Stock Price Forecasting with ML

COE 379L - Project 4 Report

Adithya Ramanathan (ar74353), Parth Patki (pap2389)

May 5, 2025

1. Introduction and Project Statement

The goal of this project is to evaluate the effectiveness of various machine learning models in forecasting the closing prices of publicly traded stocks using historical time series data. Stock price prediction is a longstanding problem in financial data science and carries real-world significance for investors, analysts, and businesses. The ability to predict short-term price movements with even modest accuracy could support trading strategies, risk management, and broader financial planning. This project not only explores the models' performance on the stock they were trained on, but also investigates how well those models generalize to other stocks—both within the same industry and across different industries.

We selected this topic due to our collective interest in financial data analysis, as well as its suitability for applying time series forecasting techniques. Stock data is publicly available, high-volume, and inherently sequential—making it an ideal domain for experimentation with models such as LSTM, RNN, and Transformers. The project was inspired by examples discussed in class and extended through references to Kaggle notebooks and financial data science literature.

Our main research questions were:

(1) Which model performs best when predicting a stock's own future prices?
(2) How well can models generalize to other stocks in the same industry?
(3) Do any models maintain predictive value when applied across industries?

To answer these, we developed a model pipeline, performed feature engineering, and ran experiments across a diverse group of stocks from Technology, Healthcare, Construction, and Food sectors. We found that while models could accurately forecast the stock they were trained on, generalization to other stocks was significantly more challenging.

2. Data Sources and Technologies Used

2.1 Data Sources

The primary dataset was obtained using the yfinance Python library, which accesses historical data hosted by Yahoo! Finance. For each stock, we collected daily-level data fields: open, high, low, close, adjusted close, volume, dividends, and stock splits. The time range of interest was from January 1, 2000, to January 1, 2020. This 20-year window provided ample training and test data while avoiding distortions caused by the COVID-19 pandemic and its effects on the global market.

We selected across four industries: technology, healthcare, construction and food. These sectors were chosen to balance market diversity and modeling complexity. Technology and Healthcare included high-volume, higher-volatility stocks, while Construction and Food offered relatively stable trends. This deliberate sampling across industries provided a robust test bed for evaluating model transferability.

2.2 Technologies Used

The project was implemented in Python using Jupyter Notebooks. Key libraries included:

- Pandas, NumPy: Data manipulation and numerical computatio.
- Matplotlib, Seaborn: Visualization of trends, losses, and predictions
- Scikit-learn: Linear regression models and error metrics
- TensorFlow / Keras: LSTM model construction and training
- yfinance: Historical stock data acquisition

All code was modular and reproducible, with notebook checkpoints for tracking performance at different pipeline stages.

3. Methods Employed

3.1 Data Acquisition and Stock Selection

To build a robust stock price prediction system, we designed a modular and reproducible machine learning pipeline that could be uniformly applied across a range of stocks and industries. Our process began with systematic data acquisition from Yahoo Finance using the yfinance Python library, which allowed us to download historical daily

stock prices—including open, high, low, close, adjusted close, volume, dividends, and stock splits—for twelve stocks spanning four key industries:

Industry	Tickers
Technology	AAPL, NVDA, GOOG
Healthcare	ABBV, LLY, MRK
Construction	TT, CRH, URI
Food	ADM, GIS, KHC

We chose this diverse set of industries to observe how different models handle volatility and stability under varying market dynamics. To ensure consistency and avoid distortion from major global events, we selected a date range from January 1, 2000, to January 1, 2020—ending the dataset just prior to the COVID-19 pandemic's onset.

3.2 Data Cleaning and Normalization

The raw stock data was first cleaned by removing missing values. In most cases, missing data occurred in blocks—typically due to a stock not existing during the earlier years of our date range, as in the case of AbbVie (ABBV), which was only listed after 2013. Since these missing entries were not sporadic, we opted to drop them rather than impute values, thus preserving the sequential integrity of each time series. We then sorted each stock's dataset chronologically and performed MinMax scaling to normalize the range of all numerical features between 0 and 1. This normalization step is critical when training neural networks, particularly LSTMs and RNNs, whose gradients are highly sensitive to the scale of input data. For some models and tasks, we also applied Standard Scaling to ensure zero mean and unit variance, depending on which model was more sensitive to input variance.

3.3 Feature Engineering

To enhance predictive power, we performed extensive feature engineering. Beyond simply feeding in raw closing prices, we constructed lag-based features that included the stock's closing price from the previous 1 to 5 days. This approach gave the models a rolling window of recent history to work with and allowed them to capture short-term momentum or reversal patterns. We also computed 7-day and 21-day simple moving averages (SMAs) to help the models learn smoothed trends over short and medium time horizons. These moving averages are frequently used by human traders as trend

indicators, and incorporating them allowed our models to leverage domain knowledge indirectly. Volume was also included as a feature, as it often correlates with price movement and volatility. Additional features like exponential moving averages (EMAs) and relative strength index (RSI) were considered but ultimately not implemented due to time constraints.

3.4 Dataset Splitting and Supervised Framing

Once features were constructed, we split the dataset for each stock into training and testing sets using an 80/20 split. It was essential to preserve the chronological order of the data, as shuffling would introduce data leakage and invalidate our time-series forecasting. The training set included the earlier 80% of the time range, and the testing set contained the most recent 20%, simulating a real-world prediction scenario in which the model must make forecasts based on past information alone. In some experiments, especially with deep learning models, we also held out the last 100 days of data to use as a pseudo-future set for extended evaluation and plotting purposes.

Each input-output pair for supervised learning was generated using a sliding window approach. A typical input to the model consisted of 60 consecutive days of features (e.g., lagged prices and moving averages), and the output was the closing price on the 61st day. This approach was used to convert the raw time-series into a supervised regression dataset suitable for training machine learning models. While some experiments used different window lengths, we found 60 to be a good compromise between capturing enough history and avoiding excessive input dimensionality.

3.5 Modeling Approaches and Implementation

We evaluated multiple modeling strategies to compare their effectiveness in stock price forecasting. These included traditional models such as Linear Regression, Random Forest, and XGBoost, as well as more advanced neural network architectures such as Simple RNNs, Long Short-Term Memory networks (LSTM), Multilayer Perceptrons (MLP), and Transformers. Each model was implemented using a consistent pipeline, including identical preprocessing steps and evaluation metrics. Our goal was not only to identify the best model for forecasting a single stock's future prices, but also to evaluate whether any model could generalize to predict other stocks in the same industry—or even across industries—using a single trained instance.

3.6 Evaluation Metrics and Feature Selection

To measure model performance, we employed Mean Absolute Error (MAE) and Mean Squared Error (MSE) as our primary evaluation metrics. MAE gives a straightforward interpretation of the model's average prediction error in the same units as the stock

prices (typically USD), while MSE penalizes larger errors more heavily, helping to highlight models that might occasionally make catastrophic predictions. Although Root Mean Squared Error (RMSE) was considered for its intuitive units and intermediate penalization properties, we primarily reported MAE and MSE for consistency and interpretability across models.

Finally, we conducted basic feature selection using correlation heatmaps and feature importance scores derived from the Random Forest model. Features that were too highly correlated with each other were either dropped or deprioritized, as highly correlated inputs do not provide additional learning value to most models. We also experimented with training models on reduced feature sets—for instance, using only the previous day's price ('Lag_1') and the 7-day moving average ('7_MA')—to evaluate whether simpler feature sets could achieve comparable performance to the full engineered dataset.

Overall, this pipeline provided a strong foundation for comparative modeling. It allowed us to fairly evaluate the predictive power of each model across different stocks, industries, and feature sets. The combination of thoughtful preprocessing, extensive feature engineering, and diverse modeling approaches enabled us to draw meaningful conclusions about both the strengths and limitations of stock price prediction using machine learning.

4. Results

Our experiments yielded several important findings regarding model performance and generalizability.

We evaluated seven different models across the selected 12 stocks and 4 industries:

- Linear Regression
- Multilayer Perceptron (MLP)
- Simple RNN, three variants of LSTM
- Random Forest
- XGBoost
- Transformer

These evaluations were done both within-stock (i.e., training and testing on the same stock), within-industry (i.e., training on one stock and testing on another in the same sector), and cross-industry (i.e., training on one industry and testing on another). The performance of each model was measured using Mean Absolute Error (MAE) and Mean Squared Error (MSE) on the held-out test set.

4.1 Within-Stock Model Performance

When models were trained and evaluated on the same stock, performance was substantially better. Notably, the Linear Regression model trained on Apple Inc. (AAPL) achieved the best accuracy across all experiments, with an MAE of 0.429 and an MSE of 0.386. The MLP model also performed well on AAPL with an MAE of 0.462, suggesting that simple models, when carefully regularized, can achieve competitive accuracy on univariate time series prediction tasks—at least when the price dynamics are relatively stable.

In contrast, the Random Forest and XGBoost models showed poor performance even on the same stock, with MAEs above 10 and MSEs exceeding 180. These results suggest that decision-tree-based models struggle to capture temporal dependencies from windowed historical data and may be unsuitable for this type of sequence forecasting without engineered features.

The LSTM models showed mixed results. When trained and tested on the same stock, LSTM architectures produced competitive results. For instance, LSTM 3 achieved an MAE of 1.095 and an MSE of 2.29 on MRK (Healthcare), and LSTM 1 reached an MSE of 2.27 on the same stock. These results were better than the RNN but not as strong as Linear Regression for AAPL. Still, the LSTM demonstrated an ability to track long-term temporal dependencies, especially for smoother, lower-volatility stocks like MRK, GIS, and LLY.

4.2 Intra-Industry Generalization

However, when generalizing to different stocks within the same industry, performance dropped sharply across all models. For example, training an MLP on AAPL and testing on NVDA yielded an MAE of 17.24 and an MSE of 481.24, while testing on GOOG led to catastrophic errors (MAE: 222.94, MSE: 71,814). Similarly, LSTM 1 trained on MRK performed poorly on ABBV and LLY (MSEs of 2224.8 and 26796.4 respectively). Even the Simple RNN, when trained on LLY and tested on MRK, produced an MSE of 180.7—substantially higher than its same-stock MSE of 4.81. These results confirm that even models with strong within-stock performance fail to generalize reliably within industry, possibly due to differences in price scale, volatility, and stock-specific market behavior.

4.3 Cross-Industry Generalization

Cross-industry predictions suffered even more. Models trained on Healthcare and tested on Construction or Food industries consistently produced the highest error scores. For example, an LSTM 1 model trained on Healthcare and tested on

Construction yielded an MSE of 323.94; even after retraining, the best result was still an MSE of 99.03. MLP models trained on AAPL and tested on Food tickers such as ADM and GIS resulted in MSEs above 87,000 and 124,000 respectively, with MAEs in the hundreds. These results illustrate the severe limitations of applying trained models across sectors where the underlying stock behavior is not just different in magnitude, but likely driven by distinct market forces.

4.4 Transformer Performance and Observations

The only model that showed marginal improvement on cross-industry tasks was the Transformer, which achieved an MSE of 13.15 and an MAE of 3.07 when trained on Construction stocks and tested on a Technology stock. While still not comparable to same-stock LSTM or MLP performance, this hints at the potential for attention-based architectures to extract more generalizable representations.

4.5 Visual and Quantitative Summary

Visualizations of predicted vs actual prices confirmed these quantitative findings. Same-stock predictions, especially for stable stocks like MRK, GIS, or LLY, showed well-aligned trends and minimal error drift. However, when tested on unseen stocks, models often failed to follow the actual price curve, particularly around local maxima and minima.

Taken together, our results demonstrate that machine learning models can be highly effective at predicting short-term price movements for a stock they are specifically trained on. However, these models do not generalize well to other stocks—even within the same industry—highlighting the importance of retraining and model specialization. This has significant implications for financial modeling workflows, as it suggests that model reuse across tickers is not feasible without significant domain-specific tuning.

5. Future Work

While this project demonstrated that individual models trained on specific stocks can reasonably predict short-term future prices, it also revealed the significant difficulty in generalizing such models to other stocks, even within the same industry. This observation highlights multiple avenues for future improvement. One primary area of extension would be the incorporation of external data sources. Our current models rely solely on historical price and volume data, but financial markets are heavily influenced by news sentiment, macroeconomic indicators, and geopolitical events. Integrating natural language processing models to extract sentiment from news headlines or financial articles could offer valuable context for improving prediction accuracy.

Another promising direction would be to explore more sophisticated deep learning architectures, such as attention-based temporal models or hybrid CNN-LSTM frameworks. These could capture both short-term fluctuations and long-term trends more effectively than standard LSTM or MLP models. The Transformer architecture showed early potential in our experiments, especially when fine-tuned, but still struggled with generalization; modifying its positional encoding strategies or using pretrained embeddings could enhance its capability. Additionally, regularization strategies like dropout and early stopping could be combined with cross-validation techniques specifically adapted for time-series data (e.g., walk-forward validation) to prevent overfitting.

From a preprocessing standpoint, further feature engineering could improve model input representations. Technical indicators such as Bollinger Bands, MACD (Moving Average Convergence Divergence), or RSI (Relative Strength Index) are widely used by traders and could be encoded as additional features to give models more market insight. It may also be worthwhile to experiment with longer prediction horizons (e.g., predicting 7 or 30 days ahead) to assess how different models perform across various time scales. Lastly, deploying these models in a real-time environment, such as a simulation-based trading bot or dashboard, could provide insights into their practical applicability and robustness in live market conditions.

6. References

1. Yahoo Finance. (n.d.). *Historical market data*. Retrieved via yfinance Python library from https://finance.yahoo.com/.

2. Kaggle. (n.d.). *Stock Market Datasets and Prediction Notebooks*. Retrieved from https://www.kaggle.com

3. Atsalakis, G. S., & Valavanis, K. P. (2009). *Surveying stock market forecasting techniques – Part II: Soft computing methods*. Expert Systems with Applications, 36(3), 5932–5941. https://doi.org/10.1016/j.eswa.2008.07.006

4. Brownlee, J. (2018). *Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python*. Machine Learning Mastery.

5. Scikit-learn Developers. (n.d.). *scikit-learn: Machine Learning in Python*. https://scikit-learn.org

6. Pandas Development Team. (n.d.). *pandas-dev/pandas: Powerful data structures for data analysis, time series, and statistics.* https://pandas.pydata.org