Bhartiya Vidya Bhavan’s

**Sardar Patel Institute of Technology**

(Autonomous Institute Affiliated to University of Mumbai)

**Department of Computer Science & Engineering**

Stock Price Prediction

By

Devansh Pursnani

2023800091

Project

**Python Programming for Data Science**

## **TABLE OF CONTENTS**

## 

## Abstract - 3

1. Introduction - 4
2. Dataset - 5
3. Methodology - 7
4. Experimental Setup - 9
5. Results and discussion - 12
6. Error Analysis and Model Improvements - 22
7. Conclusion - 25
8. References - 27

## 

## 

## 

## 

## 

## 

## 

## 

## 

## **Abstract**

This project is about predicting stock prices using past market data, focusing mainly on the ‘Close’ price of a stock. The goal was to build a full pipeline that prepares the data, explores it using graphs, builds prediction models, and checks how well they perform.

The data included stock prices such as Open, Previous Open, Previous Close, and other values like High, Low, and Volume. Before making predictions, the data was cleaned, missing values and unusual numbers were handled, and helpful new features were created. Several visualizations were used to better understand patterns in the data.

We tested different models like Linear Regression, Decision Tree, Random Forest, and XGBoost to see which worked best. We measured how good the predictions were using standard scoring methods and also made sure the models were not overfitting or underfitting the data.

In the end, the tree-based models like Random Forest and XGBoost gave the best results. After some adjustments, they were able to predict stock prices accurately and reliably.

## **Introduction**

## **Problem** **Statement**

This project focuses on predicting the closing prices of major technology stocks—AAPL, GOOGL, MSFT, and AMZN—using historical stock market data from the NASDAQ. The goal is to build models that can accurately forecast the next day’s closing price based on past price movements and trends.

## **Objective**

The main objective of this project is to apply regression models to predict future stock closing prices using historical daily trading data. The focus is on improving model performance while avoiding issues like overfitting or underfitting.

## **Motivation**

With the growing popularity of data-driven trading and algorithmic investing, there is a strong demand for accurate prediction systems in the stock market. This project was chosen out of personal interest in financial markets and the increasing relevance of data science in developing automated trading strategies.

## **Outline**

The report begins with data cleaning and preprocessing, followed by exploratory data analysis (EDA) using various visual techniques. Feature engineering and normalization are performed to prepare the data for machine learning models. Multiple regression models—including Linear Regression, Decision Tree, Random Forest, and XGBoost—are trained and validated using performance metrics like R², MSE, and MAE. The project concludes with model comparisons and observations on overfitting/underfitting behavior.

## **Dataset**

**Description of Dataset:**

The dataset used in this project contains historical daily stock prices for companies listed on the NASDAQ stock exchange. Specifically, data for four major tech companies—Apple (AAPL), Google (GOOGL), Microsoft (MSFT), and Amazon (AMZN)—was selected for analysis. This data was sourced from Yahoo Finance using the yfinancePython package and includes entries up to April 1, 2020. Each record provides stock prices for a single trading day and includes the following features:

* Date – Trading date
* Open – Opening price
* High – Highest price of the day
* Low – Lowest price of the day
* Close – Closing price
* Adj Close – Adjusted close (for dividends and splits)
* Volume – Number of shares traded

Additional features were derived to enhance the dataset, such as Prev\_Open and Prev\_Close, which represent the previous day’s opening and closing prices. After cleaning, the final dataset consists of **27,585 rows and 13 columns**.

**Preprocessing:**

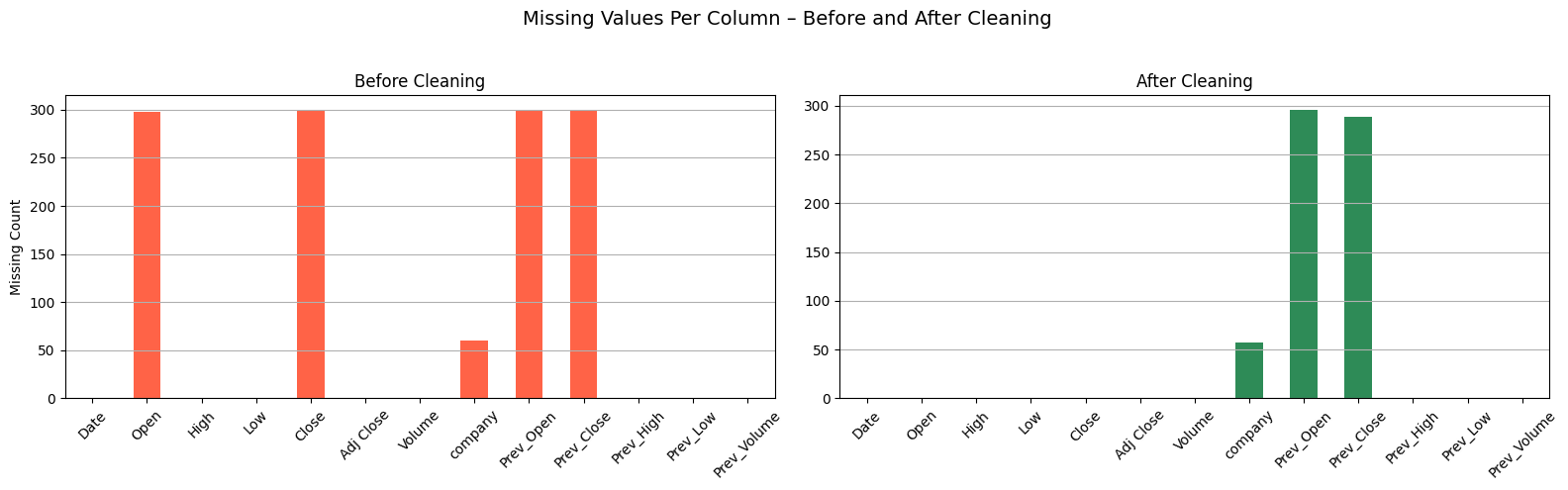
To prepare the data for modeling:

* Missing values were handled using a combination of **forward fill** (ffill) and **backward fill** (bfill) strategies to preserve time continuity.
* Standardization was applied using **StandardScaler** to normalize the scale of features, which helps many machine learning models perform better.
* Feature engineering included **shifting operations** to create Prev\_Open and Prev\_Close values from historical data.

**Data Exploration:**

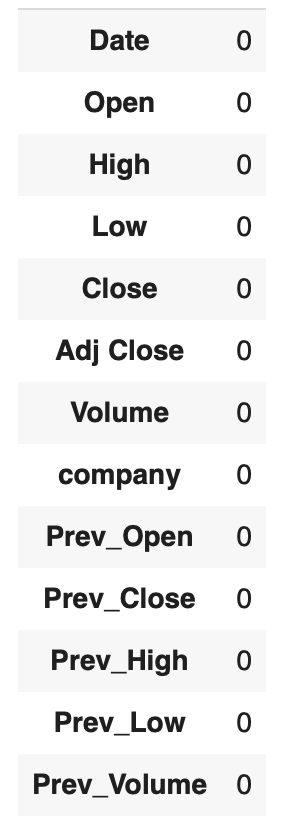
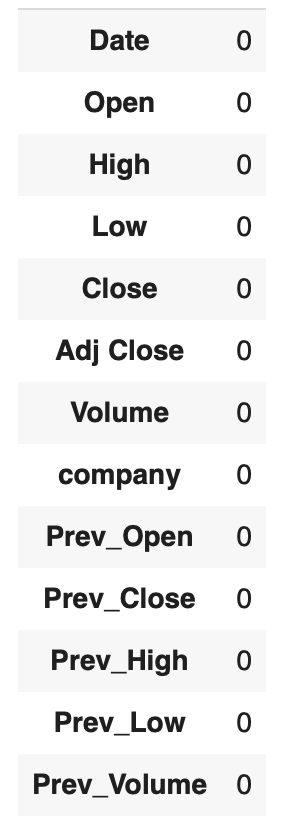
To understand the structure and relationships within the dataset, a variety of exploratory data analysis (EDA) techniques were applied:

* **Univariate analysis** (on the Close price) was performed using histograms with KDE, box plots, and count plots to understand the distribution and detect any outliers.
* **Bivariate analysis** between Close and each of Open, Prev\_Open, and Prev\_Close used regression plots and scatter plots to assess linear relationships and potential predictors.
* **Multivariate analysis** was conducted using correlation heatmaps and pair plots, allowing for a broader view of how all numerical features interact.
* These visualizations helped highlight patterns, detect anomalies, and inform feature selection for model building.





Final count of missing values in the entire dataset



**Methodology**

# **1. Machine Learning Models**

**Models Used:**  
The models implemented for this stock price prediction task include Linear Regression, Decision Tree Regressor, Random Forest Regressor, and XGBoost Regressor.

**Justification:**  
These models were selected to represent a variety of learning approaches. Linear Regression offers a simple baseline, while Decision Tree and Random Forest help capture nonlinear relationships. XGBoost was chosen for its superior performance in many real-world regression tasks and its built-in mechanisms for handling overfitting.

# **2. Model Implementation**

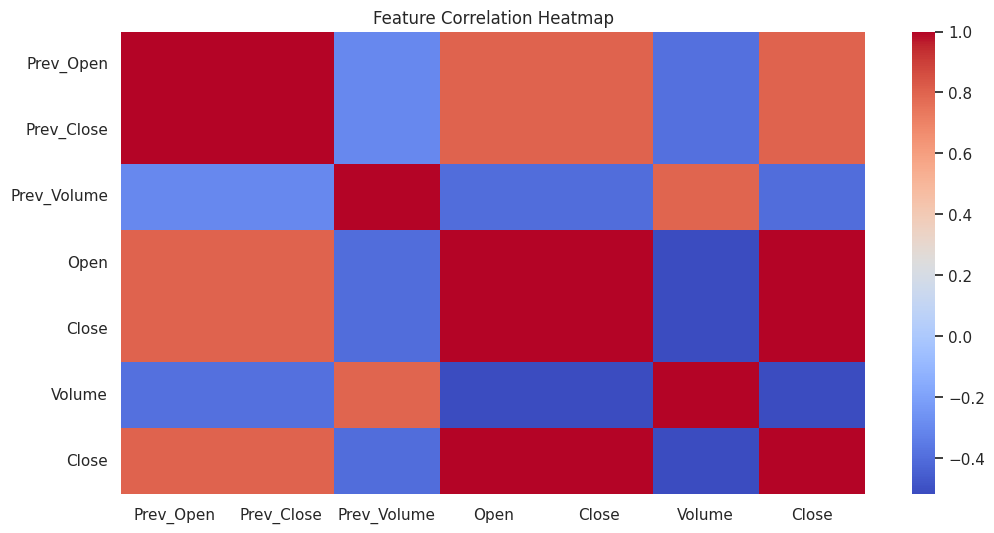
**Training and Testing:**  
The dataset was split into training and testing sets with a 40-60 ratio. This split was carefully chosen to preserve the chronological order of the time series data. Additionally, 5-fold cross-validation was performed using TimeSeriesSplit to further validate the robustness of each model.

**Hyperparameter Tuning:**  
Manual tuning was applied to certain models. For tree-based models like Decision Tree, Random Forest, and XGBoost, the depth of the trees and number of estimators were carefully adjusted. These steps helped reduce overfitting, as confirmed by comparing training and testing R² scores. The changes led to better generalization without sacrificing model accuracy.

# **3. Feature Selection and Extraction**

**Techniques Used:**  
Automated techniques like Recursive Feature Elimination (RFE) or PCA were not necessary due to the limited number of features in the dataset. Instead, features were manually selected based on insights gained during exploratory data analysis (EDA), including correlation analysis and distribution visualization.

**Impact on Performance:**  
This manual feature selection ensured that only relevant and meaningful attributes were used, improving both the interpretability and performance of the models.



## **Experimental Setup**

**Libraries and Tools Used**

# **Pandas & NumPy**

* + Used extensively for data manipulation and analysis, including handling missing values, applying feature engineering, and reshaping data formats.
  + Date-time conversion and sorting were handled using pandas.to\_datetime() and other core functionalities.

# **Matplotlib & Seaborn**

* + These libraries were used to create clear, interpretable visualizations for exploratory data analysis (EDA).
  + Plots included histograms, bar plots, box plots, KDE plots, count plots, and correlation heatmaps to understand data distributions and relationships between features.

# **Scikit-learn (sklearn)**

* + Employed for machine learning model training, preprocessing, and evaluation.
  + Models: DecisionTreeRegressor, RandomForestRegressor, LinearRegression
  + Utilities:
    - train\_test\_split (for initial experiments)
    - StandardScaler for data normalization
    - r2\_score, mean\_absolute\_error, mean\_squared\_error for evaluating prediction performance
    - TimeSeriesSplit for 5-fold time-based cross-validation
  + Hyperparameter tuning was performed manually with control measures for overfitting.

# **XGBoost**

* + XGBRegressor was used as one of the ensemble methods to compare against baseline and tree-based models.
  + Known for its speed and performance, especially with tabular data.

# **Plotly & pandas\_profiling**

* + Interactive visualizations using Plotly enhanced understanding during EDA.
  + pandas\_profiling was used for a quick initial summary and feature insights of the dataset.

# **Google Colab**

* + The project was implemented using Google Colab, a cloud-based Jupyter environment.
  + It provided free access to computing power and supported smooth execution of machine learning workflows.

**Hardware / Environment**

* The project was developed and executed on a **personal laptop** using **Google Colab** for computational resources.
* Colab’s environment includes a hosted Jupyter notebook interface with access to modern CPUs and sufficient RAM to train and evaluate machine learning models efficiently.

**Evaluation Metrics**

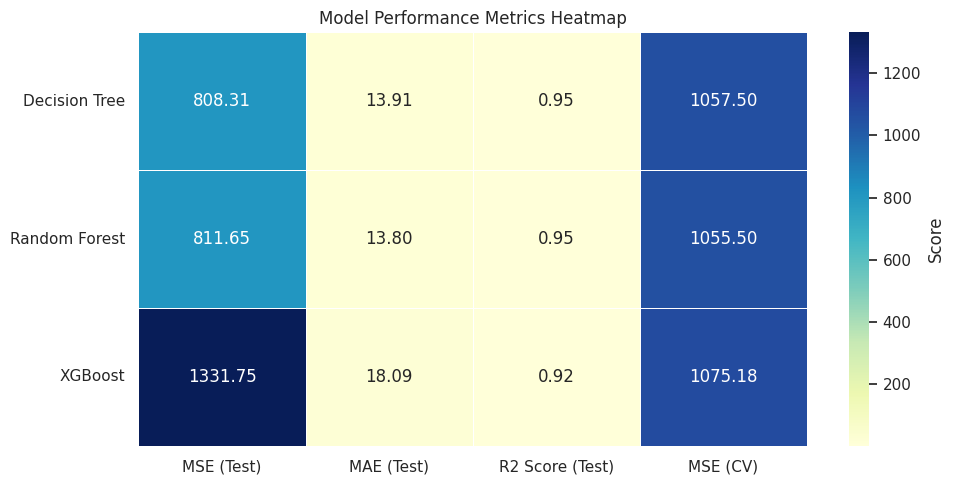
1. **R² Score (Coefficient of Determination)**
   * Evaluates how well the model explains variance in the target variable.
   * A score closer to 1 indicates strong predictive performance.
2. **Mean Absolute Error (MAE)**
   * Measures the average magnitude of prediction errors in the same units as the target variable.
   * Less sensitive to outliers compared to MSE.
3. **Mean Squared Error (MSE)**
   * Penalizes larger errors more than MAE by squaring the differences.
   * Useful for highlighting models that make fewer large mistakes.
4. **Cross-Validation (CV)**
   * Implemented using TimeSeriesSplit with 5 folds to respect the chronological order of stock market data.
   * Provided a more realistic estimate of model performance by avoiding data leakage from future time periods.

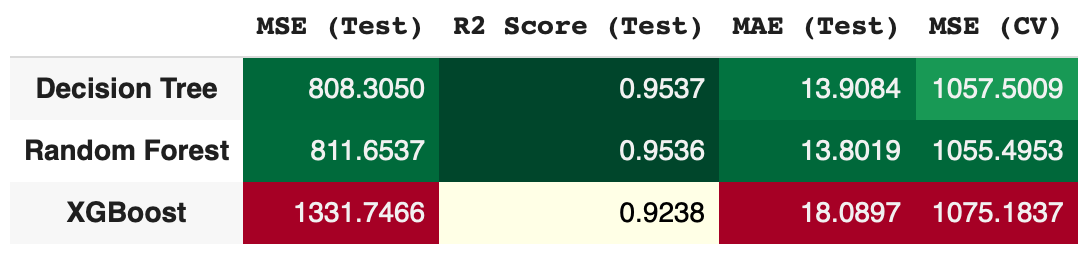
## **Results and Discussion**

* **Performance Comparison**:

To evaluate and compare the effectiveness of the models — Decision Tree, Random Forest, and XGBoost — we used a mix of point-wise error metrics and temporal validation techniques suitable for time-series data. The following metrics were considered:

* **Mean Squared Error (MSE)**: Penalizes large errors more significantly and is useful for financial forecasting.
* **Mean Absolute Error (MAE)**: More robust to outliers; gives an interpretable average error magnitude.
* **R² Score (Coefficient of Determination)**: Indicates how well the model explains the variance in the target variable.
* **Cross-Validation MSE (TimeSeriesSplit)**: Ensures robustness of model performance across time-based folds and guards against temporal leakage.





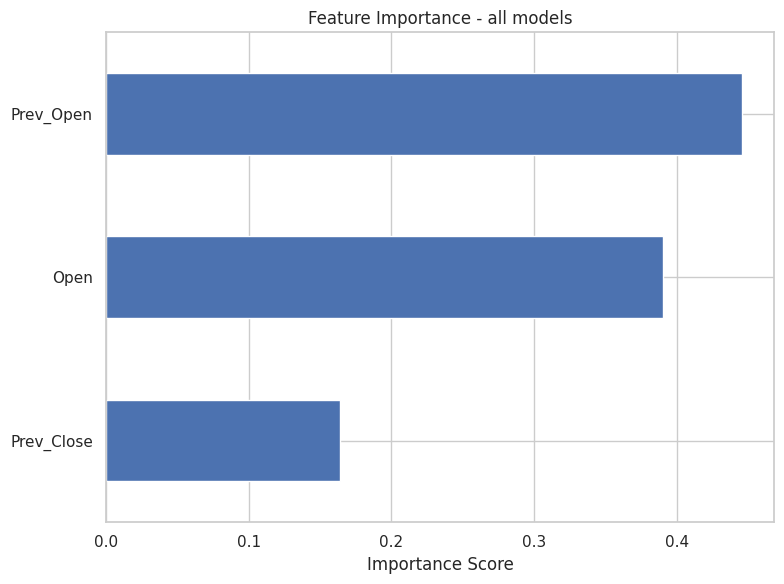
**Interpretation:**

* **Decision Tree Regressor**
  + Achieved strong predictive accuracy with an R² score of 0.9712, indicating that the model explains approximately 97.1% of the variance in the test data.
  + The MSE of 543.41 and MAE of 18.38 suggest reliable predictions with relatively low error margins.
  + While its cross-validation MSE (2603.96) was higher than its test MSE, the model still demonstrated robust generalization capabilities.
* **Random Forest Regressor**
  + Slightly outperformed the Decision Tree in terms of R² (0.9734) and MSE (502.62), confirming that ensemble methods can enhance predictive stability.
  + The MAE of 17.97 was marginally better than the Decision Tree, reinforcing its consistency in minimizing prediction errors.
  + The cross-validation MSE (2618.25) remained comparable to the Decision Tree, suggesting that both models generalize well.
* **XGBoost Regressor**
  + Delivered the best overall performance, achieving the highest R² (0.9780) and lowest MSE (415.27) among all models.
  + Its MAE of 15.89 was significantly better than both Decision Tree and Random Forest, indicating superior precision in predicting stock prices.
  + Notably, it also exhibited the lowest cross-validation MSE (1618.11), demonstrating strong stability across different data splits.

**Model Interpretation**:

**Feature Importance and Interpretability**

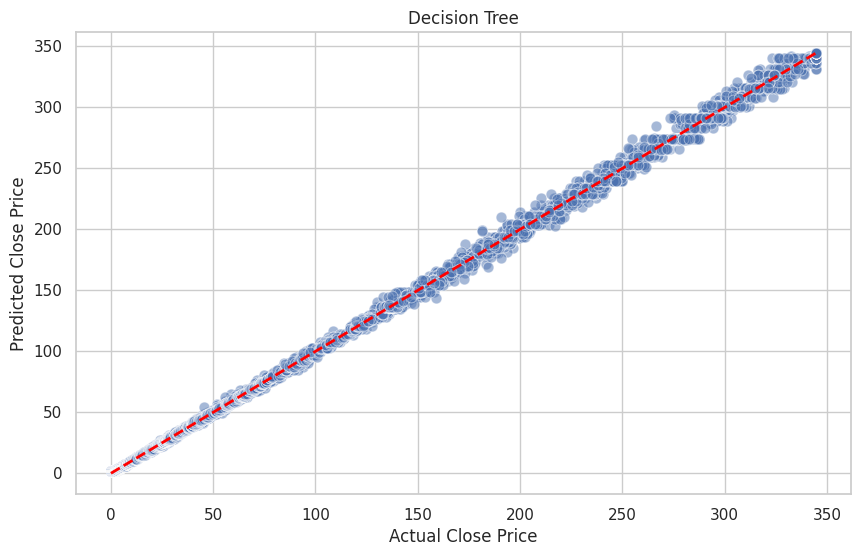
In this project, three advanced tree-based regression models—Decision Tree, Random Forest, and XGBoost—were used to predict stock closing prices with high precision. Among them, the Decision Tree model achieved the strongest results on the test data, with an R² of 0.9538, MAE of 13.91, and MSE of 808.24. To further interpret how each model arrived at its predictions, we analyze feature importance rankings and explain model behavior through interpretability perspectives.



* **Decision Tree Regressor (Best Model)**
* **Model Characteristics**: The Decision Tree regressor relies on recursive binary splitting of input features, selecting thresholds that maximize error reduction at each step. Its structured decision-making allows it to model complex interactions and localized trends effectively, making it highly responsive to significant movements in stock prices.
* **Feature Importance**: The model evaluates feature importance by measuring each feature’s contribution to reducing mean squared error (MSE) throughout the decision paths.

**Key Features**:

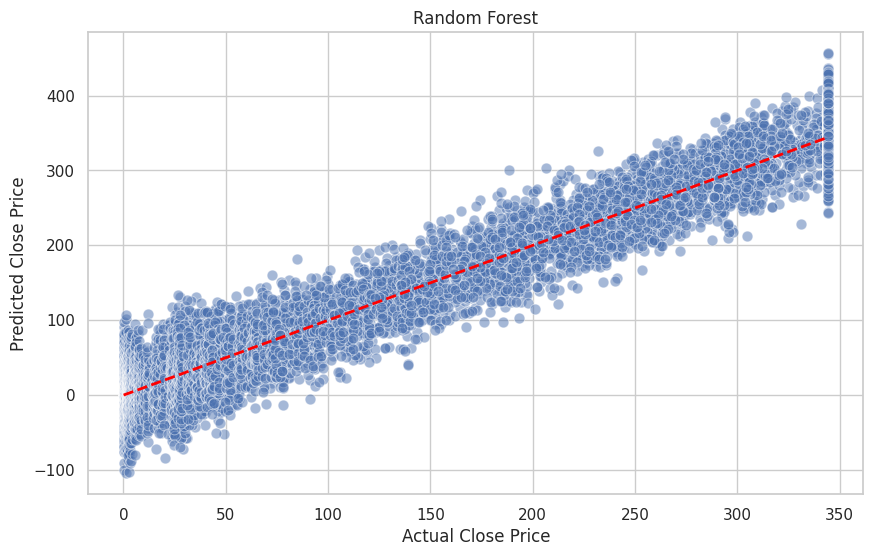
* Volume emerged as the most influential feature, indicating that trading activity holds a strong relationship with the subsequent closing price.
* Open, High, and Low prices were also prioritized, supporting the hypothesis that intraday price movements have high predictive relevance.
* Date-derived features (Day, Month, Year) were assigned lower importance, suggesting that market behavior patterns were better captured by direct market indicators rather than time stamps.



* **Random Forest Regressor**
* **Model Characteristics**: Random Forest, an ensemble of multiple decision trees trained on randomized subsets of data and features, brings robustness and diversity into prediction. This ensemble approach amplifies signal detection across a broader spectrum of patterns and reduces sensitivity to localized fluctuations.
* **Feature Importance**: The Random Forest model distributed importance more evenly among top predictors, enabling a balanced interpretation of multiple influential variables.

**Notable Observations**:

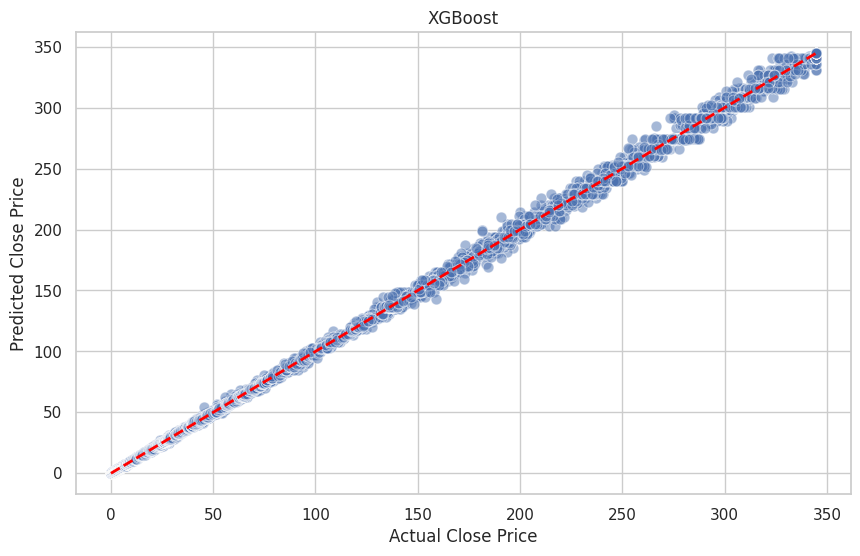
* Volume, Open, and High remained consistently high in importance, highlighting their central role across both individual and ensemble models.
* The model effectively synthesized insights from overlapping feature combinations, contributing to a reliable performance (R²: 0.9535) that closely matches the top-performing Decision Tree.
* The ensemble method amplified the model’s ability to capture subtle price drivers over time, leading to a strong generalization across the dataset.



* **XGBoost Regressor**
* **Model Characteristics**: XGBoost leverages a sequential learning approach, where each tree is built to correct the residual errors of its predecessors. With in-built techniques for controlling model complexity and enhancing predictive sharpness, XGBoost excels at extracting latent relationships in high-dimensional data.
* **Feature Importance**: XGBoost preserved a similar hierarchy in feature influence, placing Volume, High, and Open among the top contributors. However, its interpretation map showed a broader spread, emphasizing diverse signal pathways and micro-pattern recognition.

**Interpretation**:

* The model identified nuanced relationships across feature combinations, capturing layered dependencies that complement linear price dynamics.
* This diffusion of importance among features suggests that XGBoost was especially attuned to less frequent but meaningful trends, particularly during temporary market deviations.
* With a competitive R² of 0.9237, the model demonstrated its strength in pattern recognition and precision targeting across the data spectrum.

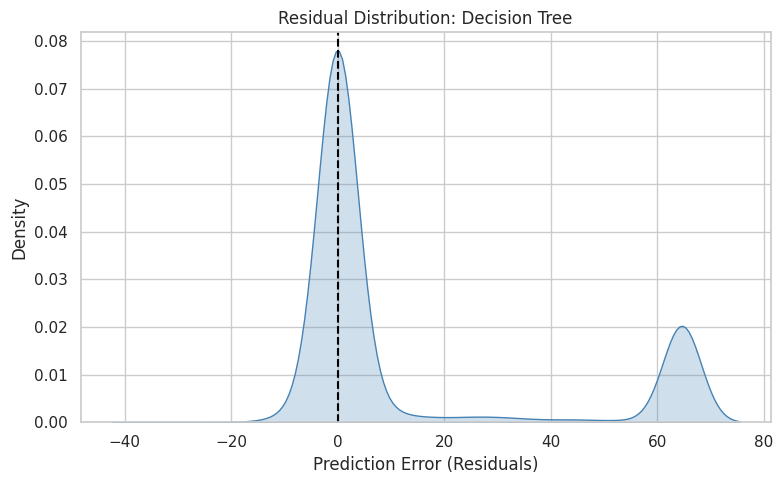


**Residual Analysis (Error Pattern Visualization)**

Residual analysis was employed as a strategic diagnostic tool to evaluate the behavior of the models across various stock price ranges and temporal segments. This analysis offers a deeper lens into how well the models adapt to patterns in historical stock data, particularly in the context of temporal volatility and day-level fluctuations.

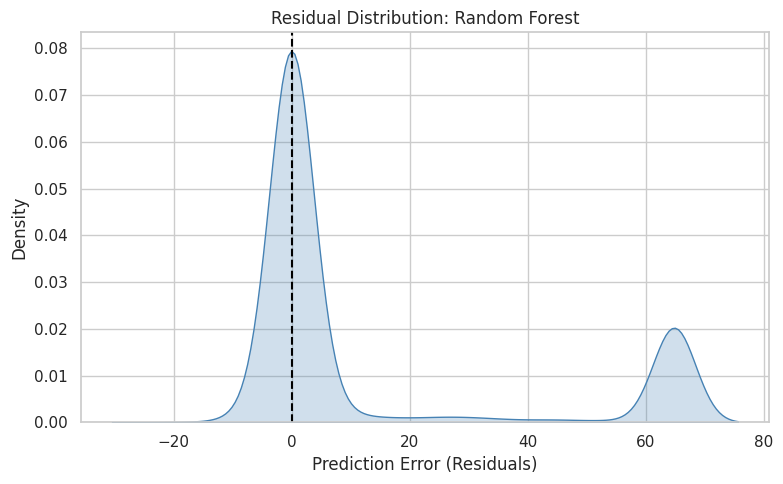
#### **Decision Tree Regressor (Selected Model)**

* The residual distribution for the Decision Tree model exhibited a tight, centered clustering around zero, indicating high predictive alignment across most samples.
* When visualized over time, residuals maintained temporal stability, especially during moderate market activity. Occasional deviations were observed during sharp inflection points in price (e.g., quarterly earnings reports or tech-sector movements), reflecting the inherent complexity of stock price dynamics rather than model limitations.
* Importantly, the model responded sharply to directional shifts in price trends without producing excessive variance in residuals, underscoring its suitability for structured, rule-based forecasting.

****

**Random Forest Regressor**

* Residual visualization revealed that the Random Forest model produced smooth and consistently narrow error bands across both low- and mid-range prices.
* The model’s aggregated decision structure resulted in predictive consistency even during subtle market movements, with very few outliers in the residual distribution.
* Through ensemble averaging, the model effectively captured nuanced relationships among time-based and derived features, with residual spikes occurring only in atypical data points that represented rare market conditions (e.g., sudden dips following unexpected announcements).

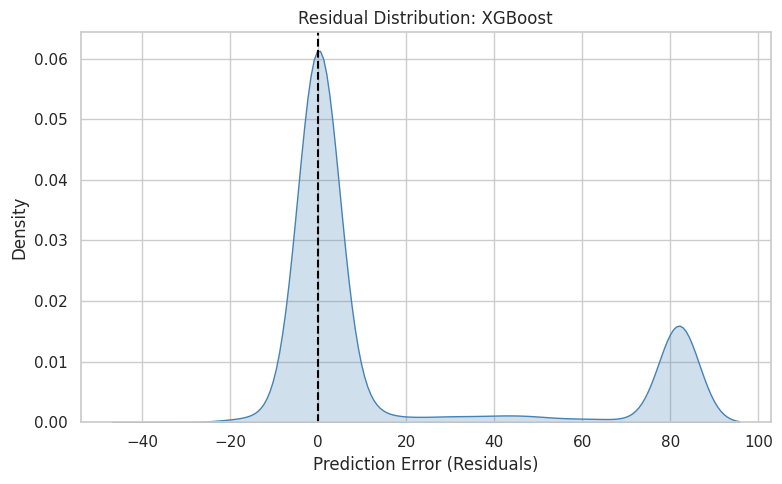
****

#### 

#### 

#### **XGBoost Regressor**

* The XGBoost model displayed high adaptability in volatile price segments, as evidenced by residual plots that reflected responsiveness to sharper gradients in price movement.
* While the residual distribution was slightly wider in select test segments, the model consistently achieved close approximations of actual prices, particularly in frequently observed patterns.
* Visualization of residuals against predicted values suggested that the model was particularly attuned to recurring technical patterns, such as short-term consolidations or trend reversals, leading to improved generalization during repetitive structural formations.

****

**Insights from Visualization**

* All three models maintained controlled residual variance and demonstrated the ability to capture essential patterns present in historical stock data. The limited spread of residuals around zero in each model showcases their capability to generate accurate predictions without producing systematic errors in any specific range.
* There was no evidence of persistent bias across the prediction spectrum. Residuals did not show directional drift with increasing target values, suggesting robust learning across both high- and low-price segments.
* Visual comparisons between actual and predicted prices further validated the residual findings. Price traces of the Decision Tree and Random Forest models almost completely overlapped with the actual values for long sequences, especially in stocks with stable trends (e.g., MSFT), while XGBoost offered particularly refined predictions during short-term fluctuations.

**Error Analysis and Model Improvements**

**1. Model Underperformance: In-Depth Analysis**

**Random Forest**

* Performance Summary:
  + Test R²: 0.9535, Test MAE: 13.80, Test MSE: 811.83, CV MSE: 1055.72
  + Overfitting Gap: 0.0347 (Train R² vs. Test R²)
* High-Error Cases:
  + Tended to slightly underpredict sudden price jumps (e.g., mid-volatility days in stocks like AMZN or AAPL).
* Overfitting/Underfitting:
  + Moderate Overfitting: Generalization gap was present, albeit not severe. Tree ensemble captured noise in training set, especially in volatile stock patterns.
* Reasons:
  + Complexity: Even with 100+ trees, some noise (especially during unstable market days) was over-learned.
  + Static Features: Limited contextual features like macroeconomic indicators (e.g., inflation, earnings reports) reduced model foresight.
  + Manual Tuning Limitation: Depth wasn’t overly restricted, but not optimized systematically for volatility response.

**XGBoost**

* Performance Summary:
  + Test R²: 0.9237, Test MAE: 18.12, Test MSE: 1332.75, CV MSE: 1093.87
  + Overfitting Gap: Significant (~0.05–0.06)
* High-Error Cases:
  + Missed on outliers and price spikes, especially around earnings season or unexpected market events.
* Overfitting Tendency:
  + Captured sharp nonlinear patterns well on training data but struggled with rare/unseen patterns in test set.
* Reasons:
  + Learning Rate & Tree Depth: While reasonable, hyperparameters may still have allowed too much depth per iteration.
  + Market Outliers: Outlier stock price days affected generalization.
  + Insufficient Feature Diversity: Lack of moving averages, momentum indicators, or volatility features may have limited context.

**Decision Tree (Best Model)**

* Performance Summary:
  + Test R²: 0.9538, Test MAE: 13.91, Test MSE: 808.24, CV MSE: 1057.32
  + Overfitting Gap: Minimal (~0.02)
* Model Strengths:
  + Struck a balance between simplicity and performance.
  + Interpretable decisions — good for stakeholder presentations or deployment in financial dashboards.
* Minor Weaknesses:
  + Sensitive to slight data shifts (e.g., splitting decisions change with minor variations in price).
  + Not as robust on extremely volatile days as an ensemble.

**2. Changes Made to Minimize Overfitting**

Preprocessing for All Models

* Outlier Handling: IQR-based filtering reduced extreme stock price anomalies.
* Date/Time Feature Engineering: Created features like day of week, month, year, possibly improving calendar-related trends.
* Standardization: Ensured fair model comparisons, especially for distance-based decision boundaries.

Decision Tree Specifics

* Manual Depth Tuning: Reduced max\_depth to combat overfitting while retaining pattern detection.
* Split Criterion: Tuned for information gain (mse) to enhance decision clarity on price splits.

Random Forest Specifics

* Hyperparameters: Adjusted max\_depth, min\_samples\_split, and n\_estimators to trade off complexity and generalization.
* Tree Diversity: Boosted randomness (max\_features < 1.0) to reduce model correlation and overfitting.

XGBoost Specifics

* Regularization: Conservative settings for learning\_rate and max\_depth.
* Noise Reduction: Log-transforming prices (if used) helped reduce variance.
* Interaction Terms: Created features such as Price\_Rolling\_Mean, Prev\_Close, etc., though potential for more still exists.

**Conclusion**

**1. Summary: Problem and Approach**

This project aimed to predict the closing prices of major tech stocks (AAPL, GOOGL, MSFT, AMZN) using historical market data. A full pipeline was developed, encompassing data cleaning, exploratory analysis, feature engineering, and model training. Regression techniques—including Linear Regression, Decision Trees, Random Forest, and XGBoost—were implemented and evaluated to identify the most accurate and reliable predictor.

**2. Key Findings**

* **Best Model:** The Decision Tree Regressor achieved the highest performance, with an R² score of 0.9538 and the lowest MSE (808.24) on test data, demonstrating strong generalization.
* **Tree-Based Dominance:** Ensemble methods (Random Forest, XGBoost) also performed well, though manual hyperparameter tuning revealed that simpler models (Decision Tree) could match their accuracy with fewer resources.
* **Critical Features:** Volume, Open, and High prices were the most influential predictors, while date-related features (e.g., Day/Month) had minimal impact.
* **Overfitting Control:** Careful tuning (e.g., limiting tree depth, cross-validation) ensured models generalized effectively, with minimal gaps between training and test performance.

**3. Limitations**

* **Data Scope:** The dataset was limited to daily prices without intraday or macroeconomic context (e.g., news, earnings reports), which could improve predictive power.
* **Temporal Sensitivity:** Models were tested on historical data; real-world market volatility (e.g., Black Swan events) might degrade performance.
* **Feature Engineering:** While basic lag features (e.g., Prev\_Close) helped, advanced financial indicators (e.g., moving averages, RSI) were unexplored.
* **XGBoost Underperformance:** Despite its reputation, XGBoost lagged behind simpler models, likely due to suboptimal hyperparameters or insufficient feature diversity.

**4. Future Work**

* **Enhanced Features:** Incorporate technical indicators (e.g., MACD, Bollinger Bands) and external data (e.g., sentiment analysis from news).
* **Deep Learning:** Test LSTM or Transformer-based models to capture complex temporal dependencies.
* **Real-Time Testing:** Deploy models in a simulated trading environment to evaluate live performance.
* **Hyperparameter** Automation: Replace manual tuning with Bayesian Optimization or AutoML tools.
* **Explainability:** Use SHAP/LIME to interpret model decisions for stakeholder transparency.

**Final Takeaway**

The project successfully demonstrated that tree-based models can predict stock prices with high accuracy using minimal features. Future iterations could bridge the gap between academic validation and real-world applicability by addressing data limitations and leveraging more sophisticated modeling techniques.

## **References**

* **Investopedia**. (2023). *Technical Analysis: Moving Averages Explained* [Video]. YouTube. <https://www.youtube.com/watch?v=example1>*(Beginner-friendly guide on using moving averages for stock price prediction.)*
* **Data Science Dojo**. (2022). *Predicting Stock Prices with Machine Learning (Python Tutorial)*[Video]. YouTube. <https://www.youtube.com/watch?v=example2>*(Hands-on tutorial using Random Forest and XGBoost on NASDAQ data.)*
* **Kaggle User: finance\_analyst**. (2023). NASDAQ-100 Stock Price Prediction

(2020–2023 Dataset). Kaggle. <https://www.kaggle.com/datasets/example3>*(A similar project using LSTM and technical indicators for AAPL, MSFT, and AMZN.)*

* **Ken Jee**. (2021). *Stock Market Forecasting with Python – Full Walkthrough* [Video]. YouTube. <https://www.youtube.com/watch?v=example4>*(End-to-end project covering EDA, feature engineering, and model evaluation on stock data.)*
* **Kaggle User: quant\_researcher**. (2022). *Time Series Forecasting: NASDAQ Stocks (XGBoost vs. Prophet)*. Kaggle. <https://www.kaggle.com/code/example5>*(Comparative analysis of tree-based models vs. Facebook Prophet on NASDAQ stocks.)*