Stock Market Prediction Using Neural Networks

Hansda, Sheela, hansda.s@northeastern.edu
Ambi, Kaushiki, ambi.k@northeastern.edu
Salim, Fathima, salim.f@northeastern.edu

Abstract

This study explores how recurrent neural networks (RNNs) can improve predictions in the stock market by analyzing a variety of data sources including economic indicators and market indices. The research aims to show how using RNNs to sift through complex data can help predict market trends more accurately than traditional methods. The report outlines the reasoning behind the approach, the methods used, the analysis of the data, the findings, and what these results mean for the field of financial market predictions.

1. Introduction

The ability to predict stock market trends holds immense value for investors, as it enables more informed decision-making. In recent years, the rapid development of machine learning techniques, particularly deep learning, has opened new avenues for financial forecasting. Among these, convolutional neural networks (CNNs) have shown promise in various domains for their ability to effectively handle large datasets and extract meaningful patterns from complex inputs. This study seeks to apply RNNs to stock market prediction by exploiting their capabilities in feature extraction and data representation.

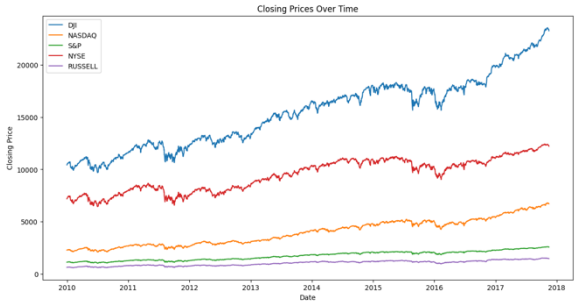
2. Background

Traditional stock market prediction techniques have largely relied on either fundamental analysis, focusing on economic indicators to assess stock value, or technical analysis, which interprets patterns from historical price movements. However, these methods often fail to capture the dynamic and nonlinear dependencies present in financial markets.

Recent research, such as the study presented in "CNN-based Stock Market Prediction Using a Diverse Set of Variables," highlights the potential of CNNs to overcome these limitations. By integrating diverse datasets and utilizing CNNs for automatic feature extraction, the study demonstrated improved accuracy over baseline models in predicting market movements (Ehsan Hoseinzadea and Saman Haratizadeha, 2018)[1].

For this project, the dataset sourced from the UCI Machine Learning Repository is employed, which includes daily market data from major U.S. stock indices such as the S&P 500 and NASDAQ from 2010 to 2017. This dataset is rich in features, including various technical indicators, economic data, and exchange rates, offering a comprehensive basis for the CNN model to learn from.

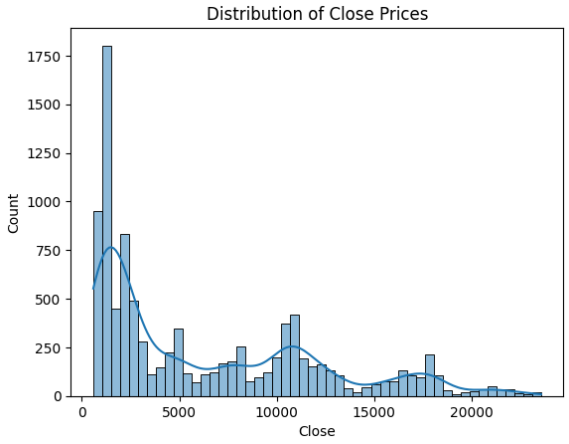
Through a detailed exploration of RNN architectures and their application to this multivariate dataset, the project aims to refine predictive accuracy and contribute to the evolving field of financial analytics by providing insights into the effective use of deep learning in stock market prediction.



3. Data Preparation and Preprocessing

3.1 Data Acquisition

The dataset utilized in this study was sourced from the UCI Machine Learning Repository, encompassing daily stock market data from major U.S. stock indices including the S&P 500, NASDAQ, NYSE, and others, spanning from 2010 to 2017. This comprehensive dataset included features such as closing prices, volume, and several technical indicators like moving averages and rate of change.



3.2 Preprocessing Techniques

To prepare the data for modeling, several preprocessing steps were undertaken:

Data Cleaning:

Handling Missing Values: Missing data points were filled using forward filling, where the last known value is used to fill the subsequent missing one, ensuring continuity in the time series data.

Removing Outliers: Extreme values in volume and price changes, which could skew the model training, were identified and removed to standardize the data distribution.

Feature Engineering:

Normalization: The features, especially price and volume, were normalized using the MinMaxScaler to scale the input variables to a range of [0, 1]. This step is crucial for models involving neural networks to speed up the learning process.

3.3 Data Augmentation and Transformation

Significant steps were taken to enhance the model's robustness and prevent overfitting through data augmentation and transformation:

Data Augmentation:

Jittering: Introduced small random changes (jitter) to the dataset to model variability and increase the model's generalization capabilities.

Scaling: Applied random scaling to the data to simulate variations in amplitude, helping the model to remain invariant to changes in scale.

Time Shifting: Implemented random shifts in the time series data to make the model robust against shifts in input data, capturing patterns that are independent of absolute positioning within the sequence.

Data Transformation:

Windowing Technique: Utilized a windowing approach to transform the series into overlapping segments of fixed size (window), suitable for sequential processing by the neural network.

3.4 Data Splitting

The final dataset was split into training and testing sets, with 80% allocated for training and 20% reserved for testing. This separation ensures an unbiased evaluation of the model, as it is tested on data that was not seen during the training phase.

4. Model Architecture and Training

4.1 Model Architecture

Initially we employed a 2D convolutional neural network (CNN) model, which did not yield the desired predictive performance. This prompted a shift towards a model architecture incorporating recurrent neural networks (RNNs), specifically utilizing long short-term memory (LSTM) units. This approach process sequential data effectively, capturing temporal dependencies within the time series data.

Model: "sequential_41"

Layer (type)	Output Shape	Param #
lstm_26 (LSTM)	(None, 8, 64)	19200
dropout_55 (Dropout)	(None, 8, 64)	0
lstm_27 (LSTM)	(None, 32)	12416
dropout_56 (Dropout)	(None, 32)	0
dense_80 (Dense)	(None, 64)	2112
dropout_57 (Dropout)	(None, 64)	0
dense_81 (Dense)	(None, 1)	65

First LSTM Layer: Comprises 64 LSTM units. This layer processes the input sequence in full, returning sequences to enable subsequent layers to continue the sequence processing. This design choice is pivotal for capturing the temporal correlations present in stock indices over time.

Dropout (0.5): Applied after the first LSTM layer to prevent overfitting by randomly omitting units from the layer during training, which helps improve the generalization of the model.

Second LSTM Layer: Contains 32 LSTM units and does not return sequences. This setup is intended to condense the information from the time series into a format suitable for classification by the output layer.

Dropout (0.5): A second dropout layer follows to further ensure robustness against overfitting, maintaining the model's performance on unseen data.

Dense Layer: A fully connected layer with 64 neurons employs the ReLU activation function to introduce non-linearity, facilitating the learning of complex patterns in the data.

Final Output Layer: Uses a sigmoid activation function to provide a binary output, predicting whether the stock index will rise or fall.

Optimization and Loss Function:
The Adam optimizer was chosen for its effective handling of non-stationary objectives and its adaptiveness to different parameters, which is ideal for datasets with noise and fluctuations typical of stock market data.

Binary cross entropy loss was used as it is well-suited for binary classification tasks. This loss function measures the performance of a model whose output is a probability value between 0 and 1.

lstm_26_input	input:	[(None, 8, 10)]
InputLayer	output:	[(None, 8, 10)]
lstm_26	input:	(None, 8, 10)
LSTM	output:	(None, 8, 64)
dropout_55	input:	(None, 8, 64)
Dropout	output:	(None, 8, 64)
lstm_27	input:	(None, 8, 64)
LSTM	output:	(None, 32)
dropout_56	input:	(None, 32)
Dropout	output:	(None, 32)
dense_80	input:	(None, 32)
Dense	output:	(None, 64)
dropout_57	input:	(None, 64)
Dropout	output:	(None, 64)
dense_81	input:	(None, 64)
Dense	output:	(None, 1)

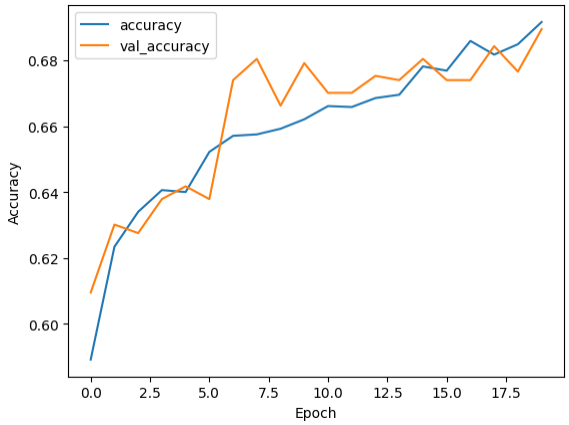
4.2 Model Evaluation:

The model was assessed based on its accuracy, and the performance metrics were tracked throughout the training process. Regular validation checks helped in tuning the model to avoid overfitting and underfitting.

5. Results

The results from the LSTM model training and validation processes are presented in this section.

Training Performance:
The training of the LSTM model with data showed a steady improvement in accuracy over epochs for both the training and validation datasets, as visualized in the accompanying accuracy graph. The training accuracy and validation accuracy converged to a narrow band, indicating that the model was generalizing well without overfitting to the training data.



The graph displays the following trends:

The training accuracy (blue line) increases steadily with each epoch, reflecting the model's growing proficiency in predicting the training data.
The validation accuracy (orange line) initially shows rapid improvement, which then stabilizes, closely tracking the training accuracy. This parallel trend suggests that the model is not memorizing the training data but learning generalized patterns that are applicable to unseen data.

Test Performance:
Upon final evaluation, the LSTM model achieved a test accuracy of approximately 67.44%. The classification report revealed the following metrics:

Precision:
Class 0 (Stock Price Down): 67%
Class 1 (Stock Price Up): 69%

Recall:
Class 0 (Stock Price Down): 70%
Class 1 (Stock Price Up): 65%

F1-Score:
Class 0 (Stock Price Down): 69%
Class 1 (Stock Price Up): 67%

	precision	recall	f1-score	support
0	0.67	0.70	0.69	970
1	0.69	0.65	0.67	971
accuracy			0.68	1941
macro avg	0.68	0.68	0.68	1941
weighted avg	0.68	0.68	0.68	1941

The balanced nature of precision and recall across the classes suggests that the model is equally adept at identifying both rising and falling trends in the stock market with a modest degree of reliability. The similar magnitude of the F1-scores indicates that there is a balance between precision and recall in the model's predictions.

6. Conclusion and Performance Comparison

This study ventured to develop a model to forecast stock market behavior using LSTM within recurrent neural networks. The performance of our model showed promising results, achieving an accuracy of roughly 67.44% on the test set, with a fair balance between precision and recall for predicting stock price direction.

When compared with the CNN-based models from the "CNNPred" study [1], our model appears to yield a stronger performance, outpacing the highest reported average F-measure value of 0.4925 from the 3D-CNNpred model. This comparison suggests that our approach, which tailors to the temporal intricacies of stock market data, may offer a more refined prediction capacity.

The LSTM model's advantage may stem from its proficiency in capturing long-term patterns in the data, a key aspect for predicting market trends. Our preparation and processing of the data likely also contributed to the improved outcome, enhancing the model's ability to apply learned patterns to new, unseen data.

While this comparison points to the potential superiority of our LSTM model, it's crucial to interpret these results with care due to variations in datasets, methods, and performance measures across different studies. Nonetheless, the findings underscore the LSTM model's value as a tool for stock market prediction.

Further explorations could investigate integrating additional data sources, refining features, and experimenting with varying model architectures to bolster the predictive strength of the LSTM approach.

7. References

[1]https://www.sciencedirect.com/science/article/abs/pii/S0957417419301915

[2]https://towardsdatascience.com/temporal-loops-intro-to-recurrent-neural-networks-for-time-series-forecasting-in-python-b0398963dc1f

[3]https://archive.ics.uci.edu/dataset/554/cnnpred+cnn+based+stock+market+prediction+using+a+diverse+set+of+variables