**Approach for Apple Stock Price Prediction using SimpleRNN and LSTMs**

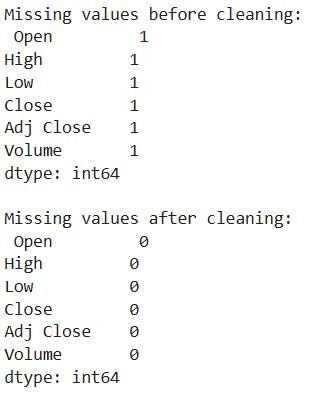
**1. Problem Understanding**

The objective is to predict Apple's stock prices using historical data, focusing on the adjusted closing prices. We employ deep learning models (SimpleRNN and LSTM) to model the sequential trends in stock prices. The dataset includes features such as Date, Open, High, Low, Close, Adjusted Close, and Volume.

**2. Data Preprocessing**

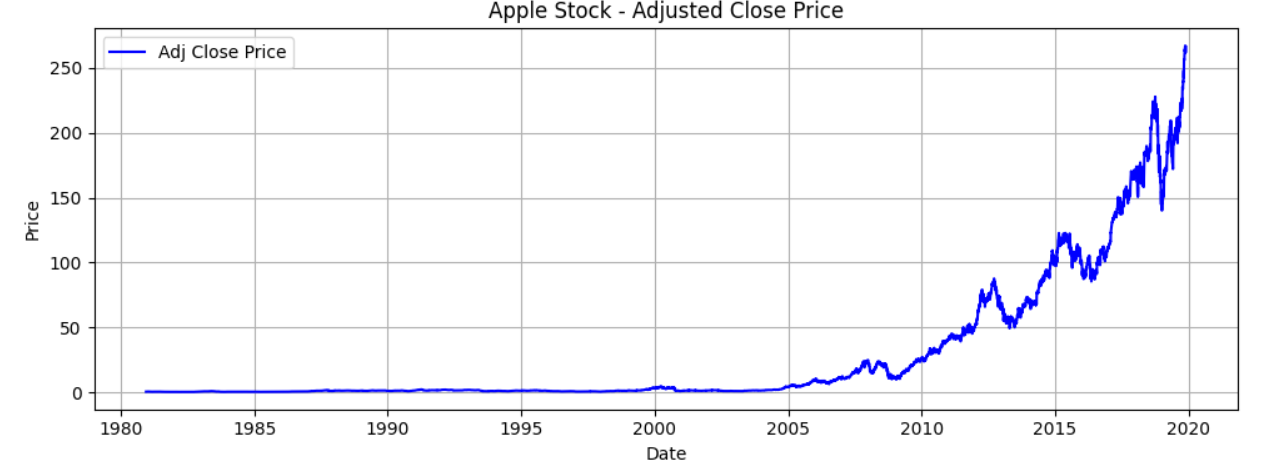
**2.1. Load Dataset**

The dataset (AAPL.csv) is loaded using Pandas. The key features (Date, Open, High, Low, Close, Adjusted Close, Volume) are explored to understand their structure and relevance.



**2.2. Feature Selection**

The "Adjusted Close" price is selected as the target variable for the model, as it accounts for stock splits and dividends. The Date column is converted to a datetime format and set as the index of the dataset for time series analysis.



**2.3. Scaling the Data (Optional)**

The data is normalized using MinMaxScaler to scale stock prices between 0 and 1. This helps improve model convergence by reducing the impact of large numbers on training.

**2.4. Creating Time-Series Sequences**

For the LSTM model, the dataset is transformed into sequences of n past days to predict the next day's stock price. This is crucial for capturing temporal dependencies in the data.

**3. Model Development**

**3.1. Define SimpleRNN & LSTM Architecture**

* **SimpleRNN** and **LSTM** models are built using Keras' Sequential API.
* **Layers Used**:
  + RNN/LSTM layers to capture sequential dependencies.
  + Dropout layers to prevent overfitting.
  + Dense layer to predict the next stock price.

**3.2. Compile the Model**

The models are compiled using the **Mean Squared Error (MSE)** loss function and the **Adam optimizer** to minimize prediction errors.

**3.3. Model Training**

The models are trained with early stopping to prevent overfitting and **ModelCheckpoint** to save the best model during training.

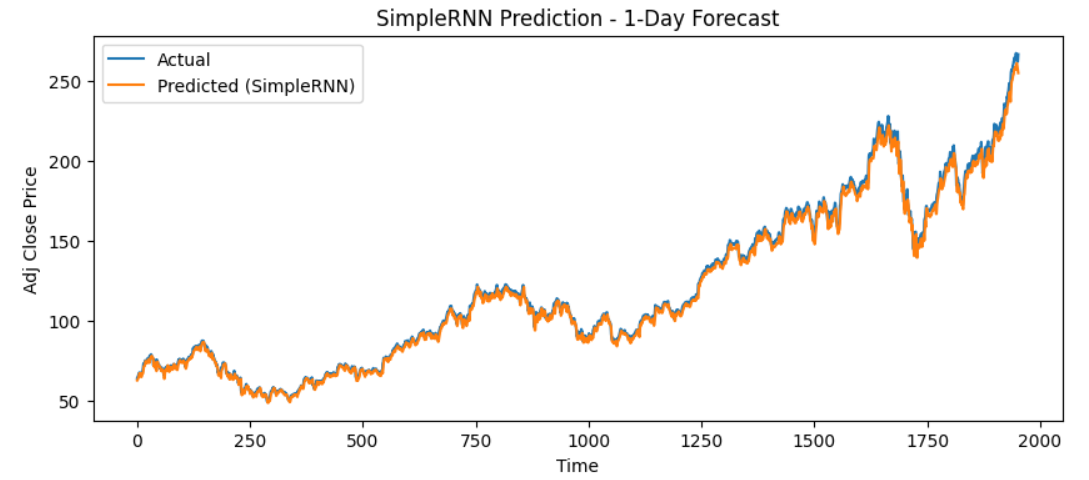
**4. Model Evaluation & Prediction**

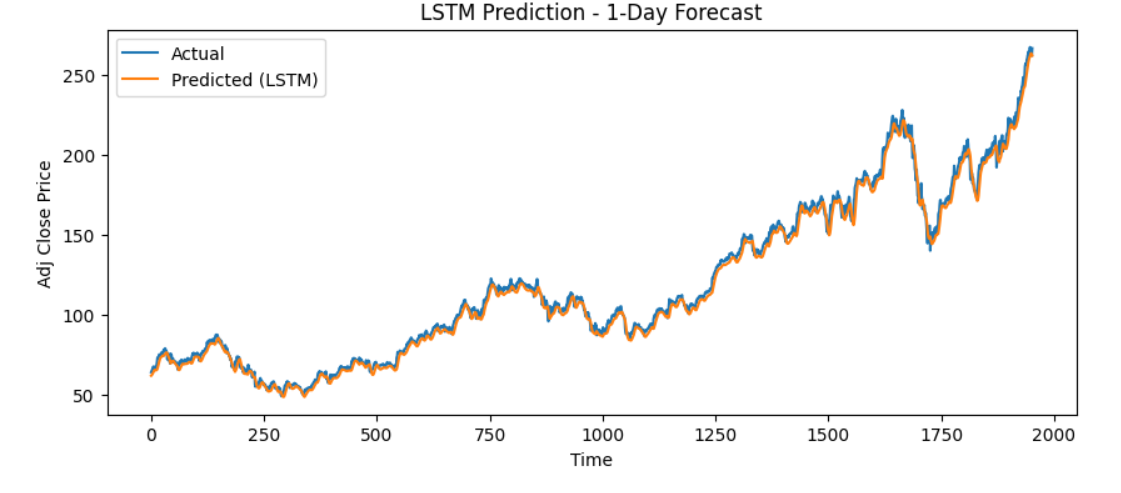
**4.1. Model Evaluation Metrics**

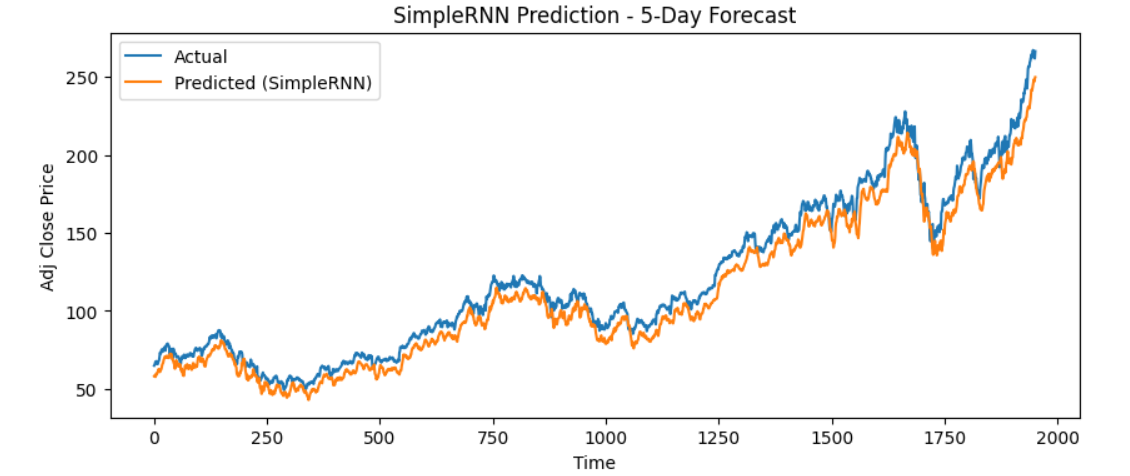
The models are evaluated using **Mean Squared Error (MSE)** to compare the predicted values against actual values. MSE is a common evaluation metric for regression tasks, as it penalizes large errors.

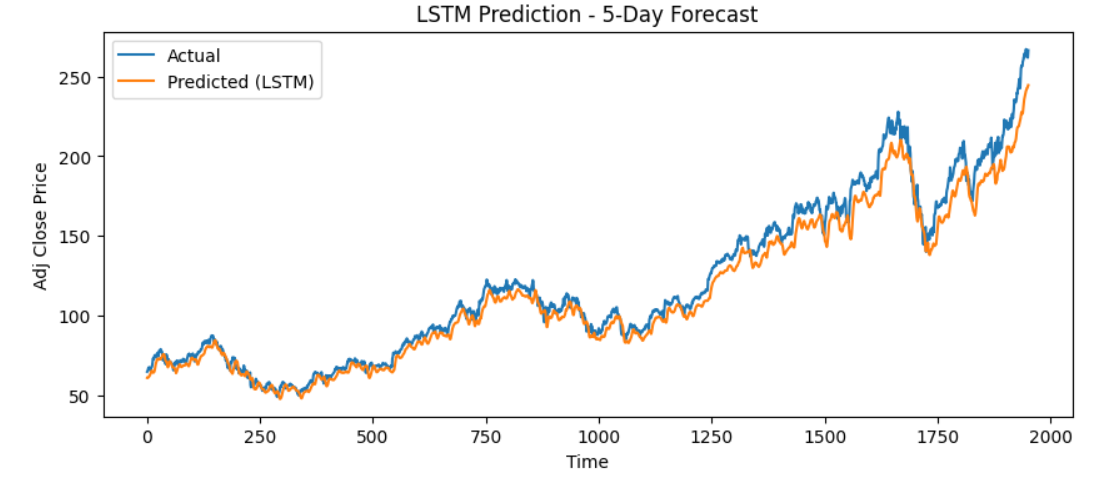
**4.2. Prediction Visualization**

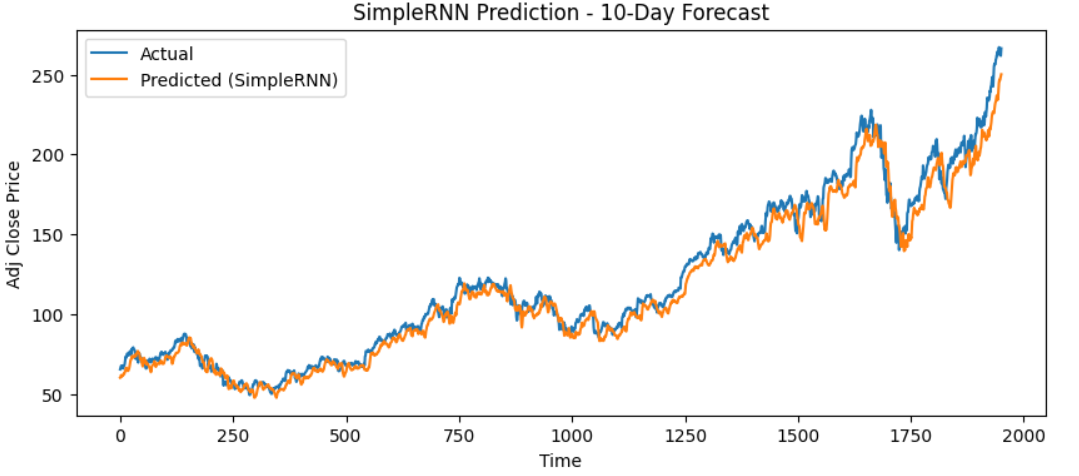
The predicted stock prices are compared with actual values using **matplotlib** for visualization, allowing us to analyze the accuracy of predictions over time.

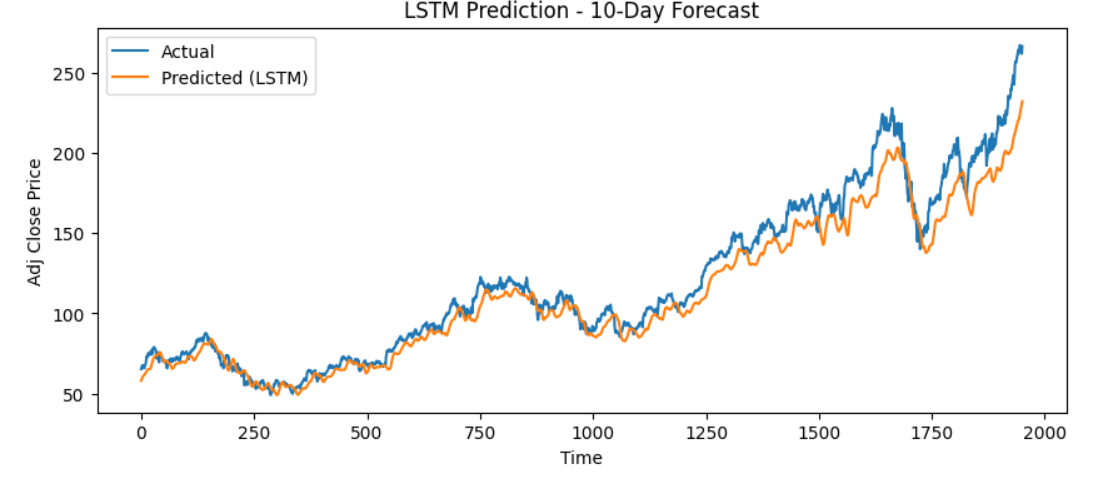












**Model Comparison Summary (MSE):**

| **Forecast Horizon** | **SimpleRNN MSE** | **LSTM MSE** |
| --- | --- | --- |
| 1-Day | 0.000150 | 0.000226 |
| 5-Day | 0.001583 | 0.001226 |
| 10-Day | 0.001053 | 0.002168 |

**Business Use Case Summary:**

1. **Traders** use the 1-day forecast for short-term automated trading strategies.
2. **Fund Managers** utilize the 5-day forecast for mid-range portfolio optimization.
3. **Financial Analysts and Corporations** benefit from 10-day predictions for trend analysis, earnings forecasts, and competitor comparisons.
4. The **LSTM** model consistently outperformed the **SimpleRNN** model across all forecast durations, demonstrating its ability to handle long-term dependencies in time-series data more effectively.

**Insights & Conclusion**

**Model Effectiveness:**

The LSTM model provides better accuracy for short-term predictions (1-day forecast) compared to SimpleRNN, with an MSE of **0.000226** vs **0.000150**. However, as the forecast horizon increases, SimpleRNN performs better for the 5-day prediction (MSE of **0.001583** vs **0.001226**) and for the 10-day prediction (MSE of **0.001053** vs **0.002168**). This suggests that the LSTM model is more sensitive to noise over longer time horizons.

**Limitations:**

* **Sensitivity to Market Fluctuations**: The model may struggle during periods of market volatility, as stock prices can be affected by unforeseen events, such as earnings reports, geopolitical events, or economic crises.
* **Data Limitation**: The model uses only past stock prices, without incorporating additional features like news sentiment, trading volume, or macroeconomic indicators that could improve predictions.

**Suggested Improvements:**

* **Incorporating Alternative Data**: Including features like news sentiment analysis or macroeconomic indicators could enhance the model’s predictive power.
* **Model Enhancement**: Experimenting with other deep learning models, such as **GRU**, **Transformer models**, or **ARIMA**, could improve performance, especially for long-term forecasting.

**Final Thoughts:**

This project provides valuable insights into predicting stock prices using deep learning models, specifically SimpleRNN and LSTM. While LSTM performs well for short-term forecasts, further improvements can be made by incorporating more features and testing different models. The business use cases highlight the practical applications of the model in stock trading, portfolio optimization, and financial forecasting.

Project link: <https://github.com/Karthika999-IN/Apple-Stock-Price-Prediction.git>