ASSIGNMENT 2 REPORT

Sam Navdeep K
24b1303

1 ARIMA and LSTM models

1.1 ARIMA Model

The AutoRegressive Integrated Moving Average (ARIMA) model is a widely used statistical method for time series forecasting.
The ARIMA model is composed of two fundamental stochastic models: the AutoRegressive (AR) model and the Moving Average (MA) model, combined with differencing to handle non-stationarity.

The AutoRegressive model of order p, denoted as AR(p), is defined as

$$y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \epsilon_t \tag{1}$$

where y_t is the current value of the time series, ϕ_i are the autoregressive coefficients, and ϵ_t is an error term at time t.

The Moving Average model of order q, denoted as MA(q), is given by

$$y_t = c + \epsilon_t + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} \tag{2}$$

where θ_j are the moving average coefficients and ϵ_t represents errors at time t.

1.2 LSTM model

LSTM (Long Short-Term Memory) is a special type of Recurrent Neural Network (RNN) designed to learn long-term time series.

An LSTM cell controls the flow of information using four gates: the forget gate, input gate, cell candidate (update) gate, and output gate. Each gate uses a sigmoid or hyperbolic tangent activation function to regulate information.

Forget Gate: The forget gate determines how much information from the previous cell state should be retained or discarded.

$$f_t = \sigma\left(W_f[h_{t-1}, x_t] + b_f\right) \tag{3}$$

Here, x_t is the input at time t, h_{t-1} is the previous hidden state, W_f and b_f are the learnable weight matrix and bias of the forget gate, and $\sigma(\cdot)$ is the sigmoid activation function.

Input Gate: The input gate controls how much new information is allowed to update the cell state.

$$i_t = \sigma\left(W_i[h_{t-1}, x_t] + b_i\right) \tag{4}$$

In this equation, W_i and b_i are the input gate parameters, and the sigmoid function scales the new information between 0 and 1.

Cell Candidate (Update Gate): This gate generates new candidate values that may be added to the cell state.

$$\tilde{c}_t = \tanh\left(W_c[h_{t-1}, x_t] + b_c\right) \tag{5}$$

The hyperbolic tangent function produces values in the range $[-1, 1]$, and W_c, b_c are the learnable parameters for updating the cell state.

Output Gate: The output gate determines which parts of the cell state are exposed as the hidden state.

$$o_t = \sigma\left(W_o[h_{t-1}, x_t] + b_o\right) \tag{6}$$

$$h_t = o_t \odot \tanh(c_t) \tag{7}$$

Here, c_t is the updated cell state, h_t is the hidden state output, W_o and b_o are the output gate parameters, and $\odot$ denotes element-wise multiplication.

1.3 Important Concepts in Time Series Forecasting

Stationarity and Differencing: A time series is said to be stationary if its statistical properties such as mean, variance, and autocorrelation remain constant over time. Most statistical forecasting models, including ARIMA, require the input series to be stationary. Real-world data such as stock prices are often non-stationary due to trends and seasonality. To achieve stationarity, differencing is applied, where the current value is subtracted from the previous value, as shown in equation below. Differencing removes trends and stabilizes the mean of the series.

$$y'_t = y_t - y_{t-1} \tag{8}$$

ACF and PACF Plots for ARIMA Parameter Selection: The AutoCorrelation Function (ACF) and Partial AutoCorrelation Function (PACF) plots are used to identify suitable parameters for the ARIMA model. The ACF plot measures the correlation between the current value and its lagged values and is primarily used to determine the moving average order q. The PACF plot measures the correlation between the current value and a lag after removing the effects of intermediate lags and is mainly used to identify the autoregressive order p. Sharp cutoffs in these plots provide guidance for parameter selection.

Sliding Window Technique for LSTM: LSTM networks require time series data to be converted into a supervised learning format. This is achieved using the sliding window technique, where a fixed number of past observations are used as input to predict a future value. By moving the window forward in time, multiple input–output pairs are generated.

This technique enables the LSTM to learn temporal dependencies and patterns across consecutive time steps.

Data Normalization and Its Importance in Neural Networks: Data normalization is a crucial preprocessing step for neural networks, as large variations in input scale can lead to slow convergence and vanishing gradients. Normalization rescales data into a fixed range, commonly using Min–Max normalization as shown in Equation below. Neural networks, including LSTMs, perform more efficiently when inputs are normalized, as it stabilizes training and improves learning. It is important that normalization parameters are learned only from training data and then applied to validation and test data to avoid data leakage.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{9}$$

2 Results and plots

2.1 ARIMA output

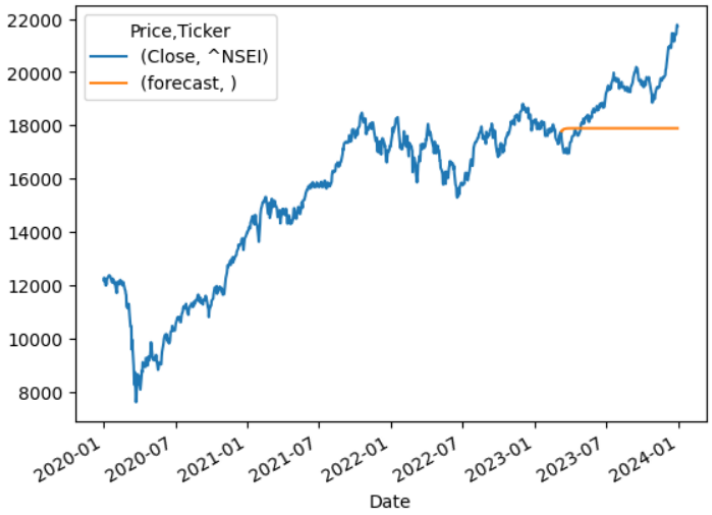

Figure 1: nifty50 stock price predictions for 4 years, and prediction for nearly an year by ARIMA model

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, root_mean_squared_error
mse = mean_squared_error(np.exp(x_test), nifty50['forecast'][len(x_train):])
mae = mean_absolute_error(np.exp(x_test), nifty50['forecast'][len(x_train):])
rmse = root_mean_squared_error(np.exp(x_test), nifty50['forecast'][len(x_train):])
print(f'MSE: {mse}, MAE: {mae}, RMSE: {rmse}')
[26]  ✓ 0.0s
...  MSE: 2741540.0378070283, MAE: 1411.3619870134582, RMSE: 1655.7596558096916
```

Figure 2: ARIMA model eval metrics

2.2 LSTM outputs

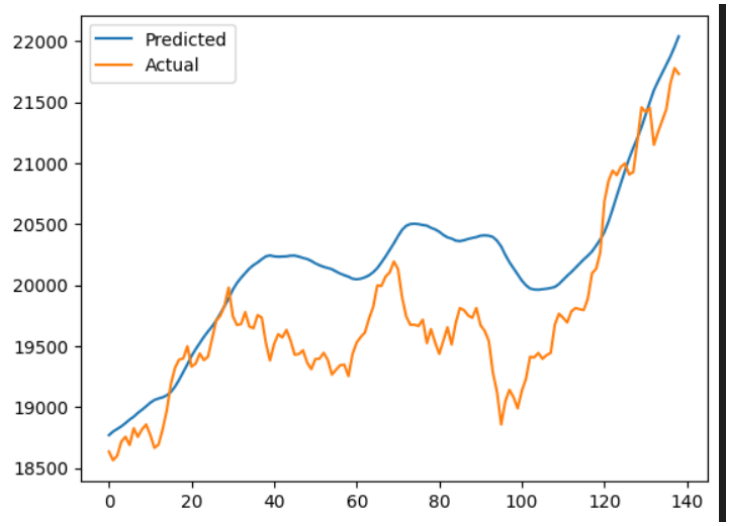

Figure 3: LSTM model prediction for nifty50 stock prices for 6 months

```
#### MODEL EVAL ######
from sklearn.metrics import mean_squared_error, mean_absolute_error
mse = mean_squared_error(np.exp(y_test_seq), np.exp(pred_price))
mae = mean_absolute_error(np.exp(y_test_seq), np.exp(pred_price))
0.0s

from sklearn.metrics import root_mean_squared_error
rmse = root_mean_squared_error(np.exp(y_test_seq), np.exp(pred_price))
print(f"MSE: {mse}, MAE: {mae}, RMSE: {rmse}")
0.0s
MSE: 320650.0, MAE: 468.9287414550781, RMSE: 566.2596435546875
```

Figure 4: LSTM model eval metrics

Note: the parameters chosen for ARIMA model was (5,1,0) by running a function which determines the best value for the parameters, it could be also gathered from the PACF and ACF plots.

3 Conclusion

The performance of the ARIMA and LSTM models was evaluated using Mean Squared Error (MSE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE). The ARIMA model produced an MSE of 2,741,540, an MAE of 1411, and an RMSE of 1655. These relatively high error values indicate that ARIMA was unable to accurately capture the complex dynamics of the time series, primarily due to its reliance on linear assumptions and strict stationarity requirements.

In contrast, the LSTM model achieved significantly lower error values, with an MSE of 320,650, an MAE of 469, and an RMSE of 566. The substantial reduction in all evaluation metrics demonstrates that the LSTM model provides more accurate and stable

predictions. This improved performance can be attributed to LSTM's ability to model non-linear relationships, learn long-term dependencies, and operate effectively without requiring the data to be strictly stationary.

Overall, the results clearly indicate that the LSTM model outperforms the ARIMA model for the given dataset, making it a more suitable and robust approach for stock market time series forecasting.

.

4 Files

The .ipynb files are in the google drive link -
`https://drive.google.com/drive/folders/1WCDlVeha2dPlbdbOuhUTYbZPzRptsVHr?usp=sharing`