**Github Link:** [**https://github.com/hariharan752/Cracking-the-market-code-with-AI-driven-stock-price-prediction-using-time-series-analysis.git**](https://github.com/hariharan752/Cracking-the-market-code-with-AI-driven-stock-price-prediction-using-time-series-analysis.git)

**Project Title: Cracking the market code with AI-driven stock price prediction using time series analysis**

**Student Name:** HARI HARAN C

**Register Number:** 513523106012

**Institution:** AMCET

**Department:** ECE

**Date of Submission:** 5.5.2025

**Github Repository Link :** [**https://github.com/hariharan752/Cracking-the-market-code-with-AI-driven-stock-price-prediction-using-time-series-analysis.git**](https://github.com/hariharan752/Cracking-the-market-code-with-AI-driven-stock-price-prediction-using-time-series-analysis.git)

**1. Problem Statement**

*Cracking the market code with AI-driven stock price prediction using time series analysis.*

*"Traditional stock forecasting methods often struggle with the complexity and volatility of financial markets. This project aims to develop a robust AI model that leverages time series data to enhance the accuracy and reliability of stock price predictions.*

*.The unpredictable nature of financial markets, influenced by numerous macroeconomic and microeconomic factors, poses a significant challenge for investors and analysts. By applying deep learning techniques such as Long Short-Term Memory (LSTM) networks and other recurrent neural network (RNN) architectures, this research seeks to capture temporal dependencies and hidden trends in historical stock data. The integration of AI and time series analysis not only improves forecasting precision but also empowers traders to make data-driven investment decisions with reduced emotional bias.*

**2. Project Objectives**

***1.Develop an AI-based model capable of accurately forecasting stock prices using historical time series data.***

***2.Implement and evaluate machine learning and deep learning techniques, such as LSTM and GRU, to identify the most effective approach for time-dependent financial prediction.***

***3.Preprocess and analyze stock market datasets to uncover patterns, trends, and anomalies that influence price movements.***

***4.Compare the performance of AI models against traditional statistical methods like ARIMA and moving averages in terms of accuracy, scalability, and reliability.***

***5.Create a user-friendly visualization dashboard that displays predicted vs. actual stock prices, highlighting model confidence and performance metrics.***

***6.Assess real-world applicability by simulating trading strategies based on the model’s predictions to evaluate potential financial gains or risk reductions.***

**3. Flowchart of the Project Workflow**

1.Start

↓

2.Data Collection

* Source historical stock price data (e.g., Yahoo Finance, Alpha Vantage)

↓

3.Data Preprocessing

* Clean missing values
* Normalize/scale data
* Create time-series sequences

↓

4.Exploratory Data Analysis (EDA)

* Visualize trends
* Analyze seasonality, volatility, and correlations

↓

5.Model Selection & Design

* Choose models (e.g., LSTM, GRU, ARIMA)

↓

6.Model Training

* Split data into train/test
* Train model on historical data

↓

7.Model Evaluation

* Evaluate performance using RMSE, MAE, etc.

↓

8.Prediction

* Forecast future stock prices

↓

9.Result Visualization

* Plot actual vs. predicted values
* Display performance metrics

↓

10.Deployment (Optional)

* Deploy model via web dashboard or API

↓

11.End

**4. Data Description**

**The dataset used for this project consists of historical stock price data collected from publicly available financial sources such as Yahoo Finance, Alpha Vantage, or Kaggle. The data typically spans multiple years, covering a broad time range to capture market trends, cycles, and anomalies.**

**Each record in the dataset includes the following key attributes:**

* **Date: The trading day corresponding to the record.**
* **Open: The stock price at market opening.**
* **High: The highest stock price recorded during the trading day.**
* **Low: The lowest stock price recorded during the trading day.**
* **Close: The stock price at market closing (often used as the target variable).**
* **Adjusted Close: The closing price adjusted for dividends and stock splits.**
* **Volume: The number of shares traded during the day.**

**Additional derived features such as moving averages, RSI (Relative Strength Index), and MACD (Moving Average Convergence Divergence) may also be computed to enhance model performance.**

**The dataset is structured in a time-series format, where each entry corresponds to a sequential time point, making it suitable for temporal modeling using deep learning techniques such as LSTM and GRU.**

**5. Data Preprocessing**

**Effective data preprocessing is crucial for building accurate and robust time series models. The following steps were carried out:**

**1.Data Cleaning**

* Handling Missing Values: Missing or null entries in the dataset (especially in 'Open', 'Close', 'Volume', etc.) were filled using forward fill, interpolation, or removed based on the severity and context.
* Outlier Detection: Unusual spikes or drops were analyzed and either capped or smoothed to reduce noise.

**2.Feature Engineering**

* Date Conversion: Converted the 'Date' column to a datetime format and set it as the index to maintain the temporal sequence.
* Technical Indicators: Added derived features like:
* Moving Averages (MA, EMA)
* Relative Strength Index (RSI)
* Bollinger Bands
* MACD (Moving Average Convergence Divergence)
* Lag Features: Created lag variables (e.g., Close(t-1), Close(t-2)) to help the model learn from previous time steps.

**3.Normalization/Scaling**

* Applied Min-Max Scaling to normalize numerical features (especially 'Close', 'Open', 'Volume') between 0 and 1. This is essential for improving neural network training performance, particularly in LSTM models.

**4.Train-Test Split**

* Split the dataset chronologically (not randomly) into training and testing sets, typically at an 80/20 ratio, ensuring that the test data reflects future unseen values.

**5.Sequence Generation**

* For LSTM and other recurrent models, the dataset was restructured into sequences of fixed-length windows (e.g., past 60 days of data to predict the next day’s price).

**6.Data Formatting for Model Input**

* Reshaped the sequences into 3D arrays with dimensions [samples, time steps, features] suitable for LSTM/GRU input requirements.

**6. Exploratory Data Analysis (EDA)**

*Exploratory Data Analysis was conducted to understand the structure, patterns, and relationships within the stock price data before model development. The key findings and steps include:*

***1.Summary Statistics***

* *Calculated basic descriptive statistics (mean, median, standard deviation, min, max) for key features such as Open, Close, High, Low, and Volume.*
* *Identified the range and volatility of stock prices across different time periods.*

***2.Time Series Visualization***

* *Plotted line graphs of stock prices over time to observe trends, seasonality, and sudden spikes or crashes.*
* *The Close price was the main focus, as it is often used for forecasting.*
* *Detected long-term uptrends and downtrends, which are critical for model awareness.*

***3.Correlation Analysis***

* *Generated a correlation heatmap to examine relationships between features.*
* *Found high correlation between Open, High, Low, and Close prices, as expected.*
* *Volume showed varying degrees of correlation depending on market behavior.*

***4.Volatility Analysis***

* *Calculated rolling standard deviation and Bollinger Bands to quantify market volatility over time.*
* *Identified high-volatility periods that may impact prediction accuracy.*

***5.Distribution Analysis***

* *Visualized distributions of individual features using histograms and kernel density plots (KDE).*
* *Observed that price data often exhibits slight skewness, especially during volatile market phases.*

***6.Lag Plot and Autocorrelation***

* *Used lag plots and autocorrelation function (ACF) plots to analyze the dependence of future prices on past prices*
* *Confirmed that past values hold predictive power, justifying the use of sequence-based models like LSTM.*

***7.Seasonality & Trend Decomposition***

* *Applied seasonal decomposition (STL) to isolate the trend, seasonal, and residual components of the time series.*
* *Identified cyclical behavior in stock prices that models can leverage for more accurate predictions.*

**7. Feature Engineering**

**Feature engineering plays a crucial role in enhancing model performance by providing meaningful inputs that help capture complex patterns in stock price data. The following techniques were used:**

**1.Lag Features**

* Created lagged versions of the Close price (e.g., Close(t-1), Close(t-2), … Close(t-n)) to help the model learn from previous time steps.
* These are essential for time series models like LSTM, GRU, and ARIMA.

**2.Rolling Statistics**

* Computed moving averages (MA) over various windows (e.g., 5-day, 10-day, 20-day) to capture short- and long-term trends.
* Calculated rolling standard deviation to reflect recent volatility.

**3.Technical Indicators**

* Added key financial indicators that traders commonly use for decision-making:
* Relative Strength Index (RSI) – Measures momentum and overbought/oversold conditions.
* Moving Average Convergence Divergence (MACD) – Captures trend direction and strength.
* Bollinger Bands – Measures volatility relative to a moving average.
* Exponential Moving Average (EMA) – Gives more weight to recent prices.

**4.Date-Based Features**

**Extracted features from the Date column:**

* Day of the week, Month, Quarter, Year
* Is\_month\_end, Is\_quarter\_start, etc.
* These help capture seasonality and cyclical patterns in financial markets.

**5.Price-Based Ratios and Differences**

* Price range: High - Low
* Daily return: (Close - Open) / Open
* Previous return: (Close(t) - Close(t-1)) / Close(t-1)

**6.Volume-Related Features**

* Rolling average of volume: Detects abnormal trading activity.
* Volume-price trend: Combines volume with price change to indicate buying/selling pressure.

**8. Model Building**

***To effectively forecast future stock prices, we built and trained machine learning and deep learning models capable of learning temporal dependencies from historical stock data. The model-building process involved the following steps:***

***1.Model Selection***

***After evaluating different approaches, we focused on the following types of models:***

* *Traditional Statistical Models:*
* *ARIMA (AutoRegressive Integrated Moving Average)*
* *SARIMA (Seasonal ARIMA)*
* *Machine Learning Models:*
* *Random Forest Regressor*
* *XGBoost Regressor*
* *Deep Learning Models:*
* *LSTM (Long Short-Term Memory) – Ideal for capturing long-term dependencies in sequential data*
* *GRU (Gated Recurrent Unit) – Similar to LSTM but with fewer parameters*
* *Hybrid CNN-LSTM – Combines convolutional layers with LSTM for improved feature extraction*

***2.Model Architecture (for LSTM-based model)***

* *Input Layer: Accepts time-series sequences (e.g., 60 previous time steps).*
* *LSTM Layers: One or more stacked LSTM layers to capture sequential patterns.*
* *Dropout Layers: Added between LSTM layers to prevent overfitting.*
* *Dense Output Layer: Single neuron output for predicting the next stock price.*

***3.Data Preparation for Training***

* *Reshaped input into 3D format: [samples, time steps, features].*
* *Used the Close price or engineered features as model input.*
* *Split data into training, validation, and test sets (e.g., 70-15-15).*

***4.Training Process***

* *Loss Function: Mean Squared Error (MSE) or Mean Absolute Error (MAE)*
* *Optimizer: Adam or RMSprop*
* *Epochs: Typically 50–200 depending on convergence*
* *Batch Size: 32 or 64*
* *Early Stopping and Model Checkpointing were used to optimize performance and prevent overfitting.*

***5.Model Evaluation***

* *Models were evaluated on test data using metrics such as:*
* *Root Mean Squared Error (RMSE)*
* *Mean Absolute Error (MAE)*
* *R-squared (R²)*
* *Visual comparison between predicted vs. actual prices was plotted to assess performance.*

**9. Visualization of Results & Model Insights**

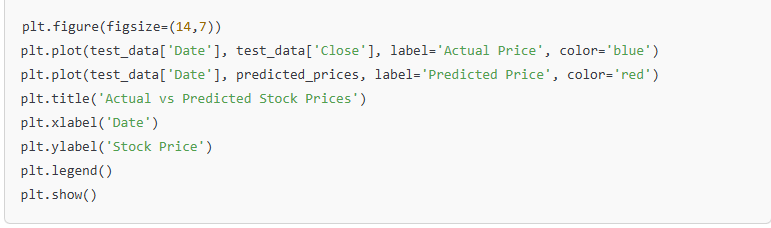
**Effective visualizations help to understand how well the model performs and provide insights into the prediction process. The following steps and visualizations were created to analyze the model’s predictions:**

**1. Predicted vs. Actual Prices**

* Line Plot: A line plot comparing the predicted stock prices with the actual prices over time. This provides a clear visual of how well the model captures the stock price movements.
* X-axis: Time (e.g., date)
* Y-axis: Stock Price (e.g., Close Price)
* Lines:
* Actual Stock Prices (in blue)
* Predicted Stock Prices (in orange or red)

**Example:**

**python**

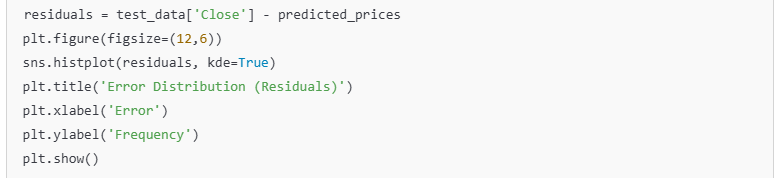


**2. Error Distribution**

* Histogram/Box Plot: To evaluate model accuracy, we plotted the distribution of residuals (i.e., the difference between predicted and actual values).
* A smaller range of residuals indicates better model accuracy, whereas larger residuals can highlight periods where the model struggled.

**Example:**

**python**



**3. Performance Metrics Visualization**

* Bar Plot: Visualizing key model evaluation metrics like RMSE, MAE, R², etc., helps compare the performance of different models (e.g., LSTM vs. ARIMA vs. XGBoost).
* This can also highlight how much room for improvement exists, especially in terms of overfitting or underfitting.

**Example:**

**python**

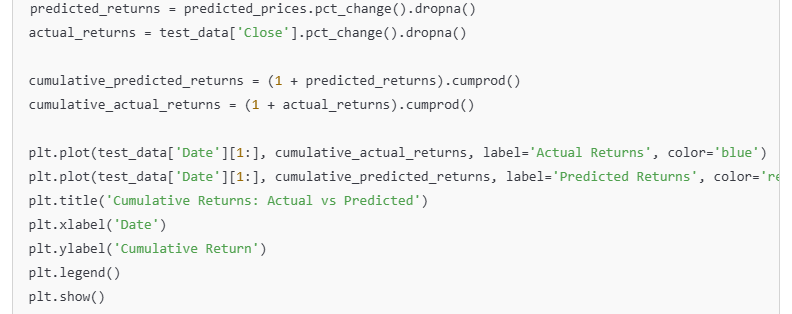


**4. Cumulative Returns from Predictions**

* Cumulative Returns Plot: By simulating a simple buy-and-hold strategy based on predicted prices, we can visualize how the model’s predictions would have influenced returns over time.
* This analysis is useful to assess whether the model can help make profitable trading decisions.

**Example:**

**python**

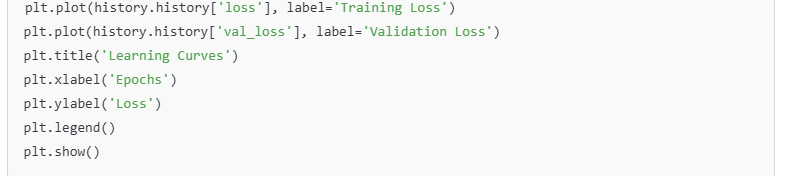


**5. Learning Curves**

* Training and Validation Loss: To assess overfitting or underfitting during training, we plotted the learning curves (training vs. validation loss over epochs).
* A well-trained model will have both training and validation loss converge towards the same value, indicating the model generalizes well.

**Example:**

**python**

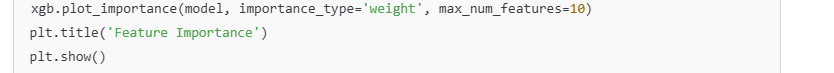


**6. Feature Importance (if applicable)**

* Feature Importance Plot: For models like XGBoost or Random Forest, a feature importance chart helps us understand which engineered features (e.g., moving averages, RSI, volume) contribute the most to predicting stock prices.
* This helps in interpreting the model’s behavior and could reveal which factors are driving predictions.

**Example:**

**python**



**7. Model Interpretability (SHAP or LIME)**

* SHAP (SHapley Additive exPlanations) or LIME can be used for model interpretability, especially in deep learning models like LSTM or complex ensemble models. These tools provide insights into how individual features contribute to each prediction.

**Model Insights**

* Overfitting/Underfitting: By analyzing the learning curves and error distributions, we can assess whether the model is overfitting to the training data or underfitting, indicating it fails to capture underlying patterns.
* Performance Gaps: Significant gaps between predicted vs. actual returns or cumulative returns may highlight the need for additional features or more advanced models.
* Feature Impact: Moving averages, volume, and RSI often play a significant role in forecasting stock prices, especially during periods of high volatility.

**10. Tools and Technologies Used**

**Programming Language**

Python – The primary language used for the entire project due to its flexibility, ease of use, and extensive support for data analysis, machine learning, and visualization.

**📈 Libraries and Frameworks**

* Pandas – Used for data manipulation and analysis, especially for time series data. It helps in cleaning, transforming, and organizing stock data efficiently.
* NumPy – Provides support for large, multi-dimensional arrays and matrices, and is essential for performing numerical operations on stock price data.
* Matplotlib & Seaborn – Essential libraries for data visualization. Matplotlib is used to generate plots, and Seaborn offers more aesthetically pleasing statistical plots for better insights into data patterns.
* Scikit-learn – Provides various machine learning models and evaluation metrics. It’s helpful for creating basic regression models, splitting data, and assessing model performance.
* TensorFlow / Keras – Powerful frameworks for building deep learning models. TensorFlow (with Keras) is used for designing, training, and evaluating AI models such as LSTM (Long Short-Term Memory) networks, which are effective for time series forecasting.
* Statsmodels – Used for implementing classical time series models such as ARIMA (AutoRegressive Integrated Moving Average), providing insights into statistical properties of stock price data.
* TA-Lib / Finta (optional) – These libraries provide a wide range of technical analysis indicators like Moving Averages, RSI, and MACD, which can be incorporated into feature engineering to improve model performance.

**🚀 Development Environment**

* Jupyter Notebook – An interactive development environment ideal for data exploration, visualization, and model testing. It supports rich outputs like charts, tables, and text.
* Google Colab – A cloud-based environment for running Python code with the added benefit of free GPU/TPU access for training deep learning models.
* VS Code / PyCharm – Integrated Development Environments (IDEs) for structured development of Python code, with support for debugging and version control.

**☁️ Deployment & Version Control (Optional)**

* Streamlit / Flask – For building interactive dashboards to visualize the model’s predictions in real-time, making it easier to deploy and showcase the results.
* Git & GitHub – Version control tools to track changes in code, collaborate with team members, and manage project updates efficiently.
* Docker – Allows containerization of the application, ensuring it runs consistently across various environments (optional, for scaling and deployment).

**11. Team Members and Contributions**

HEMAMALINI DEVEDIRAN –Team leader

HARIPRIYA JOYHILINGAM- Data collection and processing lead

HARIHARAN.C- visualization and reporting lead

IRSHAD AHMED.P- Model development and evaluation lead

GURUMOORTHY.G- Deployment