Market Intelligence Decision-Making System: Predicting Stock with ML

Rahul Chettri, Karthikeyan Sugavanan, Alexis Musaelyan
Northeastern University
{chettri.ra, ksugavanan, amusaelyan}@northeastern.edu

***Abstract*—This research presents a machine learning-driven **Market Intelligence Decision-Making System** designed to enhance market entry strategies using **S&P 500 data**. The system integrates **predictive modeling** and **Large Language Models (LLMs)** to analyze historical trends, economic indicators, and market conditions. By combining **quantitative analysis** with **natural language processing**, we aim to transform raw financial data into actionable insights, enabling **data-driven investment decisions** in volatile markets.**

***Index Terms*—Machine Learning, Market Prediction, LSTM, S&P 500, Predictive Analytics, AI for Finance**

I. Introduction

Our approach leverages **Long Short-Term Memory (LSTM)**, **XGBoost**, and **Convolutional Neural Networks (CNNs)** to build a comprehensive stock market prediction system. Each model serves a distinct purpose in addressing different aspects of market forecasting.

II. Justification for Model Selection and Comparison

A. LSTM for Time-Series Dependencies

LSTM is particularly well-suited for capturing **long-term dependencies in sequential data**, making it an ideal choice for **financial time-series forecasting**. Since stock prices often exhibit trends influenced by historical movements, LSTM helps identify **underlying temporal relationships** that traditional models might overlook.

B. XGBoost for Feature-Driven Predictions

XGBoost is a powerful **gradient-boosting algorithm** that excels when the dataset is well-structured and engineered with relevant features. By incorporating **technical indicators** such as **moving averages, Relative Strength Index (RSI), and Bollinger Bands**, we aim to enhance predictive performance.

Comparing XGBoost with LSTM allows us to evaluate whether a **feature-rich tabular data-driven model** (XGBoost) can outperform a **pure time-series approach** (LSTM). This comparison helps us identify the most **generalizable and effective model** for future implementations.

C. CNN for Pattern Recognition in Stock Charts

In addition to numerical predictions, we explored **Convolutional Neural Networks (CNNs)** to identify **chart patterns** such as **double tops, head and shoulders**, and other common stock market formations.

CNNs excel at recognizing **spatial patterns**, making them a natural fit for detecting **visual trends** in stock price movements. By implementing CNNs, we can assess the **effectiveness of deep learning in technical pattern recognition**, adding another layer of intelligence to our market prediction framework.

III. Insights from Model Comparison

- By comparing **LSTM and XGBoost**, we can determine whether **sequential memory-based learning** (LSTM) or **feature-engineered tabular learning** (XGBoost) provides more reliable market predictions.
- Evaluating **CNN-based pattern recognition** helps us understand whether **visual stock trends** can be effectively incorporated into a predictive framework.
- This comparison ensures we build a **generalized, robust model** that integrates the **best of time-series forecasting, feature engineering, and deep learning-based pattern recognition**.
- This **multi-model approach** allows us to combine **structured feature-based predictions, sequence learning, and pattern recognition**, making our system more **adaptive and intelligent** in financial market forecasting.

IV. Problem Statement: A Multi-Model Approach for Stock Market Prediction

A. Defining the Problem

Stock market forecasting is inherently complex due to its **dynamic, non-linear, and volatile nature**. Traditional statistical models struggle to capture both **temporal dependencies** and **pattern-based insights** effectively. To address this, we propose a **hybrid predictive framework** leveraging **Long Short-Term Memory (LSTM)**, **Extreme Gradient Boosting (XGBoost)**, and **Convolutional Neural Networks (CNNs)**—each serving a specialized role in market prediction.

B. Machine Learning Category

This study falls under the categories of:

- **Time-series forecasting**: Using past trends to predict future price movements.
- **Regression analysis**: Estimating continuous stock price values.
- **Pattern recognition**: Identifying key stock chart formations.

C. Models Used

Each model in our hybrid approach is tailored for a specific function:

- **LSTM**: Captures **long-term dependencies** in sequential stock price movements.
- **XGBoost**: Learns from **engineered features** (technical indicators) to predict price fluctuations.
- **CNN**: Identifies key **chart patterns** (e.g., head and shoulders, double tops) in stock price visualizations.

V. Significance

This research bridges the gap between **sequential learning, feature-driven analysis, and technical pattern recognition**, ensuring a **robust, multi-faceted market forecasting model**.

A. Academic Relevance

- Contributes to **financial machine learning** by evaluating how different architectures (**sequence-based, feature-based, and pattern-based**) complement each other.
- Assesses the effectiveness of **LSTM vs. XGBoost vs. CNN** in stock market prediction.

B. Industry Applications

- **Quantitative traders, hedge funds, and financial analysts** can leverage this hybrid model for **risk assessment, trading strategies, and market analysis**.
- Can be integrated into **algorithmic trading systems** for automated decision-making.

C. Real-World Impact

- **Retail investors** benefit from a **more reliable prediction system** that combines both **historical trends and real-time market patterns**.
- Helps reduce **uncertainty in investment decisions**, leading to better portfolio management.

VI. Model Selection and Justification

A. LSTM for Time-Series Dependencies

LSTM is particularly well-suited for capturing **long-term dependencies in sequential data**, making it an ideal choice for **financial time-series forecasting**. Since stock prices often exhibit **trends influenced by historical movements**, LSTM helps identify **underlying temporal relationships** that traditional models might overlook.

B. XGBoost for Feature-Driven Predictions

XGBoost is a powerful **gradient-boosting algorithm** that excels when the dataset is well-structured and engineered with relevant features. By incorporating **technical indicators** such as **moving averages, Relative Strength Index (RSI), and Bollinger Bands**, we aim to enhance predictive performance.

Comparing XGBoost with LSTM allows us to evaluate whether a **feature-rich tabular data-driven model** (XGBoost) can outperform a **pure time-series approach** (LSTM). This comparison helps us identify the most **generalizable and effective model** for future implementations.

C. CNN for Pattern Recognition in Stock Charts

In addition to numerical predictions, we explored **Convolutional Neural Networks (CNNs)** to identify **chart patterns** such as **double tops, head and shoulders**, and other common stock market formations.

CNNs excel at recognizing **spatial patterns**, making them a natural fit for detecting **visual trends** in stock price movements. By implementing CNNs, we can assess the **effectiveness of deep learning in technical pattern recognition**, adding another layer of intelligence to our market prediction framework.

VII. Insights from Model Comparison

- By comparing **LSTM and XGBoost**, we can determine whether **sequential memory-based learning** (LSTM) or **feature-engineered tabular learning** (XGBoost) provides more reliable market predictions.
- Evaluating **CNN-based pattern recognition** helps us understand whether **visual stock trends** can be effectively incorporated into a predictive framework.
- This comparison ensures we build a **generalized, robust model** that integrates the **best of time-series forecasting, feature engineering, and deep learning-based pattern recognition**.
- This **multi-model approach** allows us to combine **structured feature-based predictions, sequence learning, and pattern recognition**, making our system more **adaptive and intelligent** in financial market forecasting.

VIII. Data Collection and Preparation

A. Data Sources

The stock market data is sourced from **Yahoo Finance (yfinance API)**, providing **10 years of historical daily stock prices**. The dataset consists of:

- **Date**
- **Open, High, Low, Close, and Volume**

B. Feature Selection

- **Close Price**: The primary feature for model training, as it reflects the overall market sentiment at the end of each trading day.
- **High and Low Prices**: Excluded due to excessive volatility.
- **Open Price**: Omitted since it is highly influenced by pre-market events and news.

C. Data Description

- **Timeframe**: 10 years of daily stock price data.
- **Key Feature**: Close Price (Used for all models: LSTM, XGBoost, and CNN).
- **Additional Features (for XGBoost & CNN)**:
  - Moving Averages (100-day, 200-day, etc.)
  - Peak-Trough Detection for CNN
  - Momentum-based Indicators (RSI, Bollinger Bands, etc.)

D. *Handling Missing Values*

Although the dataset itself does not contain missing values, certain preprocessing steps introduce NaNs:

1) Moving Averages & Feature Engineering (XGBoost):

- The calculation of **100-day and 200-day Moving Averages** leads to missing values at the beginning (e.g., first 100 days for MA_100).
- These NaNs are handled by removing early data points where features are incomplete or by extending the dataset to ensure valid values exist.

2) Correction Strategy:

- Cutting the first 200 rows (if 200-day MA is used) to ensure complete feature values for all data points.
- Alternatively, using a tail-extension approach by replicating early values to maintain consistency.

IX. Preprocessing Steps

A. *Time Series Feature Engineering (For XGBoost)*

- **Moving Averages (MA_50, MA_100, MA_200)**: Smooth out fluctuations and capture long-term trends.
- **Exponential Moving Average (EMA_50, EMA_100)**: Provides greater weight to recent prices.
- **Relative Strength Index (RSI)**: Identifies overbought/oversold conditions.
- **Bollinger Bands**: Measures volatility around the moving average.

B. *Peak Identification for CNN*

Using scipy.signal.find_peaks() to detect:

- **Local Maxima (Peaks)** $\rightarrow$ Represents potential resistance levels.
- **Local Minima (Troughs)** $\rightarrow$ Represents support levels.

Encoding Peaks & Troughs into CNN Input Data:

- Converted into binary peak signals embedded into stock trend images.
- CNN is trained to recognize historical peak-based patterns for trend forecasting.

C. *Time-Series Data Preparation for LSTM*

- Convert Close Prices into **Windowed Sequences** (e.g., past 50 days used to predict the next day's price).
- Normalize Data (**MinMaxScaler**) for better convergence in neural networks.
- **Train-Test Split**:
  - 80% training, 20% testing while maintaining chronological order.

X. Why This Approach?

- **XGBoost**: Leverages historical trends and moving averages to predict future movements.
- **LSTM**: Captures sequential dependencies in stock price changes.
- **CNN**: Detects chart patterns using peaks and troughs, allowing the model to recognize technical formations like head & shoulders, double tops, and support/resistance zones.

XI. Selection of Machine Learning Models

A. *Model Consideration*

For this project, we explored various machine learning and deep learning models to predict stock prices and identify market trends. The selection process involved comparing **traditional machine learning techniques** and **advanced deep learning models** to determine the most effective approach.

B. *Models Considered and Justification*

1) Long Short-Term Memory (LSTM) – Time-Series Dependency Modeling: **Reason for Consideration:**

- LSTM is designed for sequential data and is well-suited for stock price forecasting.
- It captures long-term dependencies and trend patterns in stock movements.

Advantages:

- Learns patterns in historical price trends.
- Handles vanishing gradient issues better than standard RNNs.

Challenges:

- Computationally expensive compared to traditional models.
- Performance depends heavily on sequence length and hyperparameter tuning.
- Prone to overfitting if not properly regularized.

2) Extreme Gradient Boosting (XGBoost) – Feature-Based Learning: **Reason for Consideration:**

- XGBoost is highly effective for structured tabular data, allowing the integration of engineered features such as moving averages, RSI, and Bollinger Bands.

Advantages:

- Fast training and efficient for numerical data.
- Handles missing values and outliers well.
- Provides feature importance insights for model interpretability.

Challenges:

- Does not inherently consider temporal dependencies.
- Requires extensive feature engineering.

3) Convolutional Neural Networks (CNN) – Pattern Recognition in Stock Trends: **Reason for Consideration:**

- CNNs are effective for identifying visual patterns in stock price trends, such as peaks, troughs, and formations like head & shoulders and double tops.

Advantages:

- Recognizes technical chart patterns.
- Generalizes well to unseen stock data when trained properly.
- Less sensitive to short-term fluctuations compared to LSTM.

Challenges:

- Requires transforming numerical data into image-like representations.
- Longer training time compared to XGBoost.

C. Final Model Selection: CNN

Reason for Selection:

- CNN performed well in recognizing stock patterns.
- It captured technical formations effectively, reducing noise from short-term price fluctuations.
- Provided strong interpretability by visualizing market trend patterns.

XII. Evaluation Metrics Used for Comparison

To assess the performance of the models, the following evaluation metrics were considered:

- **Mean Absolute Error (MAE)**: Measures the average absolute differences between predicted and actual values.
- **Root Mean Squared Error (RMSE)**: Penalizes larger errors more significantly than MAE.
- **Mean Squared Error (MSE)**: Provides an overall measure of prediction deviation.
- **R² Score**: Measures how well predictions fit actual data.
- **Accuracy (%)**: Used primarily for CNN-based classification of stock trend patterns.

XIII. Model Development and Training

A. Architecture and Configuration

1) LSTM:

- **Number of Features**: 1 (Close price)
- **Data Split**: 70% Training, 30% Testing

2) XGBoost:

- **Number of Features**: 6 (Technical indicators such as moving averages, RSI, Bollinger Bands)
- **Data Split**: 80% Training, 20% Testing

3) CNN:

- **Number of Features**: Close price with detected peaks and dips
- **Data Split**: 80% Training, 20% Testing

B. Training Process

- **Early Stopping**: Implemented for LSTM and CNN to stop training when validation loss no longer improves.
- **Dropout Layers**: Added to LSTM and CNN architectures to prevent overfitting.
- **Batch Normalization**: Used in CNN to stabilize training.
- **Cross-Validation**: Applied in XGBoost to optimize performance and reduce variance.

C. Hyperparameter Tuning

- **Grid Search**: Used for all models to find the optimal batch size, learning rate, and number of layers.
- **Dropout Rate**: Tuned to reduce overfitting.
- **Epoch Selection**: Adjusted based on validation loss to prevent unnecessary overtraining.

XIV. Evaluation and Comparison

A. Key Performance Metrics

To evaluate and compare the models, the following performance metrics were used:

- **Root Mean Squared Error (RMSE)**: Measures the prediction error, giving higher weight to large deviations.
- **Mean Squared Error (MSE)**: The average squared difference between predicted and actual values.
- **Mean Absolute Error (MAE)**: Measures the absolute difference between predicted and actual values.
- **R² Score**: Measures how well the model explains the variance in stock prices, with values closer to 1 indicating better performance.
- **Accuracy (%)**: Used for CNN to evaluate its effectiveness in detecting stock market patterns.

B. Model Performance Comparison

1) CNN Model Performance:

- **RMSE**: 357.96
- **Accuracy**: 97.42%
- **Specialization**: CNN was primarily used for pattern recognition, identifying key stock trading patterns such as *head & shoulders, double tops, and trend reversals*.

2) LSTM Model Performance: The best performance was achieved at 100 epochs with batch size 32:

- **RMSE**: 99.10
- **MSE**: 9,044.55
- **R² Score**: 0.9483
- **MAE**: 90.54

LSTM significantly improved with more training, reducing RMSE from **242.21 (30 epochs)** to **99.10 (100 epochs)**. It effectively captured **sequential dependencies** in stock price movements.

3) XGBoost Model Performance: XGBoost used engineered features (e.g., moving averages, RSI, Bollinger Bands) and had consistent improvements across epochs. The best results at 100 epochs with batch size 32:

- **RMSE**: 105.00
- **MSE**: 11,025.00
- **R² Score**: 0.9700
- **MAE**: 85.00

XGBoost performed slightly worse than LSTM in RMSE but had a **higher R² score**, indicating better variance explanation. It was computationally efficient and handled **feature-based learning** effectively.

C. Final Model Selection: CNN

Although LSTM and XGBoost were strong candidates for numerical forecasting, CNN was chosen due to its superior **accuracy (97.42%)** in stock pattern recognition.

Reasons for Selection:

- More robust against **short-term volatility** than LSTM.
- Effectively recognized **technical patterns**, aiding traders in making strategic decisions.

- Superior at detecting **trend reversals and high-probability trading setups**.
- Pattern-based trading signals can complement numerical forecasting approaches.