Forecasting Stock Prices Using Machine Learning Model

1. INTRODUCTION

This project employs machine learning techniques to predict stock prices for enhanced financial analysis. The study focuses on two contrast stocks to explore predictive modeling across market sectors; Johnson & Johnson (JNJ), representing a stable and traditional blue-chip stock from the healthcare and consumer goods industry. Then, Microsoft (MSFT), a leading big technology stock known for its growth and innovation. By analyzing these two distinct and influential companies within project's time and resource constraints, this project aims to develop and evaluate models that capture varied market dynamics.

The first machine learning model we employ in this project is Random Forest. This ensemble method operates by constructing multiple decision trees during training. For the final prediction, the model aggregates the outputs of all individual trees. Random Forest achieves diversity through two key randomization techniques. Random forest uses bootstrapping, where each tree is trained on a randomly sampled subset of the data with replacement. This diversity in training data significantly reduces the decision tree's tendency to overfit. Second, it employs feature bagging at each node within a tree, the algorithm randomly selects a subset of features to consider for the split. This process reduces correlation between the trees, further reducing the overall model variance and enhancing its ability to generalize by capturing the underlying patterns in the data. However, a limitation arises from this approach is the potential for bias and underfitting. If important predictive features are consistently overlooked during the random selection process at tree nodes, the model may fail to learn crucial relationships, leading to biased performance.

The second model utilized in this project is Gradient Boosting. While it is similar to Random Forest in its use of an ensemble of decision trees, the methodology differs fundamentally. Gradient Boosting builds models sequentially, with each new tree specifically designed to correct the errors or residuals of the combined results from the previous step. The process begins with an initial simple model to make predictions. The difference between these predictions and the actual values (the residuals) is then calculated. A subsequent tree is trained to create predictions based on these residuals; in which learning rate is introduced, a critical hyperparameter that controls how aggressively each new tree corrects the prior mistakes, therefore gradually minimizing the overall error. This iterative process continues, with new trees added to further reduce the remaining residuals thus creating better output. The model will stop until adding further trees yields no significant improvement. However, this powerful and targeted approach introduces specific challenges. Because the model learns significantly from past mistakes, it is highly prone to overfitting. This model can eventually learn not only the underlying signal but also the noise and idiosyncrasies of the training data.

The third and final model in our comparative analysis is a neural network, introduced to address the limitations in tree-based ensembles as well as to capture complex and non-linear relationships within the data. Unlike Random Forest and Gradient Boosting, which are fundamentally constrained by hierarchical and tree-structured logic, neural networks operate as a network of interconnected layers of artificial neurons. In this architecture, each connection between nodes carries a learnable weight. During a forward pass, inputs are propagated through the network. At each neuron, inputs are multiplied by their respective weights, summed together, added to a bias term, and then passed through a non-linear activation function—such as ReLU (Rectified Linear Unit) or Sigmoid. Here neural networks present a crucial step on predicting output, activation function allows the network to model non-linear patterns that simpler models might miss. By stacking multiple layers, deep neural networks can progressively learn higher-level randomness from raw input features. However, this flexibility comes with trade-offs, including a greater risk of overfitting, a high computational cost, and the need for hyperparameter tuning.

2. DATA RESULTS

For analysis purposes, we use the historical “Open”, “Close”, “High”, “Low”, and “Volume” data from the yfinance library. Those data then preprocessed to create many different features for the analysis.

a. 'Close_Yesterday': Close price yesterday
b. 'High_Yesterday': Highest price yesterday
c. 'Low_Yesterday': Lowest price yesterday
d. 'Open_Yesterday': Open price yesterday
e. 'Volume_Yesterday': Amount of volume yesterday
f. 'High_Low_Range': Range of highest and lowest price yesterday
g. 'MA5_Weekly': Moving average of 5-days close price
h. 'MA21_Monthly': Moving average of 21-days close price
i. 'Vol_W_MA5': Moving average on the 5-days amount of volume
j. 'Vol_M_MA21': Moving average on the 21-days amount of volume
k. 'C_EMA5_Weekly': Exponential Moving Average for 5-days close price
l. 'C_EMA21_Monthly': Exponential Moving Average for 21-days close price
m. 'Vol_W_EMA5': Exponential Moving Average on the 5-days amount of volume
n. 'Vol_W_EMA21': Exponential Moving Average on the 21-days amount of volume

2.1. Random Forest

	MSFT	JNJ
Hyper Parameter Tuning		
'n_estimators'	100	300
'min_samples_split'	2	2
'min_samples_leaf'	2	2
'max_depth'	20	None
Error Calculation		
MSE	17.352	1.907
MAE	3.123	0.966
MAPE	0.012	0.007
Top 5 Feature Importance		
	1. C_EMA5_Weekly: 0.441 2. Close_Yesterday: 0.323 3. MA5_Weekly: 0.116 4. Low_Yesterday: 0.054 5. C_EMA21_Monthly: 0.030	1. Close_Yesterday: 0.600 2. C_EMA5_Weekly: 0.236 3. MA5_Weekly: 0.027 4. Open_Yesterday: 0.027 5. Low_Yesterday: 0.023

2.2. Gradient Boost

	MSFT	JNJ
Hyper Parameter Tuning		
'n_estimators'	300	200
'min_samples_split'	5	2
'min_samples_leaf'	2	2
'max_depth'	7	5
'learning_rate'	0.1	0.1
Error Calculation		

MSE	14.903	1.826
MAE	2.924	0.949
MAPE	0.011	0.007

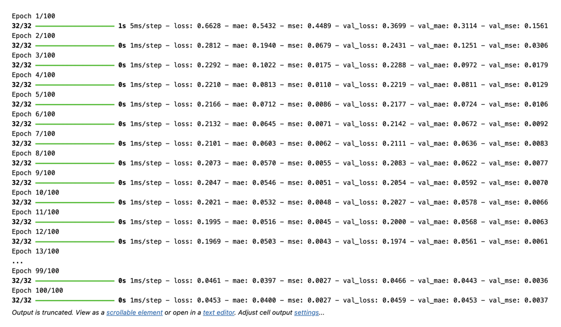
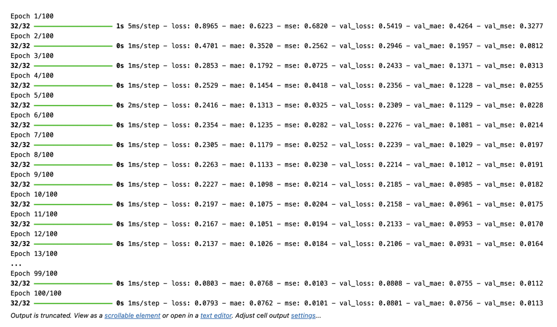
2.3. Neural Network

Data Further Preprocessing

The data undergo further preprocessing with z-score normalization as the input features have different scales and activation functions at the neural network are sensitive to input scale. Then, the shape must be adjusted so matrix multiplication can be processed at the hidden layer of the neural network.

Neural Network Architecture

1. 1st Dense Layer with 128 units, RelU activation function, and L2 (0.005) kernel regularizers
2. 2nd Dense Layer with 64 units, RelU activation function, and L2 (0.001) kernel regularizers
3. 3rd Dense Layer as output with 1 unit.
4. Adam optimizer with 0.0001 learning rate and 1.0 clip norm; calculate loss using MSE and track both MSE and MAE error
5. Train and test using 100 epochs.

MSFT	JNJ
	

```
Epoch 1/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.6628 - mae: 0.5432 - mse: 0.4489 - val_loss: 0.3699 - val_mae: 0.3114 - val_mse: 0.1561
Epoch 2/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2812 - mae: 0.1940 - mse: 0.0679 - val_loss: 0.2431 - val_mae: 0.1251 - val_mse: 0.0306
Epoch 3/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2292 - mae: 0.1022 - mse: 0.0175 - val_loss: 0.2288 - val_mae: 0.0972 - val_mse: 0.0179
Epoch 4/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2210 - mae: 0.0813 - mse: 0.0110 - val_loss: 0.2219 - val_mae: 0.0811 - val_mse: 0.0129
Epoch 5/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2166 - mae: 0.0712 - mse: 0.0086 - val_loss: 0.2177 - val_mae: 0.0724 - val_mse: 0.0106
Epoch 6/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2132 - mae: 0.0645 - mse: 0.0071 - val_loss: 0.2142 - val_mae: 0.0672 - val_mse: 0.0092
Epoch 7/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2101 - mae: 0.0603 - mse: 0.0062 - val_loss: 0.2111 - val_mae: 0.0636 - val_mse: 0.0083
Epoch 8/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2073 - mae: 0.0570 - mse: 0.0055 - val_loss: 0.2083 - val_mae: 0.0622 - val_mse: 0.0077
Epoch 9/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2047 - mae: 0.0546 - mse: 0.0051 - val_loss: 0.2054 - val_mae: 0.0592 - val_mse: 0.0070
Epoch 10/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2021 - mae: 0.0532 - mse: 0.0048 - val_loss: 0.2027 - val_mae: 0.0578 - val_mse: 0.0066
Epoch 11/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1995 - mae: 0.0516 - mse: 0.0045 - val_loss: 0.2000 - val_mae: 0.0568 - val_mse: 0.0063
Epoch 12/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1969 - mae: 0.0503 - mse: 0.0043 - val_loss: 0.1974 - val_mae: 0.0561 - val_mse: 0.0061
Epoch 13/100
...
Epoch 99/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0461 - mae: 0.0397 - mse: 0.0027 - val_loss: 0.0466 - val_mae: 0.0443 - val_mse: 0.0036
Epoch 100/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0453 - mae: 0.0400 - mse: 0.0027 - val_loss: 0.0459 - val_mae: 0.0453 - val_mse: 0.0037
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

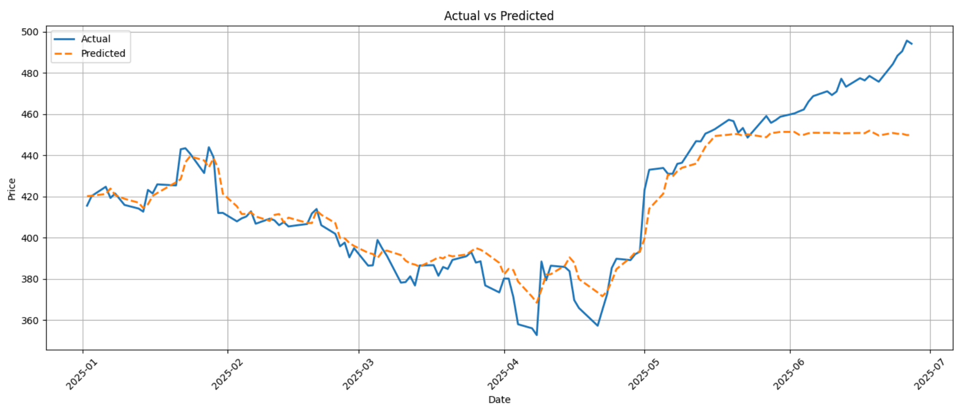
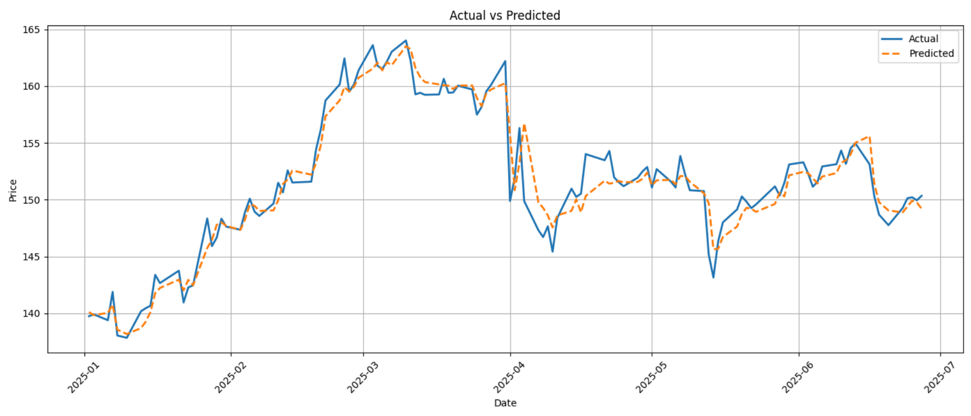
```
Epoch 1/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.8965 - mae: 0.6223 - mse: 0.6820 - val_loss: 0.5419 - val_mae: 0.4264 - val_mse: 0.3277
Epoch 2/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.4701 - mae: 0.3520 - mse: 0.2562 - val_loss: 0.2946 - val_mae: 0.1957 - val_mse: 0.0812
Epoch 3/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2853 - mae: 0.1792 - mse: 0.0725 - val_loss: 0.2433 - val_mae: 0.1371 - val_mse: 0.0313
Epoch 4/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2529 - mae: 0.1454 - mse: 0.0418 - val_loss: 0.2356 - val_mae: 0.1228 - val_mse: 0.0255
Epoch 5/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.2416 - mae: 0.1313 - mse: 0.0325 - val_loss: 0.2309 - val_mae: 0.1129 - val_mse: 0.0228
Epoch 6/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2354 - mae: 0.1235 - mse: 0.0282 - val_loss: 0.2276 - val_mae: 0.1081 - val_mse: 0.0214
Epoch 7/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2305 - mae: 0.1179 - mse: 0.0252 - val_loss: 0.2239 - val_mae: 0.1029 - val_mse: 0.0197
Epoch 8/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2263 - mae: 0.1133 - mse: 0.0230 - val_loss: 0.2214 - val_mae: 0.1012 - val_mse: 0.0191
Epoch 9/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2227 - mae: 0.1098 - mse: 0.0214 - val_loss: 0.2185 - val_mae: 0.0985 - val_mse: 0.0182
Epoch 10/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2197 - mae: 0.1075 - mse: 0.0204 - val_loss: 0.2158 - val_mae: 0.0961 - val_mse: 0.0175
Epoch 11/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2167 - mae: 0.1051 - mse: 0.0194 - val_loss: 0.2133 - val_mae: 0.0953 - val_mse: 0.0170
Epoch 12/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2137 - mae: 0.1026 - mse: 0.0184 - val_loss: 0.2106 - val_mae: 0.0931 - val_mse: 0.0164
Epoch 13/100
...
Epoch 99/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0803 - mae: 0.0768 - mse: 0.0103 - val_loss: 0.0808 - val_mae: 0.0755 - val_mse: 0.0112
Epoch 100/100
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.0793 - mae: 0.0762 - mse: 0.0101 - val_loss: 0.0801 - val_mae: 0.0756 - val_mse: 0.0113
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Error Calculation		
	MSFT	JNJ
MSE	24.484	1.650
MAE	3.676	0.912
MAPE	0.013	0.006

2.4. Ensemble Model

In this ensemble model, all of the 3 machine learning models will be ensembles to create output. The ensemble model is based on the weighted average of all 3 models. The weight is based on error resulting from the model in the training and testing phase, where lower errors means higher weight to the final ensemble model. This ensemble model then will be applied to the first-half of 2025 historical data of both stocks.

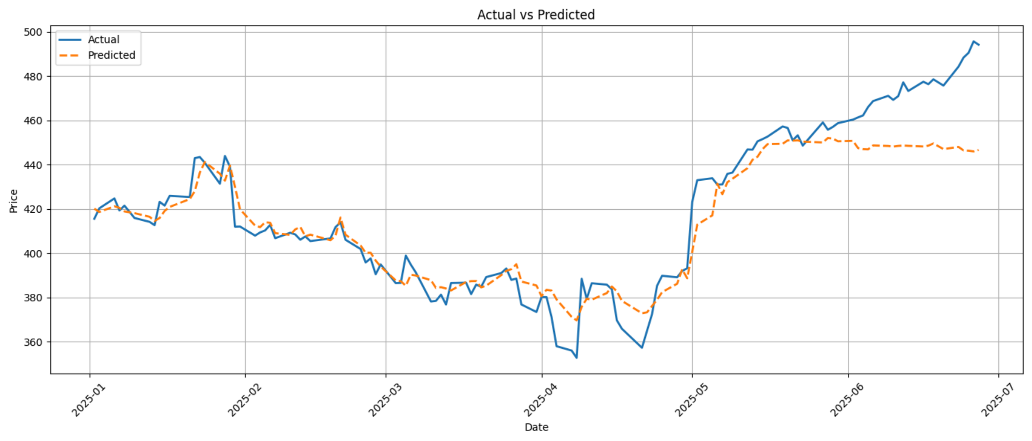
2.4.1. Random Forest

MSFT

JNJ


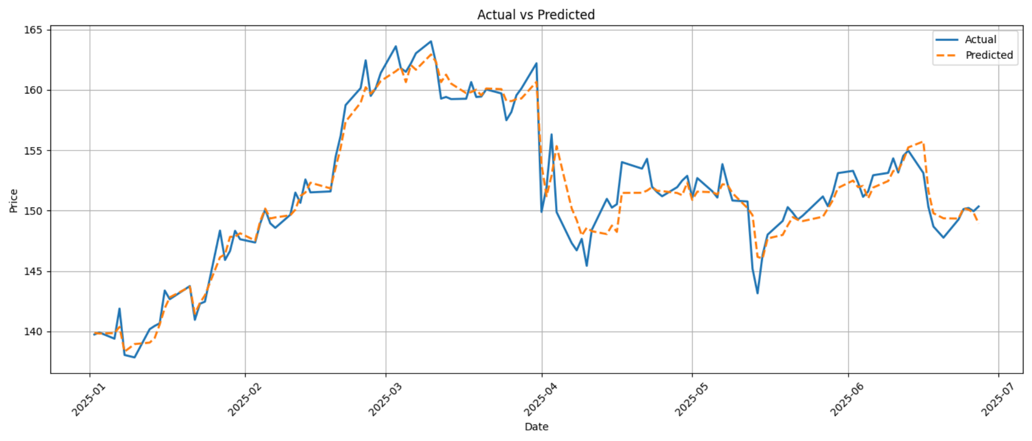
Error Calculation		
	MSFT	JNJ
MSE	172.275	2.207
MAE	9.011	1.028
MAPE	0.021	0.007

2.4.2. Gradient Boost

MSFT



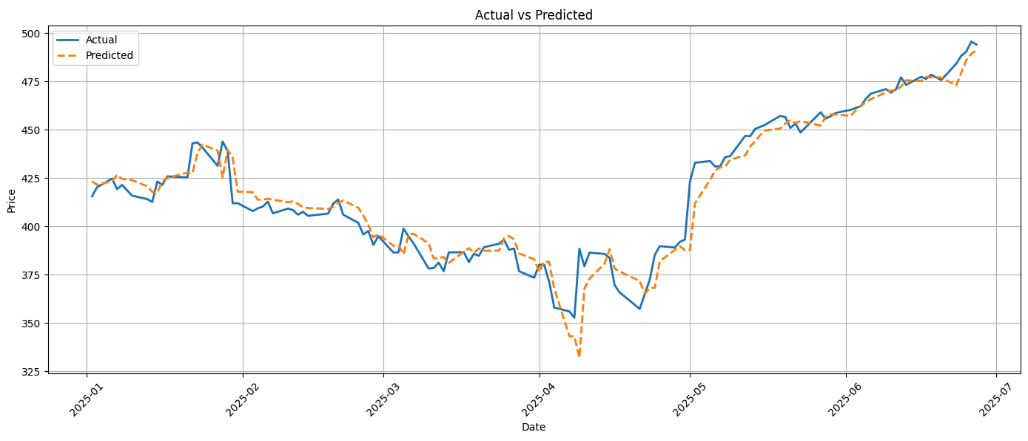
JNJ

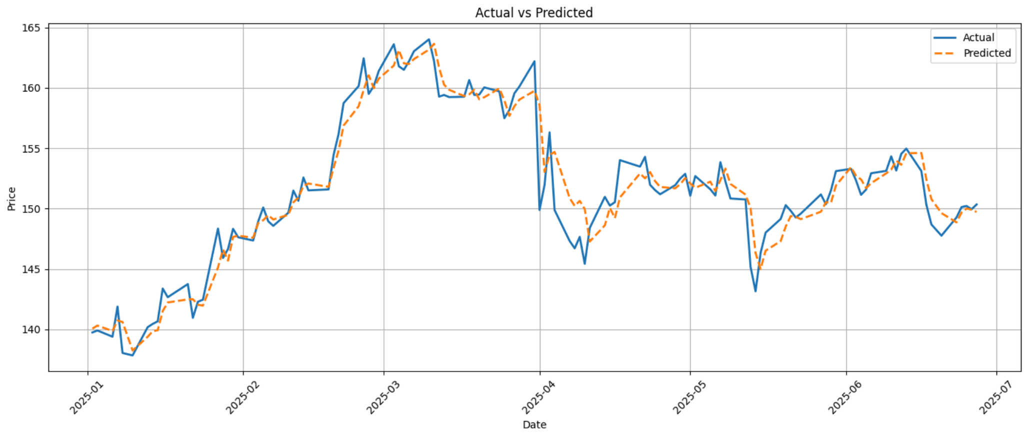


Error Calculation

	MSFT	JNJ
MSE	190.103	1.933
MAE	9.037	0.978
MAPE	0.021	0.006

2.4.3. Neural Network

MSFT


JNJ


Error Calculation		
	MSFT	JNJ
MSE	84.816	2.816
MAE	6.015	1.177
MAPE	0.015	0.008

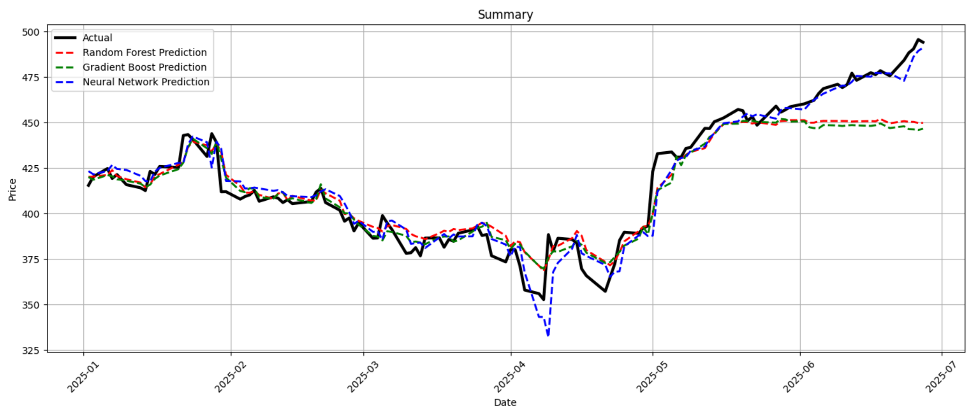
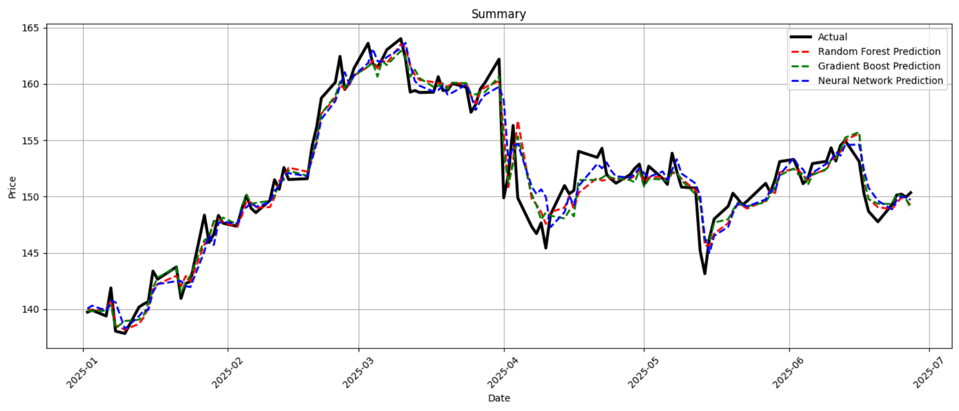
2.4.4. Final Ensemble Model

	MSFT	JNJ
Weight on Each Model		
Random Forest	0.296	0.338
Gradient Boost	0.282	0.362
Neural Network	0.422	0.300
Error Calculation		
MSE	103.763	2.113
MAE	7.355	1.013
MAPE	0.017	0.007

3. RESULT ANALYSIS

The quantitative evaluation of the three machine learning models (Random Forest, Gradient Boosting, and Neural Networks) on both JNJ and MSFT stocks reveals acceptable performance. While the calculated Mean Squared Error (MSE) values differ significantly between models and can sometimes appear large due to their sensitivity to outliers, more robust metrics like Mean Absolute Error (MAE) and, notably, Mean Absolute Percentage Error (MAPE) provide a clearer picture of practical accuracy. Across all models and for both stocks, the MAPE remains below 3%, indicating that predictions are, on average, within an acceptable and relatively narrow margin of error relative to actual prices.

However, given that error metrics alone offer an aggregated and sometimes abstract summary, the analysis places a greater emphasis on graphical interpretation. Visualizing the predicted versus actual price trajectories over time allows for a more intuitive and comprehensive assessment of model performance, revealing patterns, consistency, and the timing of errors in a way that pure numerical calculations cannot fully explain.

Model Comparison	
MSFT	
JNJ	

As illustrated in the graphical results above, a clear divergence in model performance emerges between the two stock types. All three models provide well-fitted predictions for the traditional stock, Johnson & Johnson (JNJ). They successfully capture its overall trend and moderate volatility, demonstrating the suitability of these methods for stable and blue-chip stocks with relatively predictable patterns.

However, their efficiency diminishes when applied to the big-tech stock, Microsoft (MSFT). The models struggle to accurately fit the more dynamic and volatile price movements, particularly the sudden volatility spikes observed in mid-2025. A key reason for this shortcoming is likely the limitation of using only historical price data. Big technology stocks like MSFT are often highly sensitive to external factors that are not captured in past prices alone, such as market news, earnings surprises, product launches, and shifting public sentiment.

Among the models, a crucial difference emerges in handling this tech-stock volatility. Both Random Forest and Gradient Boosting fail to capture the sharp upward spike in MSFT's price in the latter half of 2025, resulting in smoother and lagging predictions. In contrast, the Neural Network demonstrates an ability to approximate this sudden spike, giving to its greater capacity for modeling complex and non-linear relationships. This suggests that for capturing more complex volatility, a more flexible model like a neural network or an ensemble that combines the strengths on neural networks with the stability of tree-based methods could yield better performance.

Finally, the ensemble model tested that combined the strengths of all three methods. This combined approach performed well for both stock types. While it still couldn't perfectly predict Microsoft's sharp spike in mid-2025 (likely for the same reasons stated earlier) it proved to be a more balanced and reliable model overall.

By blending the predictions, the ensemble smooths out the weaknesses of any single model. This makes it less likely to overfit to the model-building specific data and better prepared to handle unexpected volatility in the future. So, while no model can fully capture sudden market shocks, the ensemble offers the best practical balance between accuracy for stable stocks and resilience for volatile ones.