ARIMA vs LSTM for Stock Price Prediction
Week 2 — Written Report

Introduction

This report summarizes the implementation, results and lessons learned from building two forecasting models — a classical ARIMA (AutoRegressive Integrated Moving Average) and a deep-learning LSTM (Long Short-Term Memory) network — on historical stock adjusted closing prices. The objective was to compare statistical and neural approaches, assess predictive performance, and analyze practical trade-offs.

Models and key concepts

ARIMA. ARIMA(p, d, q) models time series as a linear combination of past values (AR part, order p), differencing (d) to remove non-stationarity, and past forecast errors (MA part, order q). ARIMA is interpretable and effective on stationary, linear series and short-term forecasts when assumptions hold.

LSTM. LSTM is a recurrent neural network variant designed to capture long-range dependencies through gated memory cells. It models non-linear relationships and can learn complex temporal patterns from raw sequences.

Stationarity and differencing

Stationarity (constant mean/variance and time-invariant autocovariance) is vital for ARIMA. We used the Augmented Dickey–Fuller (ADF) test to detect unit roots; when ADF p-value > 0.05, first differencing ($d = 1$) was applied and retested. Differencing removes trends and stabilizes the mean; excessive differencing risks losing useful information and inducing moving-average structure.

ACF / PACF for ARIMA parameter selection

Autocorrelation Function (ACF) and Partial ACF (PACF) plots guided the initial (p, q) choices:

- ACF with a sharp cutoff after lag q suggests MA(q).
- PACF with a sharp cutoff after lag p suggests AR(p).

We complemented this with AIC-driven grid search across small p, q ranges and optionally validated with `auto_arima`.

Sliding window and normalization for LSTM

LSTMs require supervised training data; we converted the series into input-output pairs using a sliding window of past w timesteps to predict the next value. Typical w used was 60 days. Feature scaling is critical: we normalized adjusted closing prices to $[0, 1]$ using MinMaxScaler during training and inverted the scale to obtain predictions in original units. Normalization speeds convergence and prevents gradient issues.

Implementation highlights and challenges

Data preprocessing. Daily stock series were resampled to business days and forward-filled to handle missing trading days. Train/test split used an 80/20 chronological split to avoid lookahead bias.

ARIMA-specific challenges. Financial series are often non-stationary and heteroskedastic; selecting (p, d, q) by visual ACF/PACF rules can be ambiguous, so the AIC grid search was essential. Some parameter combinations failed to converge; robust code skipped these. ARIMA residual diagnostics (Ljung–Box, residual ACF, QQ plot) were used to validate the white-noise assumption.

LSTM-specific challenges. LSTMs are data-hungry and computationally heavier. Hyperparameters (layers, units, learning rate, batch size, epochs) required tuning; overfitting was mitigated using dropout and validation split. Training time and sensitivity to scaling were notable practical issues.

Observations and model comparison

Predictive behavior. ARIMA yielded competitive short-term forecasts during relatively stable periods because it captures linear autocorrelations. LSTM performed better when the series exhibited non-linear patterns or regime shifts, provided sufficient training data and careful tuning.

Metrics. We evaluated MAE and RMSE for both models (and MAPE when desired). Results often showed:

- ARIMA: lower variance in short-horizon errors, but systematic under/over-shoots during volatility spikes.
- LSTM: improved fit in non-linear intervals, but higher variance and occasional overfitting if regularization was insufficient.

Visualizations and what they reveal

(Include the notebook-generated figures when compiling the report.)

- **Price vs. time:** reveals trend, volatility, and whether seasonal patterns exist. A visible trend necessitated differencing for ARIMA.
- **ARIMA vs LSTM forecast comparison:** overlaying actuals and predictions highlights where each model tracks or diverges. Confidence intervals from ARIMA indicate model uncertainty; LSTM lacks simple closed-form CIs.
- **Residual plots:** for ARIMA, residuals should look like white noise — no serial correlation and near-zero mean. Persistent autocorrelation suggests model misspecification. LSTM residuals often require more careful analysis and can show heteroskedastic patterns.
- **LSTM learning curves:** plots of training vs validation loss identify overfitting (large gap) or underfitting (high loss on both). They guided adjustments to epochs, architecture and regularization.

Conclusions and future work

Both approaches have strengths: ARIMA is interpretable, fast, and reliable for linear stationary components; LSTM captures non-linearities and long-term dependencies but needs more data, compute and careful tuning. Key takeaways:

- Begin with diagnostics (ADF, ACF/PACF) and an ARIMA baseline. Use AIC/grid search to select (p, d, q).
- Use sliding windows and normalization for LSTM; monitor learning curves and apply regularization.
- Compare models by the same metrics and use walk-forward validation for robust assessment.

Possible improvements: include seasonal models (SARIMA/SARIMAX), exogenous features (volume, technical indicators), ensemble or hybrid models combining ARIMA and LSTM, hyperparameter tuning (Bayesian search), and probabilistic deep learning methods to quantify LSTM uncertainty.