dataset used to train and test the different RNN models will be from 2000-01-01 to 2020-06-01. Furthermore, each RNN model will be adjusted accordingly to ensure the best prediction results from each respectively.

The report was aimed to follow the structure of the second academic article, 'Jointly modeling transfer learning of industrial chain information and deep learning for stock prediction' where the RNN models would have followed the same structure. However, through investigating the different models it was evident that the structure of the RNN model mentioned by the article was not suitable for the AMP stock. Hence the structure of the RNN has been accordingly to suit the data.

Methods

The three different RNN models have been specifically built to achieve the highest possible prediction accuracy for the closing price of AMP stock.

Simple RNN

The Simple RNN model has been built utilizing the SimpleRNN with 50 units, 'tanh' activation, and the return sequences set to true. The dropout value was set at 0.2, while a single dense layer was set at 1.

Furthermore, the simple RNN model was compiled with an 'adam' activator, mean absolute error loss, and accuracy metrics.

LSTM

The LSTM model has been built utilizing the LSTM with 100 units twice and the return sequence set to true in the first layer, while the second layer was set to false. Moreover, two dense layers were utilized, where the first dense layer was 25 and the second dense layer was 1.

Furthermore, the LSTM model was compiled with an 'adam' optimizer and a mean absolute error loss.

GRU

The GRU model has been built utilizing four GRU layers, all of which consisted of 50 units, and return sequences set to true, except the final layer. The final GRU layer utilized a 'tanh' activator. Moreover, the model utilized four dropout layers in between the GRU layers to which they were set for 0.2. The model was then completed by a dense layer of 1.

Furthermore, the GRU model was compiled with an SGD optimizer, where the lr was set to 0.01, decay set to 1e-7, momentum set to 0.9, nesterov set to false and the loss was mean absolute error.

Throughout each of these models, a batch size of 13 was utilized, which were run over 10 epochs.

Findings

As the report was aimed to follow the structure of the second academic article, the models were built according to the description by the authors. However, the models yielded the following results.

Model	MSE
RNN	1.024
LSTM	0.682
GRU	0.241

Graphs 1,2, and 3 within the appendix reflect the prediction accuracy of the AMP stock price based on the second articles RNN model structures. It is evident that all three models either heavily overfitted, or underfitted the results. Furthermore, it is evident that the MSE's of the models can be better.

A mean squared error (MSE) is an indication of how much the predicted results are deviated from the actual stock price, through an absolute number. The MSE's of the models above indicate that the models were not accurate enough to predict the AMP stock price. This could be an indication that the structure of the models was not suitable for this data, as these models were able to accurately predict the datasets within article 2.

As the MSE and graph representations of the models above were not satisfactory, the structure of the models was adjusted accordingly to yield the following MSE results.

Model	MSE
RNN	0.0786
LSTM	0.0039
GRU	0.0071

Graphs 4,5 and 6 within the appendix reflect the prediction accuracy of the AMP stock price based on the structure that has been modified to suit the data. The RNN graph indicates that the RNN model has underfitted the results, while the LSTM and GRU models have accurately predicted the results. Furthermore, it is evident that the LSTM model was closer to predicting the actual stock price than the GRU model, based on the graphs.

Through comparing the MSE of the models that have been built specifically for the AMP data, it is evident that the LSTM and GRU models have had the smallest MSE. As the RNN model had an MSE of 0.0786, LSTM model had an MSE of 0.0039, and the GRU model had an MSE of 0.0071.

Through comparing these MSE results to the MSE results from the models that have been built based on the second article's structure. It is evident that the models built specifically for the AMP data

outperform the previous models. This is due to the fact that these models have a number of nodes and hidden layers specifically for this data, while also considering that the dense and dropout layers have also been adjusted for the AMP data.

Furthermore, it is evident that the LSTM and GRU models were both accurate at predicting the AMP stock price. However, if a single model was to be chosen, it would be the LSTM model. This is due to the fact that the LSTM model had the lowest MSE, and the graph indicates that the model was able to accurately predict the highs and lows of the data.

The models that have been built specifically for the AMP data have been chosen through trial and error, where each model was tested on different sizes of epochs, batch size, number of nodes, dropout, and dense. These parameters have allowed the refined RNN, LSTM and GRU models to outperform the models from the second article.

This can indicate that the structure of the models from the second article are very specifically tuned for the dataset that was present within their research. The models from the second article have indicated this by when tested on the AMP data they were unable to accurately predict the stock price.

Conclusions

Overall, it is evident that the models that were built based on the structure of the second article were not able to accurately predict the AMP stock price. This led to the construction of new models that were built specifically for the AMP stock price that were able to outperform the models that were based off of the second article.

It is evident that the models created for the AMP stock price data were able to predict the stock price more accurately than the models from the second article. This can be seen through the MSE and graph results.

Through creating new models that were designed and trained for the AMP stock price data, it is evident that the LSTM model was the most accurate at predicting the stock price from between the RNN and GRU models. Therefore, in regards of the AMP stock price data, it would be recommended to utilize the LSTM model to predict the future stock prices of this company.

In regards of utilizing this structure of RNN models on the AMP data, it would be suitable to consider that the naïve forecasting method is acceptable. This is due to the fact that the model was able to accurately predict the stock price based on previous data.

Recommendations

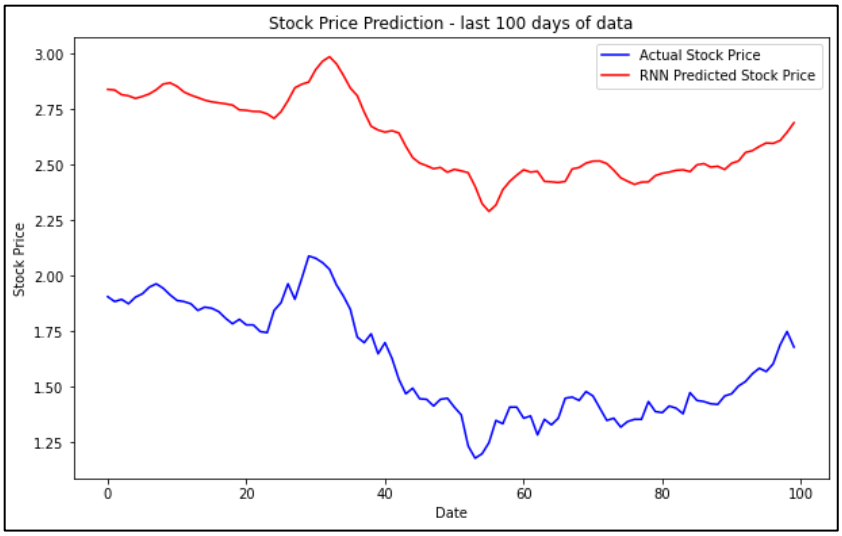
Overall, the models created for the AMP stock price data, were specifically designed, and built for this specific dataset. Moreover, the models above are only recommended for the AMP stock price prediction.

Therefore, it is recommended to create models that are able to accurately predict the stock price of any company. In other simpler terms, future models should be built to be able to predict the stock price of any company given the data.

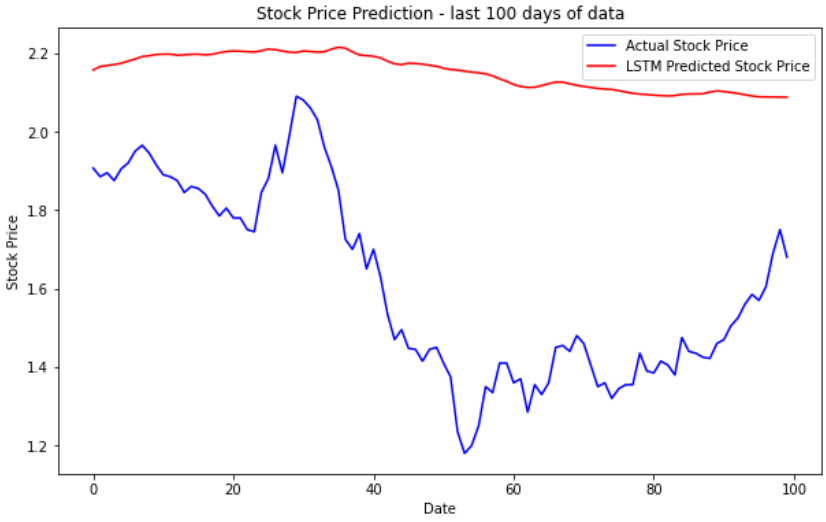
Creating models that are able to accurately predict the stock price of any given company would allow for faster and deeper analysis of the future stock price of the company. This would save time for the analysts, coders, and interpreters.

Appendix

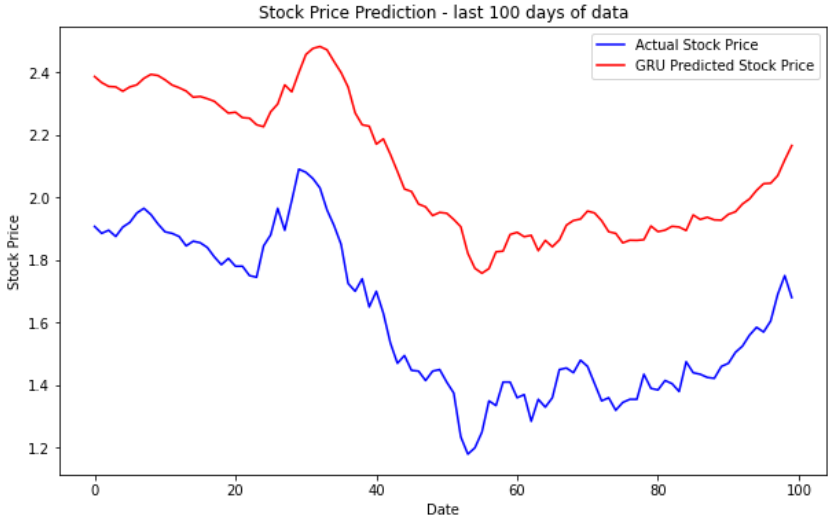
Graph 1 – RNN AMP stock price prediction. RNN structure based on second article.



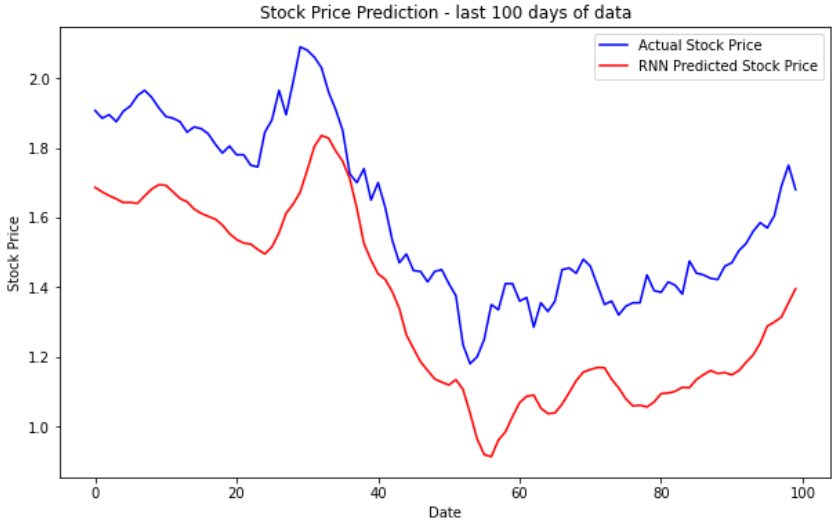
Graph 2 - LSTM AMP stock price prediction. LSTM structure based on second article**.**



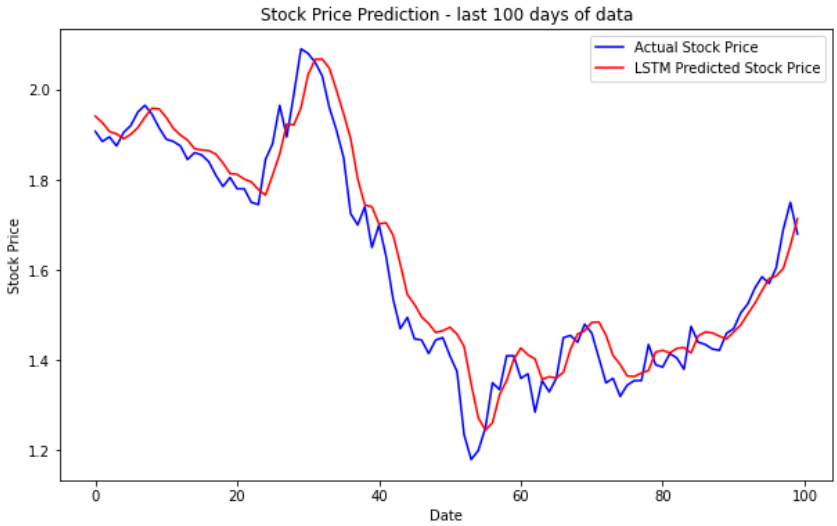
Graph 3 – GRU AMP stock price prediction. GRU structure based on second article.



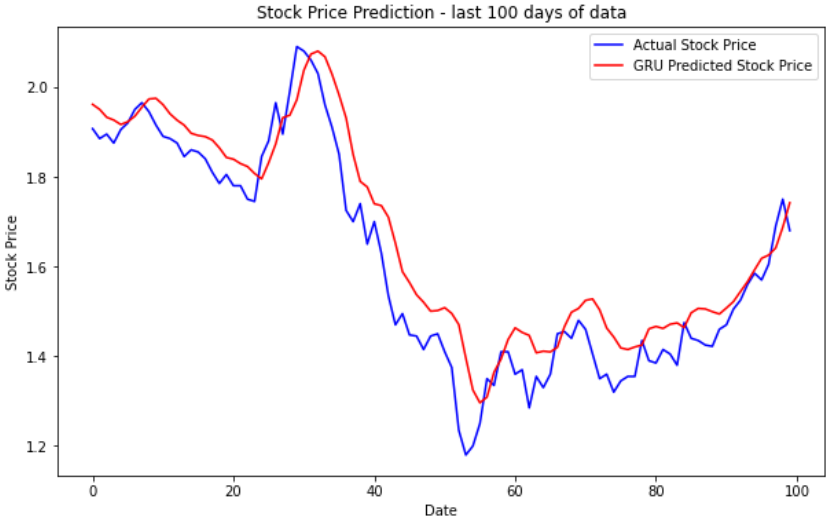
Graph 4 – RNN AMP stock price prediction. RNN structure based on data



Graph 5 – LSTM AMP stock price prediction. LSTM structure based on data



Graph 6 – GRU AMP stock price prediction. GRU structure based on data.



Code:

```
# Importing libraries
import yfinance as yf
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout, SimpleRNN, GRU
from keras.optimizers import SGD
from keras import optimizers
from sklearn.metrics import mean_squared_error

# Retrieving stock data
data = yf.download('AMP.AX','2000-01-01','2020-06-01')

# Plotting close price of AAPL
data['Close'].plot()
plt.show()

# Data pre-processing
scaler = MinMaxScaler(feature_range = (0, 1))
past_days = 60
test_split_at = len(data) - 100 - past_days # test on the last 100 days
training_processed = data.iloc[:test_split_at,[1]]
training_scaled = scaler.fit_transform(training_processed.values)

features_set = []
labels = []

for i in range(past_days, test_split_at):
    features_set.append(training_scaled[i-past_days:i, 0])
    labels.append(training_scaled[i, 0])

features_set, labels = np.array(features_set), np.array(labels)
features_set = np.reshape(features_set, (features_set.shape[0],
                                         features_set.shape[1], 1))
```

```
#############################################################################
###################################
                            # RNN Model
#############################################################################
###################################

# RNN model
modelRNN = Sequential()
modelRNN.add(SimpleRNN(units = 50, activation = 'tanh', return_sequences = True,
input_shape=(features_set.shape[1], 1)))
modelRNN.add(Dropout(0.2))
modelRNN.add(SimpleRNN(units = 50, activation = 'tanh', return_sequences = True))
modelRNN.add(Dropout(0.2))
modelRNN.add(SimpleRNN(units = 50))
modelRNN.add(Dropout(0.2))
modelRNN.add(Dense(units = 1))
modelRNN.compile(optimizer='adam', loss = 'mean_squared_error')

history = modelRNN.fit(features_set, labels, epochs = 10, batch_size = 13)

# Testing variables
testing_complete = data.iloc[test_split_at:,]
testing_processed = testing_complete.iloc[:, 1].values

test_inputs = testing_processed
test_inputs = test_inputs.reshape(-1,1)
test_inputs = scaler.transform(test_inputs)

test_features = []

for i in range(past_days, len(test_inputs)):
   test_features.append(test_inputs[i-past_days:i, 0])

test_features = np.array(test_features)
test_features = np.reshape(test_features, (test_features.shape[0],
                                test_features.shape[1], 1))

predictions = modelRNN.predict(test_features)
predictions = scaler.inverse_transform(predictions)

# Plotting prediction and actual price
plt.figure(figsize=(10,6))
plt.plot(testing_processed[past_days:,], color='blue', label='Actual Stock Price')
```

```
plt.plot(predictions , color='red', label='RNN Predicted Stock Price')
plt.title('Stock Price Prediction - last 100 days of data')
plt.xlabel('Date')
plt.ylabel('Stock Price')
plt.legend()
plt.show()

# MSE
RNN_pred = mean_squared_error(testing_processed[past_days:,],predictions)

##########################################################################
####################
                        # LSTM Model
##########################################################################
####################

# LSTM model
modelLSTM = Sequential()
modelLSTM.add(LSTM(units=100, return_sequences=True,
input_shape=(features_set.shape[1], 1)))
modelLSTM.add(LSTM(units=100, return_sequences=False))
modelLSTM.add(Dense(25))
modelLSTM.add(Dense(1))
modelLSTM.compile(optimizer='adam', loss = 'mean_squared_error')

modelLSTM.fit(features_set, labels, epochs = 10, batch_size = 13)

# Testing variables
testing_complete = data.iloc[test_split_at:,]
testing_processed = testing_complete.iloc[:, 1].values

test_inputs = testing_processed
test_inputs = test_inputs.reshape(-1,1)
test_inputs = scaler.transform(test_inputs)

test_features = []

for i in range(past_days, len(test_inputs)):
   test_features.append(test_inputs[i-past_days:i, 0])

test_features = np.array(test_features)
test_features = np.reshape(test_features, (test_features.shape[0],
                                test_features.shape[1], 1))
```

```
predictions = modelLSTM.predict(test_features)
predictions = scaler.inverse_transform(predictions)

# Plotting prediction and actual price
plt.figure(figsize=(10,6))
plt.plot(testing_processed[past_days:,], color='blue', label='Actual Stock Price')
plt.plot(predictions , color='red', label='LSTM Predicted Stock Price')
plt.title('Stock Price Prediction - last 100 days of data')
plt.xlabel('Date')
plt.ylabel('Stock Price')
plt.legend()
plt.show()

# MSE
LSTM_pred = mean_squared_error(testing_processed[past_days:,],predictions)

#########################################################################
####################################
                        # GRU model
#########################################################################
####################################

# GRU model
modelGRU = Sequential()
modelGRU.add(GRU(units=50, return_sequences=True,
input_shape=(features_set.shape[1], 1), activation='tanh'))
modelGRU.add(Dropout(0.2))
modelGRU.add(GRU(units=50, return_sequences=True,
input_shape=(features_set.shape[1], 1), activation='tanh'))
modelGRU.add(Dropout(0.2))
modelGRU.add(GRU(units=50, return_sequences=True,
input_shape=(features_set.shape[1], 1), activation='tanh'))
modelGRU.add(Dropout(0.2))
modelGRU.add(GRU(units=50, activation='tanh'))
modelGRU.add(Dropout(0.2))
modelGRU.add(Dense(units=1))
modelGRU.compile(optimizer='adam', loss = 'mean_squared_error')

modelGRU.fit(features_set, labels, epochs=10, batch_size=13)

# Testing variables
testing_complete = data.iloc[test_split_at:,]
```

```
testing_processed = testing_complete.iloc[:, 1].values

test_inputs = testing_processed
test_inputs = test_inputs.reshape(-1,1)
test_inputs = scaler.transform(test_inputs)

test_features = []

for i in range(past_days, len(test_inputs)):
    test_features.append(test_inputs[i-past_days:i, 0])

test_features = np.array(test_features)
test_features = np.reshape(test_features, (test_features.shape[0],
                                test_features.shape[1], 1))

predictions = modelGRU.predict(test_features)
predictions = scaler.inverse_transform(predictions)

# Plotting prediction and actual price
plt.figure(figsize=(10,6))
plt.plot(testing_processed[past_days:,], color='blue', label='Actual Stock Price')
plt.plot(predictions , color='red', label='GRU Predicted Stock Price')
plt.title('Stock Price Prediction - last 100 days of data')
plt.xlabel('Date')
plt.ylabel('Stock Price')
plt.legend()
plt.show()

# MSE
GRU_pred = mean_squared_error(testing_processed[past_days:,],predictions)

#########################################################################
####################################
                        # Results
#########################################################################
####################################
print("RNN model MSE: ", round(RNN_pred,4))
print("LSTM model MSE: ", round(LSTM_pred,4))
print("GRU model MSE: ", round(GRU_pred,4))
```