Stock Market Price Prediction

Valliappan Chidambaram
Utah State University
Logan, UT, USA
a02349758@aggies.usu.edu

Abstract—**Predicting stock market prices is fundamental to any investment strategy, as to make profits, one must sell a stock at a higher price than they bought it. Most traditional methods for forecasting the performance of a stock focus on long-term growth using things like a company's financial statements, though these are significantly less useful in the short-term, as short-term performance is highly noisy and volatile, dominated by things like news and investor behavior. This paper explores the possibility of using machine learning, particularly neural networks, to predict the price of stocks in the short term.**

I. Introduction

The stock market presents an opportunity to make large amounts of profits with relatively minimal efforts, but it's also highly competitive and can often lead to losing equally large amounts of money. Many large companies, such as banks and investment firms, try to accumulate money by investing in the stock market. Traditional methods usually involve looking at aspects of a company's financial statements, like their earnings per share and debt, to predict how the stock will perform over a long period of time, and then buying stocks based on their predicted long-term performance. More recently, with the advent of machine learning, much research has been done to create models to predict stock prices over shorter time periods or select portfolios that are likely to perform well. Investors and companies often use these models to augment their decision-making process when buying stocks or use them outright when performing tasks like high frequency trading. Machine learning models for stock price prediction can also be useful because they lower the amount of domain knowledge required to make profitable decisions.

The stock market is extremely noisy and volatile, with stock prices fluctuating rapidly over time from a variety of external factors, ranging from investors buying and selling a stock over the course of a day to larger events like COVID-19 [1] or news about specific companies. Most time-series forecasting methods, such as ARIMA, perform worse when the data being fit is noisy, though it is possible to work around this with methods like smoothing the data via interpolation methods [2]. Neural networks, however, are much better at handling noisy inputs and outputs [3], and recurrent neural networks such as long-short term memory (LSTM) networks can even be used solely to reduce noise [4]. This paper explores the use of neural networks, including both convolutional neural networks (CNNs) and LSTMs, to forecast stock prices.
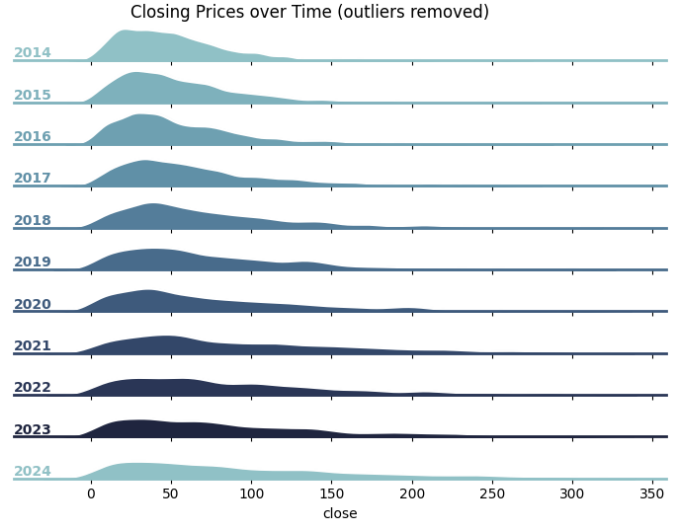

Fig. 1. Closing prices of stocks over time.

II. Method

A. Dataset

The dataset for these experiments was collected using NASDAQ's historical stock price API, which tracks the last 10 years of stock data for most stocks on their exchange. Data was collected for the largest 1000 stocks on the exchange by market cap, as large companies tend to have slightly more stable stock prices than smaller ones, and models trained on large-cap stocks were likely to generalize better to other large-cap stocks than to mid-cap or small-cap stocks. Additionally, the last 10 years of data was downloaded for the NASDAQ Composite index (COMP) and the Dow Jones Industrial Average index (DJIA), as data about overall stock market trends seemed likely to improve model performance. The data downloaded from the NASDAQ API included the opening price, closing price, high price, low price, and date for each day in the last 10 years. For stocks, but not indexes, the volume of shares traded was also included. The dataset was split into training, validation, and test sets by stock with a 60-20-20% split. The dataset was split by stock instead of time period to limit the effect of anomalous events such as COVID-19 on model generalization.

Some exploratory data analysis was conducted on the dataset to determine how useful certain features might be to machine learning models, and to gain an understanding of the
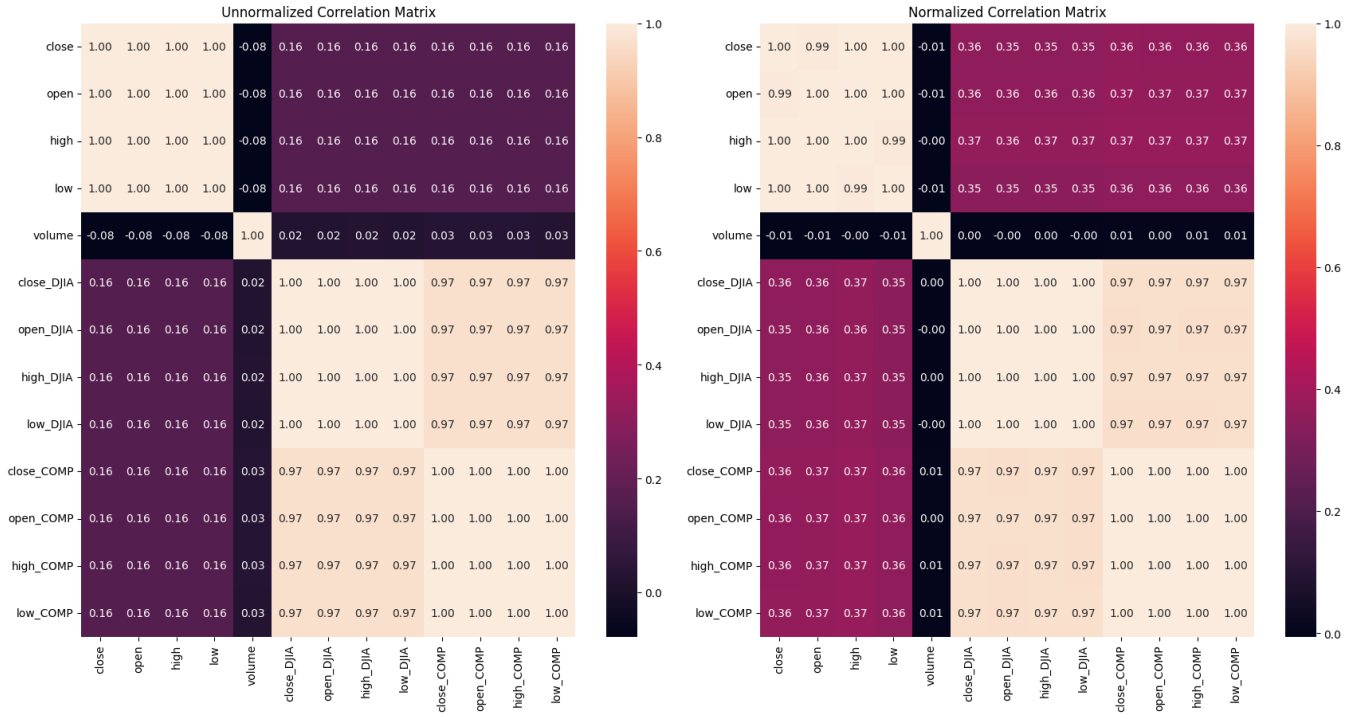
Fig. 2. Correlation of features in the dataset, unnormalized (left), and normalized by percent change from the start of the dataset (right).

difficulty of the problem. Figure 1 shows the distribution of stock's closing prices by year. It can be seen that over time, the distribution becomes flatter as stock's prices increase over time. Despite the removal of outliers from the graph, it is also clearly visible that there is a large range of possible values for stock prices, and that it changes over time, so they must be normalized for models to perform properly. Figure 2 shows an example of this. It shows two correlation matrics, consisting of all features in the dataset, including the supplemental index data. When the data is unnormalized, the correlation between the index prices and the stock prices is much lower, as individual index prices are correlated against a wide range of stock prices. After normalizing both the stock prices and index prices by their percent change from the start of the dataset, the correlation is significantly increased, showing that the historical data for the indexes is likely to be useful in predicting stock prices. Additionally, the correlation matrices show that an individual stock's or index's prices are nearly perfectly correlated, and the prices of both indices are also nearly perfectly correlated. Figure 3 shows the distribution of the change in stock price over the course of some number of days. While most stocks increase in price by an average of 0.2-0.4% over the course of 1 day, this distribution becomes flatter and wider as more time passes, showcasing the difficulty of accurately forecasting stock price changes over longer time periods.

B. Training

Models were trained and evaluated by taking 6 months of stock data (126 days) and trying to predict the following one month of closing prices (21 days). These periods were collected as moving windows over the dataset, resulting in 1200232 training examples, 406244 validation examples, and 403731 test examples. While each model was given different input features and different preprocessing for those features, the labels were always preprocessed the same way in order to be able to compare the MSE loss of different models. The labels were normalized as the percent difference in price from the last day of the input data. Two main types of models were evaluated, neural networks and linear extrapolation (which did not require any training). Models were set up as JSON files containing the features to use, the functions to preprocess them with, and for neural networks, the layers comprising the network. Several different neural network architectures were tested, primarily feedforward neural networks, CNNs, and LSTMs.

Three different preprocessing functions were tested. The first function was min-max scaling over the input time period, where the minimum value was scaled to 0 and the maximum value was scaled to 1. The second function was equivalent to the the scaling for the labels, the percent difference in price from the last day of the input window (so the last day always had a value of zero). The last function was the percent change in price from the previous day. Each expression was implemented as a lazy-evaluated expression using the Polars dataframe library, and evaluated on the input window prior to being given to the model.

Models were trained for several epochs on random samples of 5% of the training set, which were different for each epoch. Due to computational constraints arising from a lack of GPUs,
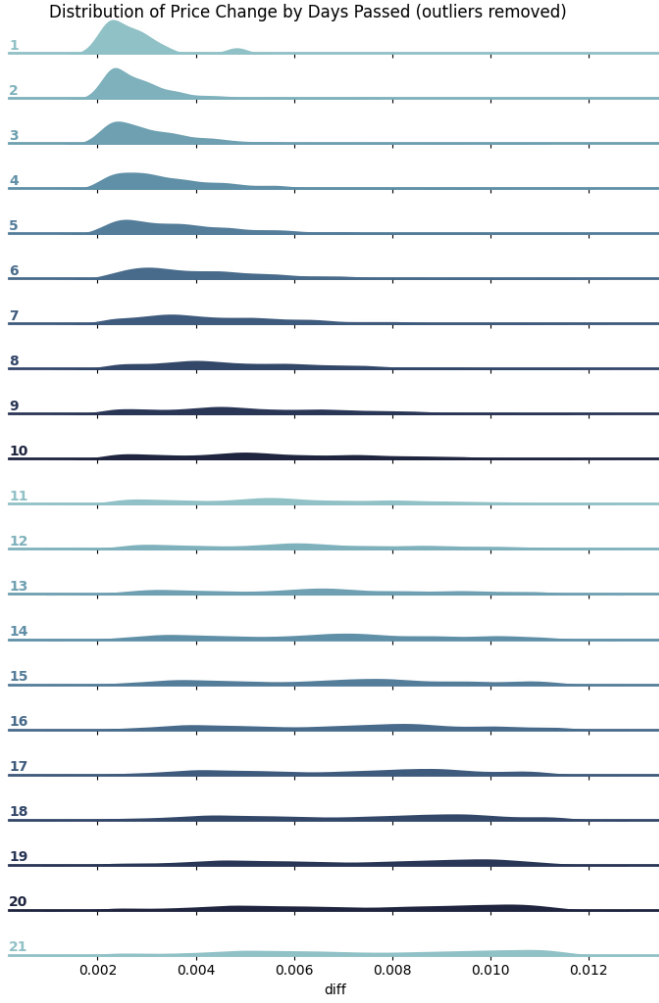
Fig. 3. Distribution of the change in stock price over the course of *N days*, *averaged by stock*.

training the models on the full training set each epoch took infeasible amounts of time. Models were trained with early stopping on the validation set, also a different random 5% sample each epoch. If a model's loss on the validation sample failed to improve after 5 epochs, training would stop and the best model would be saved. Limited hyperparameter tuning was performed by hand, mostly only the number of layers and their parameters, such as the number of neurons per layer or the size of convolution kernels. Other hyperparameters such as the learning rate of the Adam optimizer were not modified.
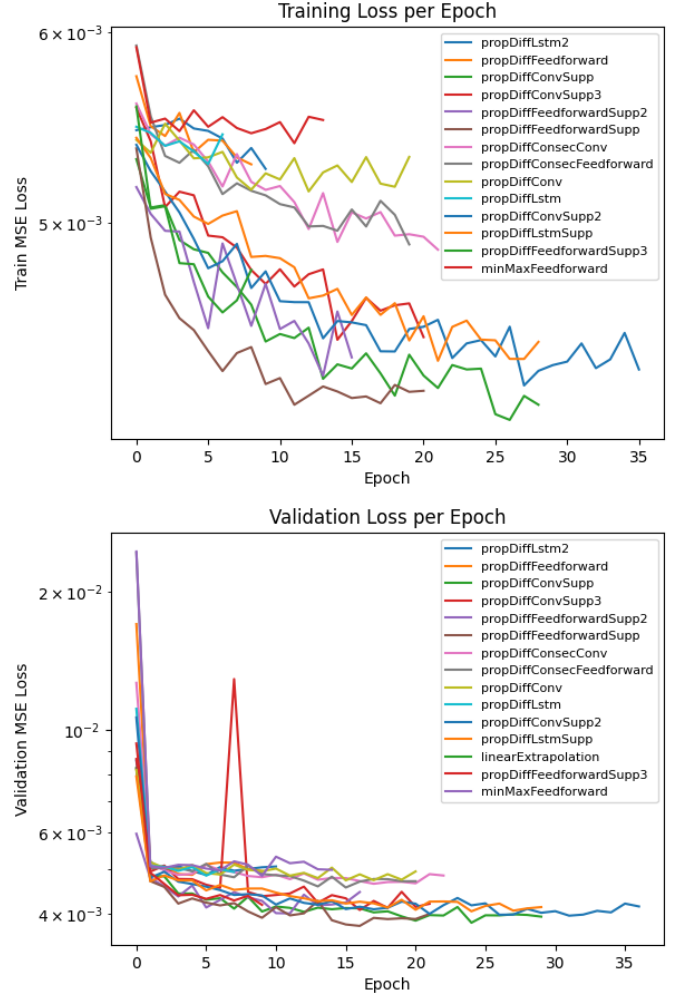
III. EXPERIMENTAL RESULTS

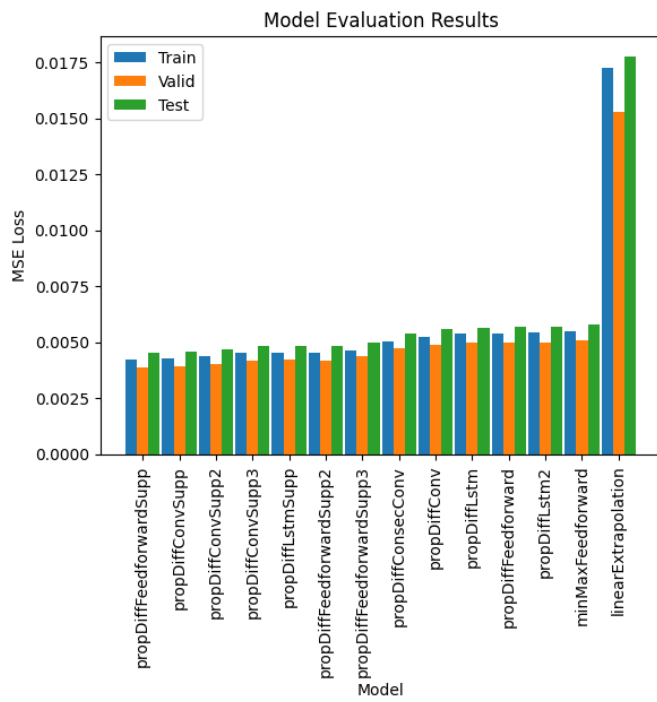Fig. 4. Training and validation MSE losses for each model.

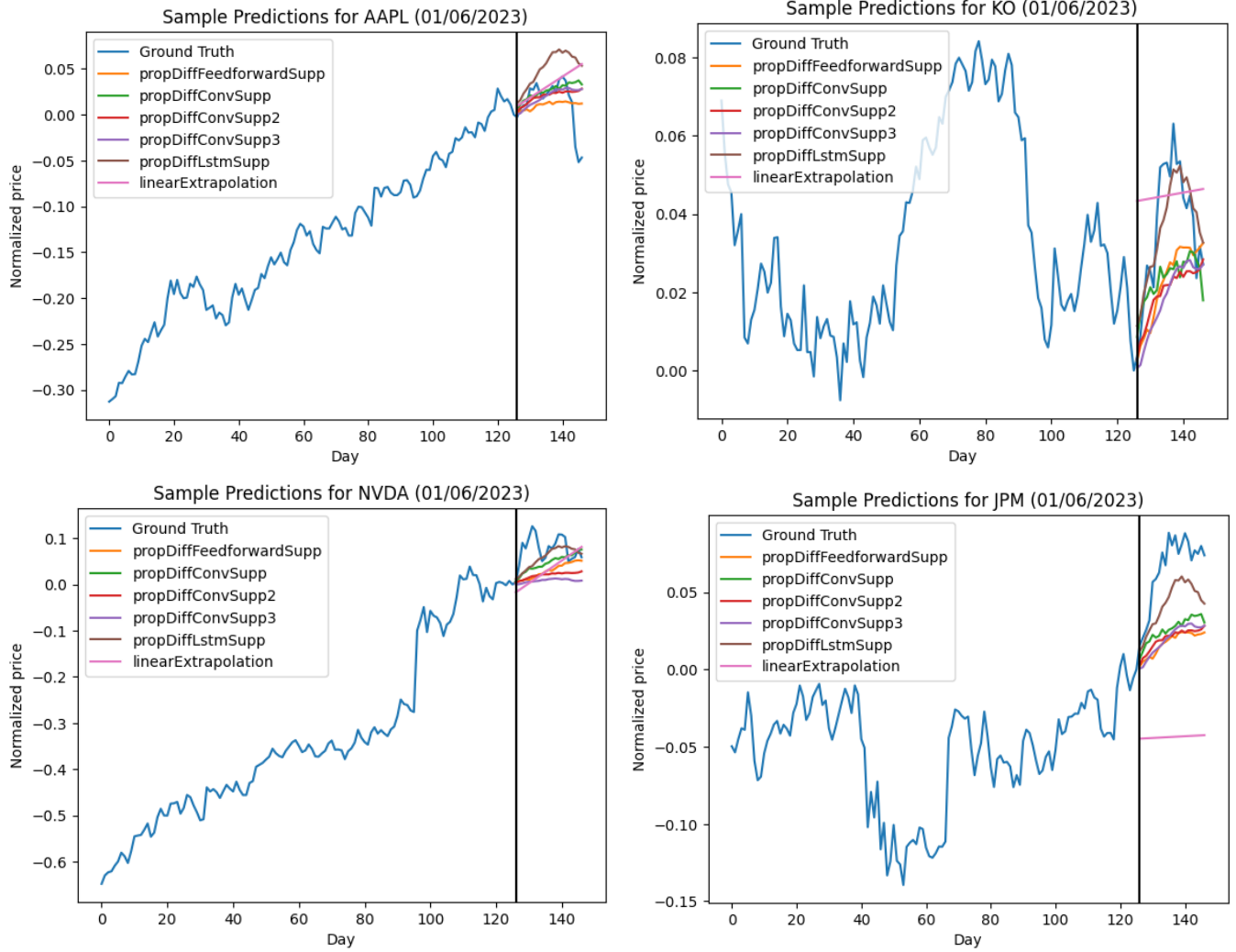Fig. 5. MSE loss on the test set for each model.

Fig. 6. Example predictions for several different stocks from the test set. The 6 month input window for each example starts on 01/06/23, and 1 month of predictions is shown for each of the top 5 models and the baseline linear regression.

REFERENCES

[1] J. Ruan, W. Wu, and J. Luo, "Stock price prediction under anomalous circumstances," in *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, Dec. 2020. [Online]. Available: http://dx.doi.org/10.1109/BigData50022.2020.9378030

[2] X. Yu, "Application research of spline interpolation and arima in the field of stock market forecasting," 2023.

[3] D. Rolnick, A. Veit, S. Belongie, and N. Shavit, "Deep learning is robust to massive label noise," 2018.

[4] Z. Wu, D. Rincon, J. Luo, and P. D. Christofides, "Handling noisy data in machine learning modeling and predictive control of nonlinear processes," in *2021 American Control Conference (ACC)*, 2021, pp. 3345–3351.