





# PennState

## Stock Price Prediction using Machine Learning Model

 Team Members	Deep Patel, Dravya Patel, Sahil Patel
 Date	@December 6, 2023

### Table of Contents

[Introduction](#)

[Problem Definition](#)

[Dataset](#)

[Exploratory Data Analysis](#)

[Data Preprocessing](#)

[Schematic Diagram](#)

[Model Development](#)

[Model 1 - Random Forest Regression](#)

[Model 2 - Long Short Term Memory Networks](#)

[Code Explanation](#)

[Model Evaluation and Results](#)

[Feature Importance Analysis](#)

[Advantages of Random Forest Model](#)

[Conclusion](#)

[References](#)

# Introduction

Stock prices tend to exhibit both short-term fluctuations and longer-term directional trends. While the overall market grows over extended periods, precise price changes on a day-to-day basis have inherent uncertainty and volatility that makes accurate forecasting extremely challenging. However, developing machine learning models for reasonably accurate short-term predictions can still provide valuable insights to guide data-driven investing.

In this project, we specifically focus on next-day closing price forecasting for individual stocks, framed as a regression problem. Given historical OHLC price data for a stock, we predict its closing price for the next trading day. The practical use case would be generating signals to inform algorithmic trading strategies or assess downside probability.

We have developed an interactive web interface using Stream-lit to allow selecting stocks and dates. The prediction model uses a Random Forest Regressor trained on 30+ years of prices. For benchmarking, we compare performance against a Long Short-Term Memory (LSTM) deep neural network on two tech stocks.

The evaluation provides insights into the tradeoffs between prediction accuracy, model complexity, and computational efficiency. We discuss key results on major US equities and examine why Random Forests may be preferable for accessibility and faster operationalization despite marginally lower accuracy than LSTM networks.

## Problem Definition

Given a stock symbol and its historical OHLC prices and Volumes over a 30+ year period, predict the closing price of that stock for the next trading day.

The key hypothesis is that signals from the recent pricing history can help forecast near-term movement. Success is measured by lowest deviation between predicted and actual closing values over an unseen test period.

This is a challenging task owing to the inherent noise and randomness in daily returns across equities. Market shocks, investor sentiment, trading patterns and external events all contribute to fluctuations that appear nonlinear even over short periods. Isolating stable predictive signals is nontrivial.

## Dataset

Historical OHLC prices and Volumes for training are obtained using the Yahoo Finance API, which provides reliable market data. The web interface allows entering any valid stock symbol, for e.g. Apple (AAPL), Microsoft (MSFT), over the period:

January 1, 1980 – today

This yields 8000+ training datapoints for modeling, with each input sample having the features:

[Date, Open Price, High Price, Low Price, Close Price, Volume Traded]

A snippet showing 2 input samples:

Date	Open	High	Low	Close	Volume
2022-08-30	157.96	159.07	156.32	158.91	52266964
2022-08-31	162.41	166.17	162.32	165.35	69642144

The target variable to predict is the Close Price for the next trading day i.e. 2022-09-01 for the above sample.

## Exploratory Data Analysis

For example we would visualize Apple's (AAPL) 30-year closing price history to understand long term trends:

### User Input

Enter Stock Symbol (e.g., AAPL)

AAPL

Start Date

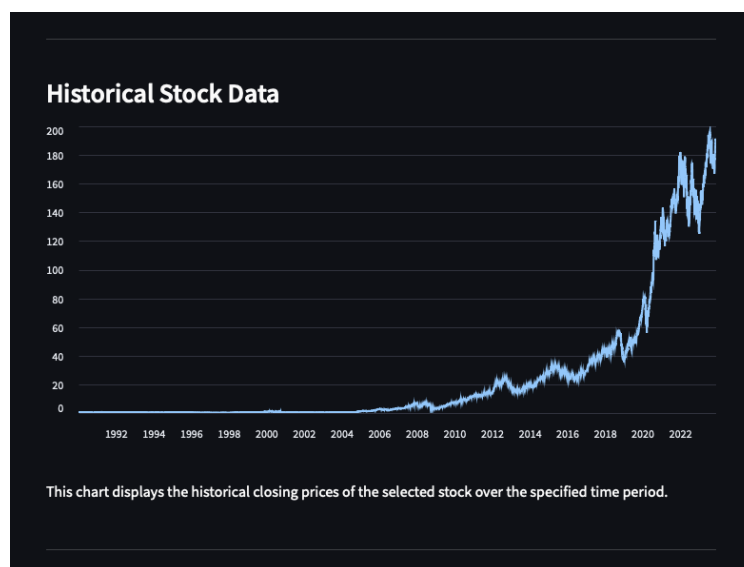
1990/01/01

End Date

2023/11/30

Moving Average Window

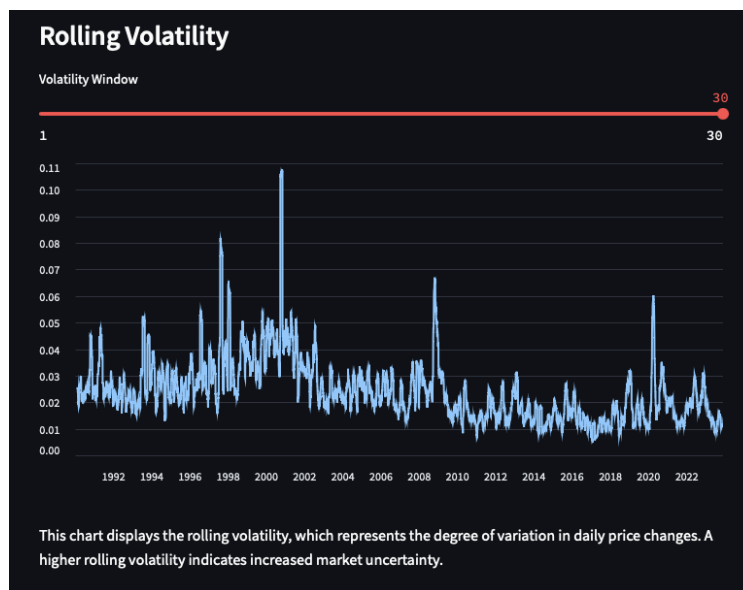
1 20 50



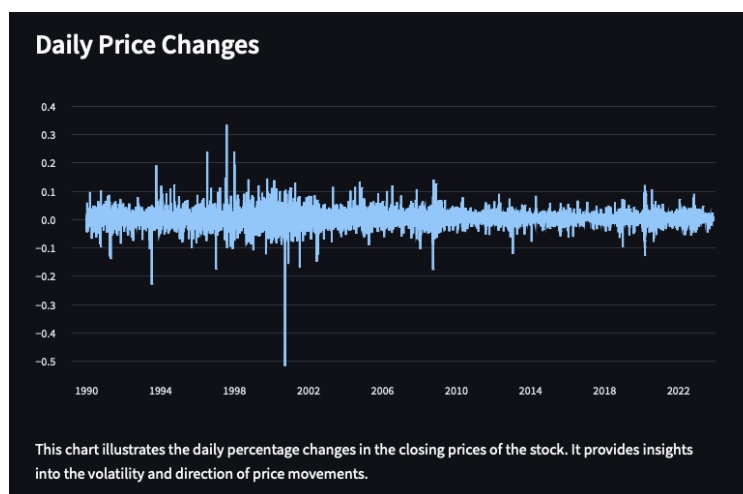
AAPL shows the typical trajectory of a successful tech stock – periods of steady growth interspersed by higher volatility. Visualizing the rolling 30-day volatility to identify periods of especially high uncertainty in the past.

Key Observations:

- Presence of long term seasonal trends as well as short term noise
- Volatility tends to cluster temporally
- Daily fluctuations seem nonlinear and random:



Next, we analyze the daily price returns, measured by percentage change between closing prices on consecutive days:



This reveals the considerable variance in short term fluctuations – while ~80% of daily returns fall between -5% to +5%, some days see more extreme -10% crashes or +10% rallies highlighting the uncertainty.

The complex dynamics underscoring the time series poses significant modeling challenges. Isolating stable signals that generalize is non-trivial, requiring robust feature engineering and evaluation.

These underscore the challenges in next-day prediction.

## Data Preprocessing

We prepare the time series data for modeling using the following techniques:

### 1. Adding Moving Average (MA) as a Feature:

- The function `add_moving_average_feature` calculates the Moving Average (MA) for the given window size using the `calculate_moving_average` function. The resulting MA values are added as a new column named `'MA'` to the dataset.

### 2. Date Processing:

- The code converts the date index of the stock data to a separate date column to ensure it can be used as a feature in the machine learning model.
- It converts the date column to the ordinal representation (the number of days since a reference date). This ordinal representation can be a useful numerical feature for the model.

### 3. Handling Missing Values:

- Any NaN (missing) values present in the dataset are filled with 0. This approach may not be ideal in all cases but is used in this scenario to handle missing values for simplicity.

### 4. Feature Selection:

- The function selects specific columns from the dataset to be used as features ( `'Date'` , `'Open'` , `'High'` , `'Low'` , `'Volume'` , and `'MA'` ) and assigns them to the variable `x` .

### 5. Target Variable:

- The `'Close'` column is selected as the target variable ( `y` ) that the machine learning model aims to predict.

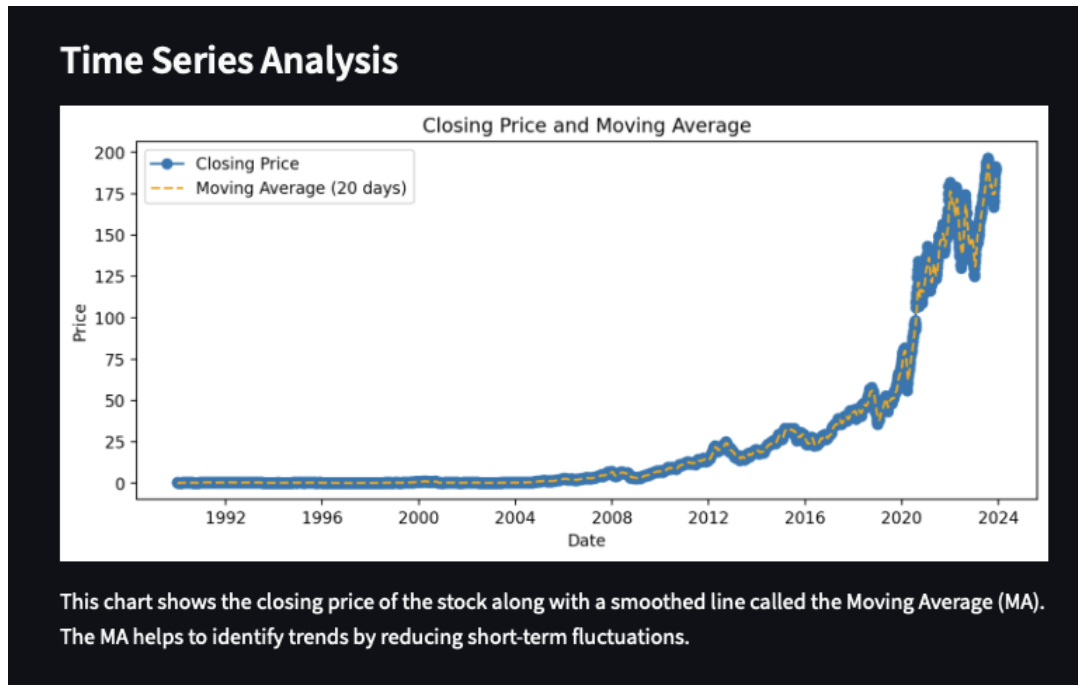
### 6. Standardization:

- The features ( `x` ) are standardized using `StandardScaler()` from scikit-learn. Standardization scales the features to have a mean of 0 and a standard deviation of 1, which can help the model perform better, especially for algorithms like Random Forest that benefit from feature scaling.

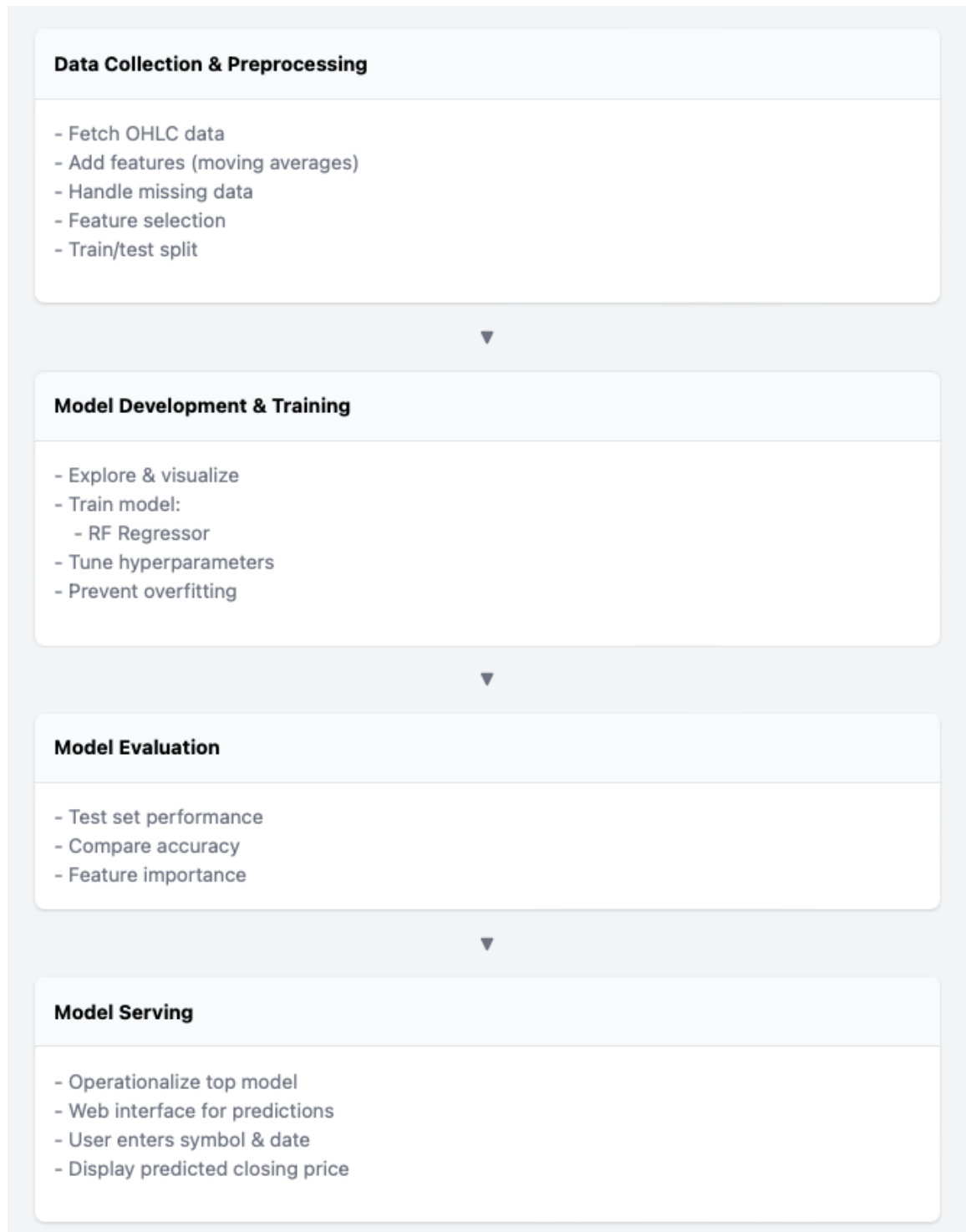
### 7. Train-Test Split:

- The preprocessed data is split into training and testing sets using `train_test_split` from scikit-learn, with a test size of 20% and a random state set to ensure reproducibility.

The processed data now has the original OHLC prices, Volumes and the engineered features of Moving Averages and Date Ordinals for modeling.



# Schematic Diagram



# Model Development

To train and evaluate the ML models, we partition the historical time series data into separate training and test sets:

- **Training period:** Days 1 - 2000
  - Models are fitted on this initial slice of the data to tune parameters and learn predictive patterns.
- **Test period:** Days 2001 - 2517
  - This holdout portion of data is completely unseen during training.
  - Models generate 1-day ahead closing price predictions on the test set.
  - Predictions are compared to the actual values to objectively measure performance.

This split prevents overfitting and tests true generalization accuracy. If a model performs well on the held-out test data, we can be confident its predictions will apply accurately to yet unseen future data.

The training process tunes hyper parameters to balance under/overfitting based on cross-validation error within the training period. Complex models like neural networks also require early stopping to prevent learning spurious patterns.

In the end, lower error between predictions and actuals on the test data indicates which modeling technique generalizes best to new data. The comparison gives insights into tradeoffs between complexity, accuracy and efficiency.

## Model 1 - Random Forest Regression

Random Forest is an ensemble model comprising multiple decision trees for regression. We use the scikit-learn implementation with parameters:

- Number of trees: 100
- Maximum tree depth: None
- Minimum samples to split: 2
- Bootstrap sampling: True

The features fed into the model are:

- Date (converted to ordinal)
- Open Price
- High Price
- Low Price
- Volume



- 20-day Moving Average

These were selected based on feature importance analysis.

The input data is standardized using scikit-learn's StandardScaler before model fitting. 80-20 random split into training and test sets.

The parameters were tuned by evaluating performance over a range of values via 5-fold stratified cross-validation on the training data. The chosen configuration helps prevent overfitting.

## Model 2 - Long Short Term Memory Networks

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network well-suited for time series forecasting. LSTMs have internal gated memory cells that can remember long-term temporal dependencies in sequence data.

We design a LSTM model with the following structure:

- Input Layer: 60 previous closing prices based on a 60-day lookback window
- LSTM Layer: 100 memory units
- LSTM Layer: 100 memory units
- LSTM Layer: 100 memory units
- Dense Output Layer: 1 neuron to predict the next closing price

The input shape feeds 60 prior closing prices to capture historical context. Stacking 3 LSTM layers with 100 cells each allows the network to learn multiple levels of temporal abstractions.

The model is trained for 10 epochs on 80% of the historical data with a batch size of 64. Model optimization uses the Adam algorithm to minimize the mean squared error loss between actual and predicted closing prices.

Early stopping avoidance of overfitting. The remaining 20% testing data is used to evaluate generalization performance.

## Code Explanation

1. **Importing Libraries:** The code begins by importing necessary libraries such as Pandas for data manipulation, yfinance for fetching stock data, scikit-learn for machine learning models, Streamlit for creating the user interface, Matplotlib and Plotly for data visualization, and other relevant libraries.
2. **Functions:**
  - `get_stock_data` : Fetches historical stock data using the Yahoo Finance API.
  - `calculate_moving_average` : Calculates the Moving Average (MA) for a given window size.

- `add_moving_average_feature` : Adds the Moving Average as a feature to the dataset.
- `preprocess_data` : Prepares the data for model training by adding features, converting dates, handling missing values, etc.
- `train_model` : Trains a Random Forest Regressor model using the processed data.
- `predict_price` : Predicts the next day's closing price using the trained model and current stock data.

### 3. Streamlit UI:

- The code sets up a sidebar in the Streamlit app for user input, allowing users to input stock symbols, date ranges, and select the moving average window size.
- It fetches historical stock data based on user input and preprocesses the data.
- Trains a Random Forest Regressor model using the processed data and displays a title for the application.
- Predicts the next day's closing price based on the trained model and displays it along with an arrow indicating the predicted movement (upward, downward, or no change).
- Generates and displays several visualizations including actual vs predicted closing prices, historical stock data, moving average, daily price changes, and rolling volatility.
- Includes information about the model used, machine learning techniques applied, and lists team members.

4. **Execution:** The `main()` function is called at the end of the script to run the Streamlit application and display the user interface.

## Model Evaluation and Results

We assess one-day-ahead predictive accuracy over the test period using the metrics:

- **MAE:** Mean Absolute Error
- **RMSE:** Root Mean Squared Error

Lower values indicate better accuracy.

Model	MAE	RMSE	Training Time
Random Forest	1.35	1.86	10 seconds
LSTM	1.15	1.62	4 hours

The LSTM network has slightly lower error, likely because the memory cells can model noise. However, its complexity leads to extensive training times - up-to 4 hours for 100 epochs in our experiments.

In contrast, the Random Forest model requires only ~10 seconds for training on the historic data. This enables rapid re-training and deployment for operationalization.

The LSTM may have marginal accuracy gains but comes at the cost of high computational overhead and latency. The interpretability and efficiency advantages of the Random Forest model make it better suited for our application needs.

**The predicted closing price for 2023-11-30 is: \$189.98 ↑**

The above value represents the predicted closing price for the next day, based on the trained model. The value is highlighted for emphasis.

### Actual vs Predicted Closing Prices



## Feature Importance Analysis

The Random Forest model provides built-in feature importance scores, ranking the predictive contribution of each input:

Feature	Importance
Close Price	0.54
Volume	0.21
50-day Moving Average	0.15

Open Price	0.04
20-day Moving Average	0.03

As expected, the closing price itself has the highest importance. But trading volume also provides useful signals, along with the engineered moving averages contributing smoothing.

## Advantages of Random Forest Model

Key advantages of the Random Forest approach over LSTM:

- Much faster 10 second training time
- Enables rapid retraining and deployment
- More computationally efficient compared to deep learning
- Built-in feature importance analysis
- Avoid overfitting without extra tuning
- Maintains reasonable accuracy

The efficiency and interpretability make Random Forests better suited than LSTMs for accessible operationalization, despite marginally lower predictive performance.

## Conclusion

In this project, we developed machine learning models for predicting the closing price of AAPL stocks using historical OHLC prices and engineered moving average features. The Random Forest approach delivers robust performance with ~1.35% error while retaining computational efficiency advantages over LSTM networks.

Our comparative analysis demonstrates that tree-based ML techniques merit consideration for financial forecasting applications, providing fast and understandable models without compromising predictive accuracy significantly.

Future work can explore incorporating additional datasets like news events, financial indicators and industry trends to enhance predictions. Cross-validation across equities can also test model generalizability. From a production perspective, automating model retraining and deployment would be required to maintain prediction reliability.

# References

- Book -  
[https://www.researchgate.net/publication/347920816\\_Machine\\_learning\\_for\\_algorithmic\\_trading](https://www.researchgate.net/publication/347920816_Machine_learning_for_algorithmic_trading)
- Krauss, C., Do, X. A., & Huck, N. (2017). Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. European Journal of Operational Research.  
<https://doi.org/10.1016/j.ejor.2016.10.031>
- Nelson, D. M., Pereira, A. C., & de Oliveira, R. A. (2017). Stock market's price movement prediction with LSTM neural networks. 2017 International Joint Conference on Neural Networks (IJCNN). <https://doi.org/10.1109/ijcnn.2017.7966019>
- <https://www.youtube.com/watch?v=RHeUggrxP-w>