METHODOLOGY

Qinglin Gou

March 2024

1 Methodology Overview

In this study, we implemented a comprehensive machine learning pipeline to predict stock prices. Our approach is systematically structured into several key stages, each contributing to the development of a robust forecasting model. The methodology encompasses data cleaning, feature engineering, regression modeling, and extensive evaluation to determine the best-performing combination of feature engineering techniques and machine learning models.

The workflow(Figure 1) begins with raw stock price data, which undergoes a rigorous preprocessing phase to ensure quality and consistency. Next, we apply four sophisticated feature engineering methods to extract informative attributes that could enhance the predictive models' performance. Upon feature extraction, we train four different machine learning models—Linear Regression, Lasso Regression, Random Forest, and LSTM—each with its specific architecture and hyperparameter tuning process designed to optimize performance. The final stage evaluates these models to identify the most effective approach based on various metrics. The overarching goal is to integrate the strengths of each method to forecast stock prices with high accuracy, thus assisting in investment decision-making.

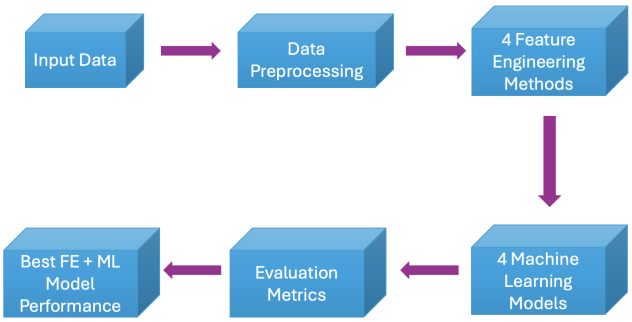

Figure 1: A flowchart representing the machine learning pipeline for stock price prediction.

1.1 Purpose of Methodology

The purpose of the employed methodology is to harness the predictive power of historical stock price data and to explore the efficacy of different feature engineering techniques in enhancing model accuracy. Each step in our pipeline—from data preprocessing to the final model evaluation—was crafted with the intent to align with the overall project goals of achieving reliable and actionable stock price forecasts. The feature engineering methods, including Boruta, PCA, and t-SNE, were selected for their ability to uncover the most relevant features from complex financial datasets. The diverse set of regression models was chosen to capture both linear and non-linear relationships within the data. This methodological framework ensures a comprehensive analysis, allowing us to ascertain the best-performing feature-model combination and, ultimately, provide robust forecasts that can inform strategic investment decisions.

2 Problem Statement

2.1 Define the Problem

In the ever-evolving landscape of financial markets, the ability to accurately predict stock prices is a crucial asset for investment firms looking to optimize their investment strategies. The primary challenge addressed by this machine learning (ML) project is the prediction of short-term stock prices, with a specific focus on Apple Inc. (AAPL) stock. This project is classified under regression, a type of ML problem, where the objective is to forecast future stock prices based on historical data. The regression model aims to understand the relationship between several independent variables, such as technical indicators derived from historical stock prices (e.g., moving averages, volume), and a dependent variable, the future stock price.

2.2 Significance

The significance of solving this problem lies in its substantial impact on investment decision-making and strategy formulation. Accurate short-term stock price predictions can significantly enhance an investment firm's ability to allocate resources efficiently, manage risk more effectively, and ultimately, increase the return on investment (ROI). This capability is particularly vital in the volatile environment of the stock market, where informed decisions can make the difference between substantial profits and considerable losses.

From an academic perspective, this project contributes to the ongoing research in financial market prediction using machine learning techniques. It explores the application of various feature engineering methods and ML models, including Linear Regression, Lasso Regression, Random Forest, and Long Short-Term Memory (LSTM) networks, to the domain of stock price forecasting. This endeavor not only bridges a gap in academic research by applying advanced ML techniques to real-world financial data but also offers insights into the comparative effectiveness of different models in capturing the complexities of stock price movements.

In the context of industry needs, this project responds to the increasing demand for fintech solutions that leverage data science and machine learning to provide actionable insights. By developing a predictive model that accurately forecasts stock prices, this project directly supports investment firms in enhancing their market performance, illustrating a novel application of ML in financial analytics.

Overall, the problem of accurately predicting stock prices is both significant and worthy of exploration due to its potential to revolutionize investment strategies, contribute to academic research in financial machine learning, and fulfill specific industry needs by offering advanced analytical tools for financial decision-making.

3 Data Collection and Preparation

3.1 Data Sources

The data for this project was sourced from Yahoo Finance, a reputable and widely acknowledged platform for financial information. Specifically, we utilized historical stock price data for Apple Inc. (AAPL) covering the period from January 1, 2018, to January 26, 2024. Yahoo Finance is known for its comprehensive coverage of financial data, including daily open, close, high, low prices, and trading volume, making it an ideal source for this analysis. The dataset can be accessed through Yahoo Finance's website at `https://finance.yahoo.com/quote/AAPL/history`, where users can download historical stock data in CSV format.

3.2 Data Description

The dataset comprises approximately 1527 daily records, reflecting the trading activity of AAPL stock over the specified period. Each record includes the following features: Open Price, Close Price, High Price, Low Price, and Volume, along with the calculated technical indicators used as additional features(Table 1). In total, 31 technical indicators, such as the Simple Moving Average (SMA), Exponential Moving Average (EMA), Relative Strength Index (RSI), and Moving Average Convergence Divergence (MACD), were derived to enrich the dataset. The primary label for prediction is the next day's closing price, making this dataset particularly relevant for forecasting short-term stock price movements.

3.3 Preprocessing Steps

The preparation of the dataset for modeling involved several critical preprocessing steps to ensure data quality and model readiness:

Cleaning: Initial inspection of the dataset revealed a high level of completeness with no missing values or duplicates, attributed to the reliability of the data source. However, we conducted a thorough review to confirm there were no anomalies or errors in the data.

Normalization: Given the diverse range of values across different features, particularly technical indicators, we applied normalization to scale the data. This step ensures that no particular feature disproportionately influences the model due to its magnitude.

Feature Extraction: The 31 technical indicators were calculated from the basic price and volume information provided in the dataset. This process involved using both traditional indicators, such as SMA and EMA, and more complex indicators, such as MACD and RSI, to capture different aspects of the stock's behavior.

Data Augmentation: To augment our dataset and improve model robustness, we performed feature engineering to include historical price trends and volatility measures. This approach helped in capturing temporal patterns and trends that are not immediately apparent from daily price data alone.

Splitting: The dataset was divided into training and testing sets, with the training set comprising data from January 1, 2018, to December 31, 2022, and the testing set covering January 1, 2023, to January 26, 2024. This split was designed to evaluate the model's performance on unseen data, reflecting a realistic forecasting scenario.

By meticulously following these preprocessing steps, we ensured that the data was not only clean and consistent but also enriched with features that are crucial for understanding stock price movements. This comprehensive preparation forms the foundation for developing a predictive model capable of accurate short-term stock price forecasting.

Technical Indicator Name	**Calculation**	**Number of days (n)**
Simple Moving Average (SMA)	$\frac{(C_t+C_{t-1}+\cdots+C_{t-n+1})}{n}$	5,10,14,30,50,100,200
Exponential Moving Average (EMA)	$(C_t-SMA(n)_{t-1})*(2/n+1)+SMA(n)_{t-1}$	5,10,14,30,50,100,200
Momentum Indicator (MOM)	$C_t - C_{t-9}$	5,10,14
Stochastic Oscillator (STCK)	$100 * ((C_t - L_t(n))/(H_t(n) - L_t(n)))$	14
Stochastic Oscillator (STCD)	$(100 * ((C_t - L_t(n))/(H_t(n) - L_t(n))))/3$	14
Moving Average Convergence/Divergence (MACD)	$SMA(n) - SMA(m)$	26,13,19,45,25,15
Relative Strength Index (RSI)	$100 - (100/(1 + Avg(Gain)/Avg(Loss)))$	14,28
Williams %R (R)	$((H_t - C_t)/(H_t - L_n)) * 100$	14,28,50,100
Accumulation/distribution index (A/D)	$(H_t - C_t)/(H_t - L_t)$	14
Commodity Channel Index (CCI)	$\big((H_t + L_t + \frac{C_t}{3}) - SMA\big)$ / (0.015 × Mean Deviation)	14,50,100

Table 1: Technical Indicator Formulas

4 Model Selection

4.1 Model Consideration

Linear Regression: Linear Regression is a statistical method that models the relationship between a scalar dependent variable, y, and one or more explanatory variables (or independent variables), denoted X. The case of one explanatory variable is called simple linear regression; for more than one, the process is called multiple linear regression.

The Linear Regression model can be mathematically expressed by the equation:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} + \epsilon_i$$

where:

- y_i is the dependent variable for the i^{th} observation. - $x_{i1}, x_{i2}, \ldots, x_{ip}$ are the independent variables for the i^{th} observation. - $\beta_0, \beta_1, \ldots, \beta_p$ are the coefficients of the model to be estimated. - ϵ_i is the error term for the i^{th} observation.

The OLS estimator for the coefficients is given by the formula:

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

where $\hat{\beta}$ is the vector of estimated coefficients, X is the matrix of input features, and y is the vector of observed values of the dependent variable.

Lasso Regression: Lasso Regression, an extension of Linear Regression, introduces a regularization term to the model's cost function — also known as L1 regularization. This regularization term is the sum of the absolute values of the model coefficients, multiplied by a regularization parameter, λ. The primary objective of Lasso Regression is not only to minimize the sum of squared residuals but also to enforce the model coefficients to be as small as possible, thereby promoting sparsity and aiding in feature selection. The Lasso Regression model is mathematically formulated as follows:

$$\min_{\beta} \left\{ \frac{1}{2n} \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

Here, y_i denotes the dependent variable, x_{ij} represents the independent variables, β_j are the coefficients of the model, β_0 is the intercept, n is the number of observations, p is the number of features, and λ is the regularization parameter that controls the strength of the penalty applied to the size of the coefficients.

Random Forest: Random Forest is an ensemble learning technique utilized for both classification and regression tasks. It operates by constructing a multitude of decision trees during training and outputs either the mode of the classes for classification tasks or the mean prediction of the individual trees for regression tasks.

For regression tasks, the prediction formula can be expressed as:

$$Y = \frac{1}{B} \sum_{b=1}^{B} f_b(X)$$

where:

- Y is the predicted output.
- B is the number of trees in the forest.
- $f_b(X)$ is the prediction of the b^{th} tree on input X.

For classification, the final prediction is determined by the majority vote across all trees:

$$Y = \text{mode}\{f_1(X), f_2(X), \ldots, f_B(X)\}$$

where $f_b(X)$ represents the predicted class for the b^{th} tree.

LSTM (Long Short-Term Memory) : A specialized form of recurrent neural networks capable of learning long-term dependencies. LSTMs are particularly well-suited for time-series forecasting due to their ability to remember information over long periods, which is crucial in stock price movements that are influenced by past trends. Their architecture includes a cell state and three gates (input, output, forget) for managing memory effectively. Given an input sequence $(x_1, x_2, \ldots, x_t)$, LSTM updates through the following steps at each time step t:

The forget gate decides which information is discarded from the cell state:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

The input gate decides which new information is stored in the cell state, and a tanh layer creates a candidate vector $\tilde{C}_t$ of new values:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

The cell state is updated to the new state C_t:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Finally, the output gate decides which parts of the cell state are output, and the hidden state h_t is updated:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

Where: - σ represents the sigmoid function, limiting values between 0 and 1, useful for making binary decisions. - W and b are the weights and biases for the forget gate (f), input gate (i), output gate (o), and cell candidate (C) vectors. - $*$ denotes element-wise multiplication. - $[h_{t-1}, x_t]$ is the concatenation of the previous hidden state and the current input.

4.2 Final Model Selection

In addressing the complex task of stock price prediction, a strategic ensemble of models was adopted to encapsulate the multifaceted nature of financial time series data.

Linear Regression served as the baseline model, providing a reference point for performance due to its simplicity and interpretability. Its function, grounded in the assumption of a linear relationship between independent variables and the stock price, is pivotal for initial diagnostics and understanding of the data.

Lasso Regression was chosen for its proficiency in feature selection, utilizing a regularization parameter that constrains the model to consider only the most significant predictors, which is instrumental in refining the model for better performance.

The **Random Forest** algorithm, known for its capacity to capture non-linear dependencies without succumbing to overfitting, offers a robust alternative to traditional regression models. It operates by constructing a multitude of decision trees, thereby harnessing a more diversified and non-linear decision-making process.

Lastly, **LSTM** networks, with their unique architecture, address the challenge of long-term dependencies in time series data, making them particularly suitable for the temporal dynamics in stock market trends.

Each model was rigorously evaluated for its performance on preliminary tests, with careful consideration given to their established applications in financial forecasting. The selected models collectively enhance the predictive power of our analysis, ensuring a robust approach to stock price prediction.

5 Model Development and Training

5.1 Architecture and Configuration

Linear Regression: The Linear Regression model is implemented within a scikit-learn Pipeline, using `LinearRegression()` as the estimator. It's designed for simplicity and effectiveness in modeling linear relationships. The key configuration parameter `fit_intercept` is tested with values [True, False], allowing the model to evaluate the necessity of an intercept in the linear equation.

Lasso Regression: Lasso Regression, encapsulated within a Pipeline using `Lasso()` as the regressor, introduces L1 regularization to enforce sparsity in the coefficients. The hyperparameters `alpha` and `max_iter` are selected with values [0.5, 1] and [1000], respectively, to control the strength of regularization and ensure convergence of the optimization algorithm.

Random Forest: The Random Forest model, constructed with `RandomForestRegressor()`, employs an ensemble of decision trees to improve prediction accuracy and control overfitting. The parameters `max_depth`, `n_estimators`, `criterion`, `min_samples_split`, and `max_features` are varied with values [8, 16, 24], [100, 200], ['squared_error'], [2, 4], and ['sqrt'], respectively, to fine-tune the complexity and generalization capabilities of the model.

LSTM: In the LSTM network, a Sequential model is constructed with a Bidirectional LSTM layer to enhance learning from the sequential data by processing it in two directions. Following this, a standard LSTM layer captures temporal dependencies, with the number of units in both layers chosen from {50, 100} through hyperparameter optimization. Regularization is applied via dropout layers with rates sampled uniformly between 0.1 and 0.2. The network culminates in a Dense layer with a linear activation function for output. The model is trained using the Adam optimizer with a learning rate selected from a log-uniform distribution within the range [0.001, 0.01], and these hyperparameters, along with batch size options of {32, 64}, are optimized using Optuna to minimize prediction error on the validation set.

5.2 Training Process

The dataset, extending from January 2, 2018, to October 26, 2024, was partitioned into a training set with 1259 records up to January 1, 2023, and a testing set comprising 267 records

thereafter. Prior to model training, four feature engineering techniques were employed: no feature engineering , Boruta, PCA, and t-SNE. Each method prepared the dataset differently, aiming to uncover various intrinsic data characteristics that could enhance model performance.

For the traditional regression models, a `TimeSeriesSplit` with $cv\_n\_splits = 5$ ensured a temporal and sequential split of the dataset, an essential aspect considering the chronological nature of the data. The models — Linear Regression, Lasso, and Random Forest — were trained and their hyperparameters were optimized through a grid search with cross-validation, evaluating performance over multiple time-based folds to address overfitting and evaluate their generalizability.

The LSTM network's training incorporated preprocessing steps like normalization and sequence generation, aligning with the model's need for structured time-series input. Optuna facilitated hyperparameter optimization by conducting trials to find the best configuration, such as the number of LSTM units and dropout rates. Each model's performance, using various feature engineering methods, was meticulously recorded, with results consolidated for a comprehensive comparison. Model predictions were preserved, facilitating further analysis of each approach's effectiveness on the forecasting task.

5.3 Hyperparameter Tuning

The approach to hyperparameter tuning varied across the models to leverage the most suitable optimization techniques for each. For Linear Regression, Lasso Regression, and Random Forest, a grid search method implemented via scikit-learn's `GridSearchCV` was employed. This exhaustive search over specified parameter values for an estimator was particularly effective in systematically exploring the models' parameter space. Specifically, `GridSearchCV` was utilized to identify the optimal settings by iterating over predefined parameter grids:

- **Linear Regression** explored the inclusion of an intercept to understand its impact on model predictions. The hyperparameter `'regressor__fit_intercept'` was tested with options:
  - `True`: To include the intercept in the model.
  - `False`: To exclude the intercept, forcing the model through the origin.
- **Lasso Regression** focused on fine-tuning the regularization strength and the algorithm's convergence through the parameters:
  - `'regressor__alpha'` with options [0.5, 1], determining the intensity of the L1 penalty to enforce sparsity in the coefficients.
  - `'regressor__max_iter'` set to [1000], specifying the maximum number of iterations for the optimization algorithm to ensure convergence.
- **Random Forest** parameters were comprehensively optimized to enhance the model's accuracy and generalizability, including:
  - `'max_depth'` with options [8, 16, 24], controlling the maximum depth of each tree to prevent overfitting by limiting how complex individual trees can become.
  - `'n_estimators'` with values [100, 200], setting the number of trees in the forest, affecting both the performance and computational cost of the model.
  - `'criterion'` set to ['squared_error'], determining the function used to measure the quality of a split.

- 'min_samples_split' with options [2, 4], specifying the minimum number of samples required to further split a node, influencing the tree's growth and complexity.
- 'max_features' set to ['sqrt'], dictating the number of features to consider when looking for the best split, thereby introducing randomness and diversity among the trees.

The LSTM network's architecture was refined through hyperparameter optimization using Optuna, which integrates Bayesian optimization, gradient descent, and evolutionary algorithms for a comprehensive search of the hyperparameter space. This process targeted several key parameters for tuning:

- **Number of LSTM Units:** Optuna explored configurations with `lstm_units` set to either 50 or 100, determining the complexity and capacity of the LSTM layers to capture temporal dependencies in the data.
- **Dropout Rate:** To prevent overfitting, dropout rates were varied within a continuous range, specifically between 0.1 and 0.2. This parameter dictates the proportion of neurons randomly dropped during training, encouraging the model to learn more robust features.
- **Learning Rate:** The optimization algorithm's learning rate was examined across a logarithmic scale from 0.001 to 0.01. This parameter affects how quickly or slowly a model learns, where too high a rate can cause the model to converge too quickly to a suboptimal solution, and too low a rate can slow down the training process.
- **Batch Size:** The size of data batches for training was chosen between 32 and 64. Batch size impacts the gradient estimation during training, balancing between computational efficiency and the stability of the learning process.

These parameters were meticulously selected to ensure a balanced exploration of the model's potential configurations, with Optuna dynamically adjusting its search strategy to efficiently identify the optimal settings. This nuanced approach allowed for significant improvements in the LSTM network's performance, highlighting Optuna's capability to enhance model architecture through informed hyperparameter tuning.