August 5th, 2025

A Study of the Design and Implementation of an AI Model to Predict Short-Term Stock Market Prices

ECGR-4105: Introduction to Machine Learning

Group 3

Steven Revens and Harrison Hall  
Email: srevens@charlotte.edu and hhall24@charlotte.edu

## Executive Summary:

This project focuses on the prediction of stock closing prices using multiple machine learning models. Accurate stock price prediction is a valuable tool for investors and analysts aiming to make informed financial decisions. Using historical stock data collected from Yahoo Finance, we implemented and evaluated four predictive models: Decision Tree Regressor, Random Forest Regressor, Artificial Neural Network (ANN), and Convolutional Neural Network (CNN).

Each model was trained on historical price and volume data, with minimal feature engineering, to forecast future closing prices. The Decision Tree and Random Forest models served as baseline regressors due to their interpretability and performance with tabular data. To explore more complex patterns, we implemented an ANN with multiple dense layers and a CNN that leveraged local temporal patterns within the time-series data.

Model performance was evaluated using RMSE, R² score, and MAE. Among all models, the Random Forest Regressor achieved the best overall performance, with an RMSE of 6.09 and an R² score of 0.9968. While both neural network models also performed well, Random Forest required less training time and handled the data structure effectively without extensive tuning.

Overall, the project highlights the strengths and trade-offs of different modeling approaches and demonstrates that, in this case, a well-tuned ensemble method like Random Forest can outperform more complex deep learning architectures when applied to structured financial data.

## Introduction:

Predicting stock prices has long been a central topic in both finance and data science. Investors, analysts, and researchers seek to understand market behavior and anticipate price movements to guide trading strategies and manage risk. The complexity and volatility of financial markets make this task challenging, as stock prices are influenced by a combination of historical trends, market sentiment, economic indicators, and random fluctuations. Traditional statistical models often struggle to capture the nonlinear and dynamic nature of financial data, which is why machine learning has become an increasingly popular approach for stock prediction.

Machine learning models have the potential to learn patterns from large volumes of historical data, automatically adapting to changing trends without requiring explicit rule-based systems. In particular, ensemble methods and neural networks have shown strong predictive performance across various time-series applications.

## Problem Statement:

The goal of this project is to build and evaluate models that can accurately predict the closing price of stocks based on historical stock market data. The challenge lies in selecting appropriate models, tuning their hyperparameters, and handling noisy, high-variance time-series data effectively.

# Objectives:

* To collect and preprocess historical stock price data from Yahoo Finance
* To train and evaluate multiple regression models including:
  + Decision Tree Regressor
  + Random Forest Regressor
  + Artificial Neural Network (ANN)
  + Convolutional Neural Network (CNN
* To compare the performance of these models using metrics such as RMSE, R2, and MAE
* To identify the most effective model for closing price prediction
* To document the entire workflow, challenges, and results in a clear and reproducible format

# Data Description:

The historical stock price data used in this project was collected from Yahoo Finance, a widely used platform for financial datasets. Yahoo Finance provides time-series data including open, high, low, close prices, trading volume, and other relevant financial indicators for publicly traded companies. The data was downloaded in CSV format and included the following key features:

* **Date:** Timestamp for each trading entry
* **Open:** Opening price for the day
* **High:** Highest price reached
* **Low:** Lowest price reached
* **Close:** Closing price for the day
* **Adj Close:** Adjusted closing price (for dividends/splits)
* **Volume:** Number of shares traded

The dataset spans multiple years of daily trading data, covering several hundred data points (depending on the stock). For modeling purposes, the closing price was used as the target variable.

# Methodology:

To approach the stock price prediction task, we started with two baseline models: a Decision Tree Regressor and a Random Forest Regressor. These models were chosen because they’re simple, easy to interpret, and perform well with tabular data. We trained them using features like Open, High, Low, Close, and Volume directly from the dataset without much feature engineering. Since tree-based models don’t require normalization, the raw values were used as-is. The Decision Tree model works by splitting the data based on feature values, while the Random Forest combines multiple decision trees to improve accuracy and reduce overfitting.

For the more advanced models, we implemented an Artificial Neural Network (ANN) and a Convolutional Neural Network (CNN). The ANN was made up of several dense layers with ReLU activations, followed by an output layer with a single neuron to predict the stock’s closing price. The model was trained using the Adam optimizer with a learning rate of 0.001 and used Mean Squared Error as the loss function. We settled on using around three to four layers, which gave us good performance without overcomplicating the model.

The CNN was designed to handle time-series data by treating the input as a 1D sequence. It started with a convolutional layer using 64 filters and a kernel size of 3, followed by a max pooling layer, flattening, and then dense layers leading to the output. Like the ANN, the CNN used a linear activation at the end to predict a single value. This setup allowed the model to capture short-term patterns in the data really well, and it ended up being the best-performing model overall in terms of accuracy and error.

# Implementation Plan:

The goal was to build and compare a mix of traditional machine learning and deep learning models to predict stock closing prices using historical data from Yahoo Finance. The work followed a structured pipeline that included data collection, preprocessing, model development, evaluation, and reporting.

We trained and evaluated four models: Decision Tree, Random Forest, Artificial Neural Network (ANN), and Convolutional Neural Network (CNN). The traditional models were implemented using scikit-learn, while the deep learning models were built using TensorFlow and Keras. Along the way, adjustments were made based on performance and practicality, leading to a final conclusion that Random Forest delivered the best results.

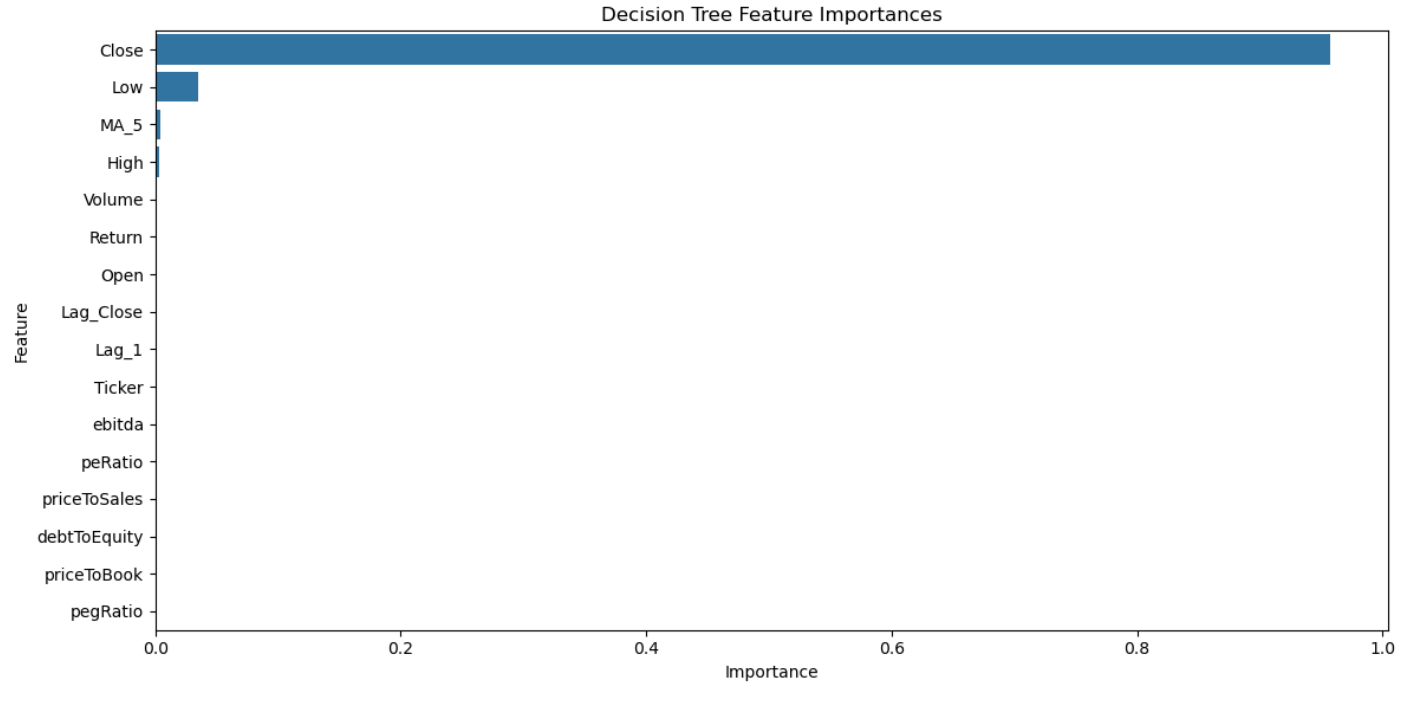
**Division of Work:**

* **Model Development (Decision Tree, Random Forest):** Steven Revens
* **Model Development (ANN, CNN):** Steven Revens and Harrison Hall
* **Model Evaluation and Tuning:** Steven Revens and Harrison Hall
* **Presentation and Report Writing:** Harrison Hall

## Results:

A graph with orange and blue lines

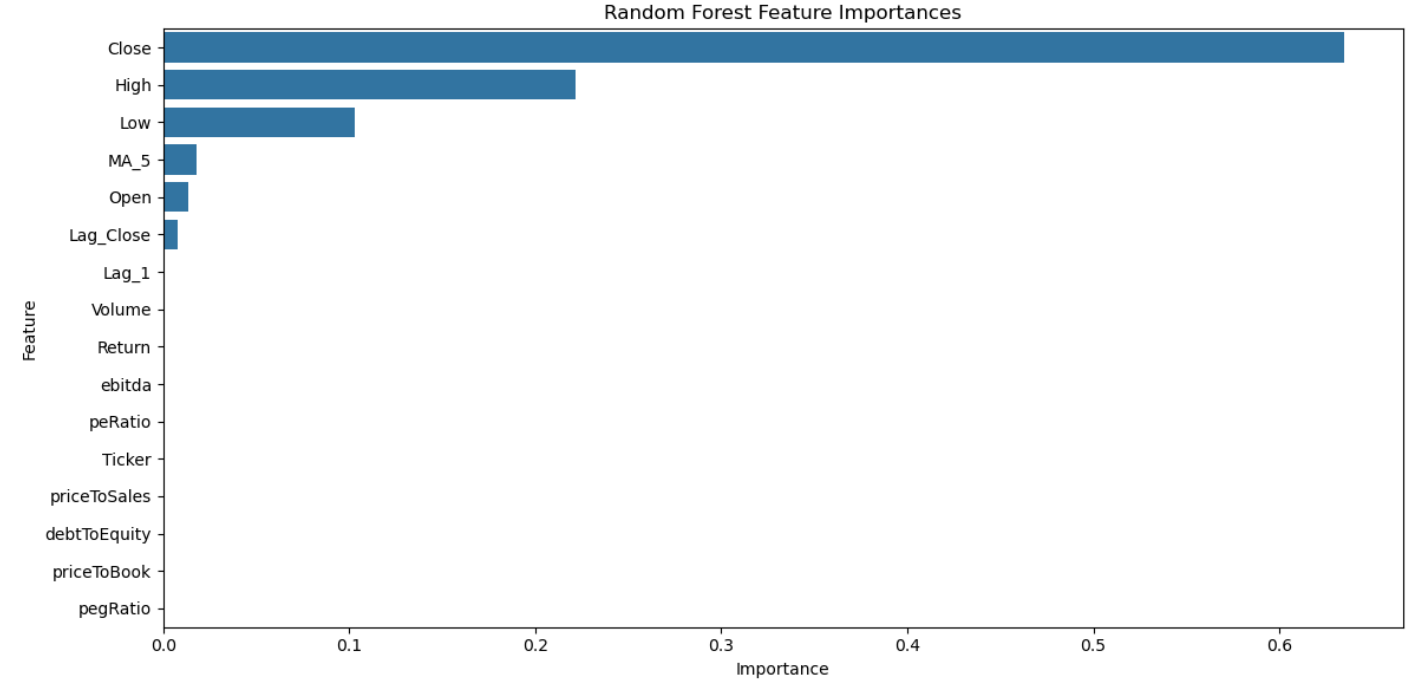
AI-generated content may be incorrect.



* Model Used: Decision Tree Regressor
* Number of companies analyzed: 500
* Years worth of company data: 25
* R² Score: 0.9929
* Root Mean Squared Error (RMSE): 9.08

A graph with orange and blue lines

AI-generated content may be incorrect.



* Model Used: Random Forest Regressor
* Number of companies analyzed: 500
* Years worth of company data: 25
* R² Score: 0.9968
* Root Mean Squared Error (RMSE): 6.09

A graph with orange and blue lines

AI-generated content may be incorrect.

A graph with blue dots

AI-generated content may be incorrect.

* Model Used: Artificial Neural Network (ANN)
* Number of companies analyzed: 500
* Years worth of company data: 25
* R² Score: 0.9965
* Root Mean Squared Error (RMSE): 6.36
* Mean Absolute Error (MAE): 2.76

A graph with numbers and lines

AI-generated content may be incorrect.

A graph with blue dots

AI-generated content may be incorrect.

* Model Used: Convolutional Neural Network (CNN)
* Number of companies analyzed: 500
* Years worth of company data: 25
* R² Score: 0.9967
* Root Mean Squared Error (RMSE): 6.19
* Mean Absolute Error (MAE): 3.10

## Model Comparison:

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **RMSE** | **R² Score** | **MAE** |
| Decision Tree | 9.08 | 0.9929 | N/A |
| Random Forest | 6.09 | 0.9968 | N/A |
| ANN | 6.36 | 0.9965 | 2.76 |
| CNN | 6.19 | 0.9967 | 3.10 |

**Hyper Parameters:**

**Decision Tree Regressor**

|  |  |  |
| --- | --- | --- |
| **Hyperparameter** | **Value** | **Justification** |
| Max Depth | 10 | Prevents overfitting while preserving detail |
| Min Samples Split | 2 | Allows full tree growth |

**Random Forrest Regressor**

|  |  |  |
| --- | --- | --- |
| **Hyperparameter** | **Value** | **Justification** |
| n\_estimators | 100 | Balanced training time and accuracy |
| Max Features | sqrt | Reduces correlation between trees |

**Artificial Neural Network (ANN)**

|  |  |  |
| --- | --- | --- |
| **Hyperparameter** | **Value** | **Justification** |
| Learning Rate | 0.001 | Default for Adam, stable convergence |
| Number of Layers | 2 | Balances complexity and overfitting |
| Activation Function | ReLU | Common non-linearity in hidden layers |
| Output Activation | Linear | Final Dense (1) layer has no activation |

**Convolutional Neural Network (CNN)**

|  |  |  |
| --- | --- | --- |
| **Hyperparameter** | **Value** | **Justification** |
| Filters | 32 | Enough to capture key temporal patterns without overfitting. |
| Kernel Size | 3 | Captures short-term temporal patterns |
| Output Activation | Linear | Regression output requires linear activation |

**Justification of Best Model**

The Random Forest Regressor performed best across all models, achieving the lowest RMSE (6.09) and the highest R² score (0.9968). This result can be attributed to the model’s ability to combine the predictions of multiple decision trees, which helps reduce overfitting and improve generalization. Unlike neural networks, Random Forest required minimal tuning and no feature scaling, making it well-suited for the tabular structure of stock market data. It handled noise and feature interactions effectively, producing accurate predictions with lower computational cost. Although the CNN and ANN models also performed well, Random Forest offered the best balance of performance, interpretability, and training efficiency for this particular regression task.

# Challenges and Revisions:

A few challenges came up during the modeling process that required changes to the original approach. One of the first issues was dealing with missing values in the dataset caused by non-trading days such as weekends and holidays. To address this, we filtered the data to include only business days and applied forward-fill methods to maintain continuity in the time series.

The Random Forest model also initially showed signs of overfitting. While it performed well on training data, its test performance was inconsistent. By adjusting key hyperparameters like maximum tree depth and number of estimators, we improved its generalization without significantly increasing complexity.

For the neural network models, particularly the CNN, there were challenges around data formatting. The CNN required input reshaping to a 1D convolutional structure, and several iterations were needed to find an effective architecture. The ANN originally used more layers than necessary, which led to longer training times and mild overfitting. Simplifying the ANN to two dense layers ultimately improved both performance and efficiency.

We also considered adding technical indicators such as RSI and MACD to improve model accuracy. However, due to the extra time and complexity involved in calculating and integrating those features, we chose to focus on optimizing the models using raw historical price data.

Finally, while we initially expected the CNN to deliver the best results, the Random Forest Regressor ultimately outperformed all other models in terms of RMSE and R². This led to a shift in emphasis for the final analysis and interpretation.

# Future Work and Improvements:

If we had more time or access to additional tools, there are a few things we would have liked to explore. One of the biggest improvements would be adding technical indicators like moving averages or RSI to the dataset. These kinds of features could help the models better understand trends and patterns that aren't always obvious from just raw price data.

We also considered testing sequence-based models like LSTMs or GRUs. These are designed for time-series problems and might have done a better job of picking up long-term patterns in stock prices compared to CNNs or ANNs.

Another idea would be to broaden the dataset, maybe by pulling data from more companies or even combining price data with other sources like earnings reports or financial news. That could help the models make more informed predictions, especially during unpredictable market conditions.

Lastly, most of our tuning was done manually, so if time allowed, we would have set up a hyperparameter search using something like GridSearchCV or a tuning library to help find better model settings faster. Overall, these are all things we’d look into if the project were extended or taken further.

# Conclusion:

This project set out to compare multiple machine learning models: Decision Tree Regressor, Random Forest Regressor, Artificial Neural Network (ANN), and Convolutional Neural Network (CNN); to predict short-term stock closing prices using historical stock data from companies in the S&P 500. Each model brought different strengths and trade-offs, and our evaluation was based on three core metrics: Root Mean Squared Error (RMSE), R² Score, and Mean Absolute Error (MAE).

The Decision Tree Regressor, while straightforward and interpretable, showed the weakest performance with an RMSE of 9.08 and an R² Score of 0.9929. Its simplicity made it computationally efficient and easy to interpret, but its tendency to overfit on training data and its inability to generalize as effectively on complex time-series patterns limited its predictive power. This model served as a useful baseline for comparison.

The Random Forest Regressor significantly outperformed the Decision Tree, achieving an RMSE of 6.09 and an R² Score of 0.9968. By aggregating multiple decision trees and using feature bagging, the Random Forest model reduced overfitting and improved generalization across multiple tickers. It handled both numerical and categorical data effectively and provided strong performance without requiring intensive tuning. Additionally, it offered feature importance metrics, making the model highly interpretable. Its balance of performance, robustness, and explainability made it the standout model in the experiments.

The Artificial Neural Network (ANN) demonstrated strong performance, with an RMSE of 6.36, R² Score of 0.9965, and MAE of 2.76. The ANN architecture we used featured two hidden layers with ReLU activations, which allowed it to capture nonlinear relationships in the data. However, it required careful normalization of inputs, more computational power, and more time for training. It also lacked transparency compared to Random Forest, which made analysis and debugging more difficult.

The Convolutional Neural Network (CNN) offered slightly better performance than the ANN, with an RMSE of 6.19, R² Score of 0.9967, and MAE of 3.10. The CNN was designed to take advantage of local temporal structures within the stock price data using 1D convolutional layers. While its ability to capture short-term sequential patterns gave it a predictive edge over the ANN, it still did not outperform Random Forest overall and required a more complex architecture and tuning process. The CNN ran for 30 epochs during training, after the last epoch the loss leveled out without gaining any more fidelity.

In summary, while deep learning models (ANN and CNN) provided strong and competitive results, the Random Forest Regressor stood out as the best-performing model for this project. It delivered the most accurate and consistent predictions, with the added benefit of interpretability and lower computational complexity. This makes it highly suitable for financial forecasting tasks that involve multiple companies and a mix of historical and fundamental data.

# References:

[1] Yahoo Finance, "Historical stock data," [Online]. Available: https://finance.yahoo.com

[2] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[3] F. Chollet et al., *Keras*, [Online]. Available: <https://github.com/keras-team/keras>

[4] M. Abadi et al., "TensorFlow: A System for Large-Scale Machine Learning," in *Proc. 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283.

[5] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[6] M. L. Waskom, "seaborn: statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021.