**Apple Stock Purchase Signal Prediction**

*Course Project: Data Science in Practice, Aug 2024*

*Indian Institute of Science, Bangalore*

*Submitted By: Ishwer Kumar, Manish Kumar Singh, Bhaskar Garg, Priti Narayan*

Table of Contents

[Introduction 3](#_Toc184235041)

[Motivation 3](#_Toc184235042)

[Objective 3](#_Toc184235043)

[Challenges 3](#_Toc184235044)

[Dataset 3](#_Toc184235045)

[Input Features 3](#_Toc184235046)

[Output Labels 4](#_Toc184235047)

[Output Label Generation 4](#_Toc184235048)

[Data Preprocessing 4](#_Toc184235049)

[Data Transformation Steps 4](#_Toc184235050)

[Data Merging Strategy 5](#_Toc184235051)

[Moving Average Calculation Summary 5](#_Toc184235052)

[Exploratory Data Analysis & Feature Engineering 6](#_Toc184235053)

[1. Stock Price and Net Income Analysis 6](#_Toc184235054)

[2. BUY/SELL Label Distribution 6](#_Toc184235055)

[3. Cumulative Profit Analysis 7](#_Toc184235056)

[4. Top 10 Correlations with Close Price: 7](#_Toc184235057)

[5. BUY/SELL Signals Scatter Plot 8](#_Toc184235058)

[6. Principal Component Analysis 8](#_Toc184235059)

[Purpose of PCA 8](#_Toc184235060)

[Process 8](#_Toc184235061)

[Benefits Observed 9](#_Toc184235062)

[Model Development 9](#_Toc184235063)

[LSTM Model 9](#_Toc184235064)

[Overview 9](#_Toc184235065)

[Why Use LSTM for Stock Price Prediction? 9](#_Toc184235066)

[Key Steps in the Process 9](#_Toc184235067)

[LSTM Model Architecture 10](#_Toc184235068)

[Model Compilation & Training 10](#_Toc184235069)

[Advantages of Using LSTM for Stock Prediction 10](#_Toc184235070)

[Model Evaluation 11](#_Toc184235071)

[Random Forest Model 11](#_Toc184235072)

[Overview 11](#_Toc184235073)

[Why Random Forest? 11](#_Toc184235074)

[Dataset Split 12](#_Toc184235075)

[Hyperparameter Tuning 12](#_Toc184235076)

[Model Training & Tuning Process 13](#_Toc184235077)

[Best Hyperparameters 13](#_Toc184235078)

[Model Evaluation 13](#_Toc184235079)

[Conclusion and Future Scope 13](#_Toc184235080)

[1. Extending the Model to Multiple Stocks 14](#_Toc184235081)

[2. Why This Is Useful for Investors 14](#_Toc184235082)

[3. Potential Applications and Use Cases 14](#_Toc184235083)

[4. Incorporating Additional Data and Advanced Techniques 14](#_Toc184235084)

[Conclusion 15](#_Toc184235085)

[References 15](#_Toc184235086)

# Introduction

Predicting stock price movements for a tech giant like Apple is challenging due to the complex interplay between market sentiment, company performance, and external factors.

# Motivation

1. **High Investor Interest**: Apple is a market leader, making its stock an attractive target for AI-driven analysis.
2. **Data Availability**: Apple has a long history of detailed quarterly earnings reports and strong stock price data, offering a rich dataset for AI.
3. **Competitive Edge**: Using AI to analyze and predict stock movements can offer traders an advantage in terms of profitability and decision-making efficiency.

Objective

To create an AI model that makes daily buy or sell recommendations for Apple stock based on today's stock price and the company’s financial performance data (quarterly earnings, profit, loss, revenue).

Challenges

* **Volatility**: Stock prices can fluctuate significantly due to external events, affecting model performance.
* **Feature Selection**: Identifying which financial metrics most impact stock price predictions.
* **Market Noise**: Short-term movements may not always reflect a company's true performance.

## Dataset

The stock price data for this project was sourced from the **yfinance** library, which provides reliable historical market data, including daily closing prices, highs, lows, and market capitalization. Additionally, quarterly financial data, such as revenue and net profit, was obtained from [**Kaggle** dataset](https://www.kaggle.com/datasets/ryanmaxwell/apple-historical-financial-data), ensuring comprehensive insights into Apple’s financial performance. Combining these data sources allows for a robust model to predict BUY/SELL signals with both technical and fundamental analysis.

### Input Features

Key financial and stock metrics used for prediction:

* **Closing Price**: The final price at market close.
* **Day’s Highest & Lowest Price**: Intraday price range.
* **Quarterly Revenue**: Financial performance indicator.
* **Net Profit**: Company profitability metric.
* **Market Cap**: Total market value of Apple’s outstanding shares.
* **Other indicators**: Technical and fundamental data.

### Output Labels

* **BUY**: Suggests buying the stock at the closing price.
* **SELL**: Suggests selling the stock at the closing price.

### Output Label Generation

1. **Define Moving Average Period for Label Generation**

labels\_moving\_average\_days is set to 10, specifying the number of days for the rolling average calculation used to generate the buy/sell signals.

1. **Calculate Moving Average**

The 10-day rolling average of the close price is calculated using the rolling(window=labels\_moving\_average\_days) method, and a new column, 10\_Day\_Avg, is added to the DataFrame to store this value.

1. **Shift the Moving Average**

The 10\_Day\_Avg column is shifted down by 10 days (using .shift(-labels\_moving\_average\_days)) to compare each day's close price with the next 10-day moving average. This shifted average is stored in a new column, Next\_10\_Day\_Avg.

1. **Generate Buy/Sell Signals**
   * A new column, signal, is created and initialized with a default value of "SELL."
   * A condition is applied to identify where the close price is lower than the Next\_10\_Day\_Avg. For these rows, the signal column is updated to "BUY."

#### Rational for Choosing this as target variable

* Difficult to predict prices in stock market due to various factors and variations.
* While we can predict the trend up or down for a short or long period of time based on trend like Moving Average Comparison but this are lagging indicators.
* These indicators smoothens the prices but due to lagging indicators all the action has already been done.
* If we can predict the moving average cross with current close price beforehand x days then we can participate in the trend early and get some benefits.
* The model will learn to predict these intersections beforehand.
* During training, we're essentially teaching the model to recognize patterns that lead to these intersections.

# Data Preprocessing

## Data Transformation Steps

The following data pre-processing steps are applied to clean and organize the DataFrame:

1. **Transpose the Data**  
   The DataFrame is transposed to switch rows and columns, allowing for easier data manipulation and have dates as rows.
2. **Set New Header from First Row**  
   The first row of the transposed DataFrame is set as the new header, and this row is then removed from the data.
3. **Remove Columns with All Missing Values**  
   Columns that contain only missing (NaN) values are dropped to streamline the data.
4. **Reset Index and Rename Date Column**  
   The index is reset, creating a new "date" column from the original index values, improving readability.
5. **Standardize Column Names**  
   Column names are formatted by converting them to lowercase and replacing spaces with underscores to ensure consistency and avoid potential naming conflicts.
6. **Remove Duplicate Columns**  
   Any duplicate columns are removed, ensuring each column name is unique in the final DataFrame.

These steps result in a well-structured and standardized DataFrame, ready for analysis.

## Data Merging Strategy

* **Standardized Date Format**:  
  Both datasets were converted to a consistent datetime format, enabling accurate time-based alignment.
* **As-of Merging Approach**:  
  A backward "as-of" merge was applied, ensuring each stock price entry is paired with the most recent available financial data prior to that date.
* **Preventing Data Leakage**:  
  The merging strategy ensures only past financial information is linked to stock prices, avoiding future data influence and maintaining prediction integrity.
* **Comprehensive Dataset**:  
  The final dataset integrates both stock price data and quarterly financial metrics, providing a complete view for accurate model training and analysis.

## Moving Average Calculation Summary

* **Moving Average Period**:  
  A 30-day moving average was calculated to smooth stock price fluctuations over time.
* **Calculation Process**:  
  The moving average was computed using a rolling window applied to the 'close' price, producing a new column labeled "30-Day Moving Average."
* **Visualization**:  
  A plot was created showing both the actual close prices and the 30-day moving average, providing a clear comparison of price trends over time.
* **Enhanced Readability**:  
  The plot includes labeled axes, a descriptive title, a grid for clarity, and a legend to distinguish between the close price and the moving average.

# Exploratory Data Analysis & Feature Engineering

## 1. Stock Price and Net Income Analysis

To understand the relationship between stock price movements and financial performance, we plotted the normalized closing price against the normalized net income. The graph shows how the stock price trends align with the company's net income over time. This helps to identify any correlations or divergences between market valuation and earnings performance.

A graph with orange and blue lines

Description automatically generated

## 2. BUY/SELL Label Distribution

We analyzed the distribution of BUY and SELL signals using a bar chart. This gives insights into the balance of trading signals, helping us understand the model's tendency to favor either signal. A balanced distribution is essential for robust trading strategy development.

A blue rectangular bars with white text

Description automatically generated

## 3. Cumulative Profit Analysis

To evaluate the practical impact of the BUY/SELL signals, we calculated and plotted cumulative profit over time. This analysis demonstrates the potential profitability of following the predicted signals, offering a real-world perspective on model performance.

A graph with a line going up

Description automatically generated

## 4. Top 10 Correlations with Close Price:

1. The feature with the highest correlation to close price is 10 DMA (1.0 correlation), indicating a strong relationship between the 10-day moving average and the stock price.
2. Other highly correlated features include cumulative relative profit (0.93), debt-to-equity ratio (0.81), and price-to-book value (0.77).
3. Lower correlations are seen for features like market capitalization (0.73), research & development (0.71), and common stock (0.68).

A graph of blue rectangular bars

Description automatically generated

## 5. BUY/SELL Signals Scatter Plot

* The chart shows a time series of buy and sell signals for the stock over the past decade.
* There are clear periods of sustained buy signals (e.g. 2010-2012, 2016-2018) as well as sell signals (e.g. 2014-2015, 2018-2019).
* The buy and sell signals appear to generally align with the overall price trend, with buy signals preceding price increases and sell signals preceding price declines.

A graph with a line graph

Description automatically generated

## 6. Principal Component Analysis

In this project, Principal Component Analysis (PCA) was utilized to enhance the predictive model for generating buy/sell signals based on historical close prices and financial indicators, including quarterly net income and profit metrics.

### Purpose of PCA

PCA was employed to address two key challenges:

1. **Dimensionality Reduction:** The dataset contained a high number of correlated features (e.g., close prices, earnings, profits, etc.), leading to potential multicollinearity issues.
2. **Feature Extraction:** PCA enabled the transformation of these correlated features into a smaller set of uncorrelated principal components, retaining the majority of the variance in the data.

### Process

1. **Data Preprocessing:** The financial time series data was standardized to ensure all features had zero mean and unit variance, a prerequisite for effective PCA.
2. **PCA Application:** The standardized data was decomposed into principal components, with the first few components capturing a significant portion of the variance. This reduced the feature set without losing critical information.
3. **Component Selection:** Based on the explained variance ratio, the first *n* principal components (typically those that captured over 95% of the variance) were selected for model input.

### Benefits Observed

* **Reduced Overfitting:** By lowering the feature dimensionality, PCA mitigated overfitting risks, particularly in volatile market conditions.
* **Improved Model Efficiency:** Fewer input features resulted in faster model training and inference.
* **Better Interpretability:** The model’s reduced complexity made it easier to analyze the influence of different financial factors on the buy/sell signals.

# Model Development

## LSTM Model

### Overview

This model focuses on predicting stock prices using a **Long Short-Term Memory (LSTM)** neural network. LSTM is a type of Recurrent Neural Network (RNN) designed to learn from sequential data, making it particularly well-suited for time-series forecasting problems like stock price prediction.

### Why Use LSTM for Stock Price Prediction?

**1. Handling Sequential Data**

**Stock prices are time-dependent**: Historical stock prices are a sequence where each value depends on previous values. Traditional models like linear regression fail to capture this temporal dependency, but LSTM can.

**2. Memory Retention**

**Captures long-term dependencies**: Unlike regular RNNs, which suffer from short-term memory issues, LSTM is designed to **retain long-term patterns** using mechanisms like **cell states** and **gates** (input, output, forget).

**3. Handling Noisy Data**

**Robustness to noise**: Stock price data can be noisy, with sudden fluctuations. LSTM's ability to generalize over long periods makes it more robust in predicting trends amidst noise.

### Key Steps in the Process

**1. Data Normalization**

Stock price data is scaled to a range between **0 and 1** using **MinMaxScaler**. This ensures that all input values are comparable, which enhances the LSTM model's performance and speeds up training convergence.

**2. Dataset Preparation**

The dataset is structured into input-output pairs, where:

* **Input (X)**: A sequence of historical prices over a defined **look-back period**.
* **Output (Y)**: The price immediately following the sequence.

**3. Train-Test Split**

The data is split into **training** (80%) and **testing** (20%) sets to evaluate the model's generalization capabilities. This prevents overfitting and ensures reliable performance on unseen data.

### LSTM Model Architecture

**1. Input Layer**

The input consists of **sequences of historical prices** represented as 3D arrays: [samples, time steps, features].

**2. LSTM Layers**

Two stacked LSTM layers are used:

* **First LSTM Layer**: Outputs sequences to the next LSTM layer (return\_sequences=True).
* **Second LSTM Layer**: Outputs a single vector summarizing the learned information.

**3. Dense Layer**

* A fully connected layer with one neuron is added to predict the next stock price based on the output from the LSTM layers.

### Model Compilation & Training

**Loss Function**

The model is optimized using **Mean Squared Error (MSE)**, a standard loss function for regression problems, minimizing the average squared difference between predicted and actual values.

**Optimizer**

The **Adam optimizer** is chosen for its adaptive learning rate, which ensures faster and more stable convergence.

**Training Parameters**

The model is trained for **30 epochs** with a **batch size of 32**, balancing between computational efficiency and model performance.

### Advantages of Using LSTM for Stock Prediction

1. **Temporal Awareness**: Captures both short-term and long-term dependencies in stock price patterns.
2. **Reduced Overfitting**: Through mechanisms like dropout and gating, LSTM helps mitigate overfitting by selectively retaining relevant information.
3. **Trend Detection**: Effectively models complex time-series patterns, enabling detection of trends that may influence BUY/SELL decisions.

### Model Evaluation

#### Performance Metrics

* **Precision (1.0 class)**: **90.72%** - Most positive predictions are correct.
* **Recall (1.0 class)**: **87.13%** - High proportion of true positive cases identified.
* **F1-Score (1.0 class)**: **88.89%** - Balanced precision and recall.

#### Confusion Matrix

* True Negatives (65) | False Positives (9)
* False Negatives (13) | True Positives (88)

##### Overall Accuracy:

We were able to achieve overall accuracy of **87.43%**

#### Class-wise Summary

* **Class 0.0**: Precision: **83.33%**, Recall: **87.84%**, F1-Score: **85.53%**
* **Class 1.0**: Precision: **90.72%**, Recall: **87.13%**, F1-Score: **88.89%**

#### Conclusion

The model demonstrates strong performance in predicting stock prices with high precision and recall, making it reliable for BUY/SELL decisions.

## Random Forest Model

### Overview

This Random Forest model is used to predict stock price signals (BUY/SELL) using **RandomizedSearchCV** for hyperparameter tuning. The goal is to maximize the **F1-score** by finding the best combination of hyperparameters.

### Why Random Forest?

Random Forest is a powerful and versatile machine learning algorithm that can be highly beneficial for predicting buy/sell signals based on historical close prices, quarterly earnings, net income, profit, and other financial indicators. Here’s why:

**1. Handles High-Dimensional and Correlated Data**

* **Robust to Multicollinearity:** Financial data often has highly correlated features (e.g., close prices over different periods or earnings and profits). Random Forest is not sensitive to multicollinearity because it uses decision trees, which split the data based on feature importance rather than feature correlation.
* **Automatic Feature Selection:** It ranks features by importance, helping to focus on the most relevant financial metrics without needing extensive manual feature engineering.

**2. Nonlinear Relationships and Complex Patterns**

* **Captures Nonlinear Interactions:** The relationship between financial indicators and market movements can be complex and nonlinear. Random Forest, through its ensemble of decision trees, is well-suited for identifying these intricate patterns in the data.

**3. Resilience to Overfitting**

* **Ensemble Learning Approach:** By averaging predictions from multiple trees, Random Forest reduces overfitting, which is particularly beneficial in financial markets where data can be noisy and volatile.
* **Out-of-Bag (OOB) Error Estimation:** This built-in cross-validation mechanism allows for a reliable estimation of model performance without the need for a separate validation set, improving generalization.

**4. Interpretability and Feature Importance**

* **Feature Importance Scores:** Random Forest provides feature importance metrics, helping analysts understand which financial indicators (e.g., quarterly earnings or historical close price trends) have the greatest influence on buy/sell signals.
* **Partial Dependence Plots (PDPs):** These can visualize how specific features affect the model's predictions, offering insights into the decision-making process.

**5. Handles Missing and Noisy Data**

* **Robust to Missing Values:** Random Forest can handle missing data internally by substituting missing values with reasonable estimates (e.g., median values from training).
* **Noise Tolerance:** Its ensemble nature helps filter out noise, making it a good fit for financial data, which often contains anomalies due to market fluctuations.

**6. Scalability and Efficiency**

* **Parallelizable Algorithm:** Random Forest can efficiently handle large datasets, such as years of historical market data, by building trees in parallel, reducing training time.

### Dataset Split

* **Features (X)**: Stock-related attributes excluding the target signal.
* **Target (y)**: Signal representing BUY/SELL decision.
* The dataset is split into **80% training** and **20% testing** sets.

### Hyperparameter Tuning

* **RandomizedSearchCV** is used to optimize parameters such as:
  + **n\_estimators**: Number of trees in the forest.
  + **max\_depth**: Maximum depth of each tree.
  + **min\_samples\_split**: Minimum samples required to split a node.
  + **min\_samples\_leaf**: Minimum samples required at each leaf node.
  + **max\_features**: Feature selection method (‘sqrt’ or ‘log2’).
  + **bootstrap**: Use of bootstrapping in training.
* **Search Space**: A broad grid is searched across **50 iterations** using **5-fold cross-validation**.

### Model Training & Tuning Process

* The model is evaluated using the **F1-score** to ensure balanced precision and recall.
* **RandomizedSearchCV** leverages all available processors (n\_jobs=-1) for efficient parallel computation.

### Best Hyperparameters

After tuning, the model identifies the optimal hyperparameters for the most effective stock prediction. These parameters enhance both the model’s precision and recall in predicting stock signals.

'bootstrap': False

'max\_depth': 72

'max\_features': 'log2'

'min\_samples\_leaf': 2

'min\_samples\_split': 11

'n\_estimators': 63

### Model Evaluation

#### Performance Overview

* Accuracy: 73.45%
* Precision (1): 74.23%
* Recall (1): 76.60%
* F1-Score (1): 75.39%

#### Confusion Matrix

* **True Positives**: 72 | **True Negatives**: 58
* **False Positives**: 25 | **False Negatives**: 22

#### Class-wise Performance

* **Class 0**: Precision **72.50%**, Recall **69.88%**, F1-Score **71.17%**
* **Class 1**: Precision **74.23%**, Recall **76.60%**, F1-Score **75.39%**

#### Conclusion

The model shows solid overall performance with a balanced F1-score. There is room to improve precision and recall, especially for class 0 predictions.

# Conclusion and Future Scope

The current buy/sell signal prediction model was developed and trained exclusively using historical data from **Apple Inc. (AAPL)**, leveraging key financial indicators such as close prices, quarterly net income, and profits. While this single-stock focus allowed us to fine-tune the model for a specific company, there is significant potential to extend its scope across a broader range of stocks and markets.

## 1. Extending the Model to Multiple Stocks

* **Multi-Stock Adaptability:** The next step involves expanding the model to predict buy/sell signals for other stocks across various industries and sectors (e.g., technology, healthcare, energy). By training on diverse datasets, the model can generalize beyond Apple and account for industry-specific dynamics.
* **Portfolio-Wide Optimization:** Investors will benefit from a unified model capable of generating signals for multiple stocks, allowing for diversified portfolios and reduced market risk through exposure to different sectors.

## 2. Why This Is Useful for Investors

* **Broader Investment Opportunities:** Extending the model to other stocks enables investors to discover profitable opportunities across different markets, improving their ability to adjust strategies in response to sector trends.
* **Risk Management through Diversification:** A multi-stock model helps investors balance their risk exposure by spreading investments across different industries, particularly useful during sector-specific downturns.
* **Personalized and Data-Driven Strategies:** The expanded model can cater to investors with varying preferences, such as those focusing on high-growth tech stocks or stable dividend-paying companies.

## 3. Potential Applications and Use Cases

* **Robo-Advisory Platforms:** Financial technology platforms can leverage the extended model to automate stock recommendations for clients, providing diversified portfolio suggestions beyond a single stock.
* **Institutional Trading and Asset Management:** Hedge funds, asset managers, and institutional investors can integrate the multi-stock model for cross-sector analysis and high-frequency trading strategies.
* **Investment Research and Advisory:** Wealth management firms and financial advisors can utilize the model to offer tailored advice on a broader set of stocks, enhancing their advisory services with data-driven insights.
* **Cross-Market Expansion:** The model could eventually support international stocks, commodities, and ETFs, allowing for a comprehensive, multi-asset trading platform suitable for global investors.

## 4. Incorporating Additional Data and Advanced Techniques

* **Sector-Specific Indicators:** Incorporating unique financial metrics for different industries (e.g., R&D spending for tech, oil reserves for energy) will improve prediction accuracy when applied to various sectors.
* **Global Macroeconomic Data:** Integrating international economic indicators, such as interest rates, inflation, and geopolitical risks, will make the model more robust in global markets.
* **Alternative Data Sources:** Using sentiment analysis from news, social media, and analyst reports could enhance prediction accuracy by incorporating market sentiment into buy/sell signals.

# Conclusion

While the current model has been successful in generating buy/sell signals for Apple, its future lies in broadening its scope to include other stocks, industries, and asset classes. This expansion will empower investors with more comprehensive insights, enabling better risk management, diversified portfolio strategies, and a competitive edge in dynamic financial markets.

# References

* Datasets:
  + <https://www.kaggle.com/datasets/ryanmaxwell/apple-historical-financial-data>
  + <https://pypi.org/project/yfinance/>
* <https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/>
* <https://www.geeksforgeeks.org/complete-guide-to-sarimax-in-python/>