Time-Series Forecasting of Stock Prices

The primary objective of this project is to develop two distinct time-series forecasting models: ARIMA and LSTM, to predict future stock prices. The project will compare their predictive accuracy using RMSE and MAPE to determine which model generalizes better and provides more reliable forecasts for financial data.

About the Dataset:

The analysis was performed on the historical stock price data for Apex Footwear Ltd., loaded from the **APPEX_FOOT_data.csv** file. The dataset contains daily stock market information, including Open, High, Low, Close, and Volume.

```
dataset.head()
```

	Date	Open	High	Low	Close	Volume
0	10/1/2012	180.165289	184.100748	175.521448	181.503345	22000
1	10/2/2012	184.179457	186.462023	179.535616	185.202676	54400
2	10/3/2012	186.540732	199.842582	186.540732	196.615506	101200
3	10/4/2012	193.624557	197.953562	189.138135	193.860685	13200
4	10/7/2012	192.837466	192.837466	182.054309	183.943329	10000

Next steps: Generate code with dataset   New interactive sheet

Fig: APPEX_FOOT_data.cv overview

For the purpose of this forecasting task, the analysis was focused exclusively on the daily **'Close'** price, as it is the most common benchmark for a stock's value on any given day. All other columns were removed to simplify the models into a univariate time series problem.

Data Preprocessing:

The raw data was first split into a training set and a testing set, with 80% of the data used for training the models and the remaining 20% reserved for testing and evaluation. Distinct preprocessing steps were then applied for each model to meet their specific input requirements.

LSTM Preprocessing: A more involved, two-step preprocessing pipeline was required for the LSTM model:

1. **Data Scaling:** All closing prices in the dataset were scaled to a range between 0 and 1 using MinMaxScaler. Neural networks perform significantly better with normalized data, as it helps the learning algorithm converge more efficiently.
2. **Sequence Creation:** The time-series data was transformed into input-output pairs. A "lookback" period of 100 days was defined, meaning the model was trained to predict the price of the 101st day using the prices of the preceding 100 days as input. It looks like a DataLoader. Previous 100 days will be input, based on 100 days predicting a price for the next day. Its looks like that:

F1	**F2**	**F3**	**………..**	**F99**	**F100**	**Output**
14	13	7		11	10	11.5
13	7	12		10	11.5	12

ARIMA Preprocessing: The **pmdarima.auto_arima** function was utilized, which automatically handles a key requirement for ARIMA models: **stationarity.** The function internally performs differencing on the time series to make its mean and variance constant, a necessary step for the model to work correctly.

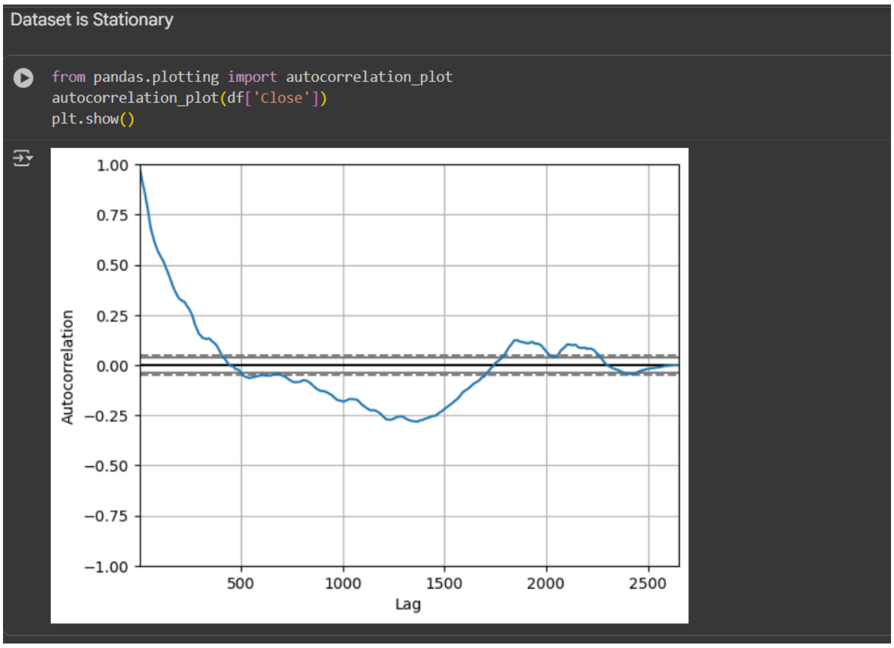

Fig: Behavior of dataset stationarity

Models and Training:

1. **Model 2: LSTM (Long Short-Term Memory):**

This is a deep, stacked Bidirectional LSTM (Bi-LSTM) network designed to capture complex patterns in time-series data.

In short, the model processes an input sequence of 100 time steps through three successive Bi-LSTM layers (with 128, 64, and 32 units). These layers read the sequence both forwards and backward, allowing them to learn from past and future context simultaneously. To prevent overfitting, two Dropout layers are included, each deactivating 30% of neurons during training. Finally, the processed information is passed through standard dense layers to produce a single numerical prediction.

```
model.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
bidirectional_6 (Bidirectional)	(None, 100, 256)	133,120
dropout_4 (Dropout)	(None, 100, 256)	0
bidirectional_7 (Bidirectional)	(None, 100, 128)	164,352
dropout_5 (Dropout)	(None, 100, 128)	0
bidirectional_8 (Bidirectional)	(None, 64)	41,216
dense_4 (Dense)	(None, 64)	4,160
dense_5 (Dense)	(None, 1)	65

Total params: 342,913 (1.31 MB)
Trainable params: 342,913 (1.31 MB)
Non-trainable params: 0 (0.00 B)

Fig: LSTM Model summary

2. **Model 1: ARIMA (Autoregressive Integrated Moving Average):**

The provided code implements an ARIMA (Autoregressive Integrated Moving Average) model, a classical statistical method for time-series forecasting. Specifically, an ARIMA(5, 1, 1) model is being trained on the historical closing prices. This configuration means the model predicts the next day's price based on a linear combination of two key components: the Autoregressive (AR) term, where p=5, indicates that the model uses the price data from the previous five days to make a prediction; and the Integrated (I) term, where d=1, means the raw price data has been differenced once to stabilize it, effectively making the model work on the daily price changes rather than the absolute price level. The Moving Average (MA) term is q=1, indicating that the model does not use past forecast errors in its calculations.

Accuracy & Validation:

To evaluate the models, two standard regression metrics were calculated on the test set: Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE).

Model Performance Comparison (LSTM vs ARIMA):

	Metric	LSTM Train	LSTM Test	ARIMA Test
0	RMSE	5.478190	6.924076	25.010571
1	MAPE	0.014353	0.016766	0.074163

The results clearly show that the **LSTM** model significantly outperformed the **ARIMA** model on both metrics.

Performance Comparison & Recommendation:

Based on both quantitative metrics and qualitative visual inspection, the **LSTM** model is overwhelmingly better for this forecasting task.

1. **Superior Accuracy:** The LSTM's RMSE (6.92) is nearly four times lower than ARIMA's (25.01), and its MAPE (1.67%) is also substantially better than ARIMA's (7.42%). This means the LSTM's predictions were, on average, much closer to the true stock prices.
2. **Ability to Capture Non-Linearity:** Stock market data is inherently complex, noisy, and non-linear. ARIMA, being a linear model, is fundamentally limited in its ability to capture these intricate patterns. The LSTM, as a deep learning model, excels at learning complex, non-linear relationships and long-term dependencies from the data, which is essential for modeling financial markets.
3. **Better Generalization:** The superior performance on the unseen test data shows that the LSTM model generalized far better from the training data. The visualization confirms that the LSTM's predictions are more dynamic and responsive to price fluctuations, unlike the more rigid forecast produced by the ARIMA model.

Prediction Results & Visualization:

The final 30-day forecasts from both models were plotted against the actual historical data.

1. **LSTM Forecast Plot:** The LSTM model's forecast visually tracks the actual price movements in the test set much more closely. It successfully captures the underlying trend and some of the volatility, demonstrating a more nuanced understanding of the data's patterns. Here is a plot of 30 days stock “Close” price based on the last 100 days of dataset stock “Close” price.

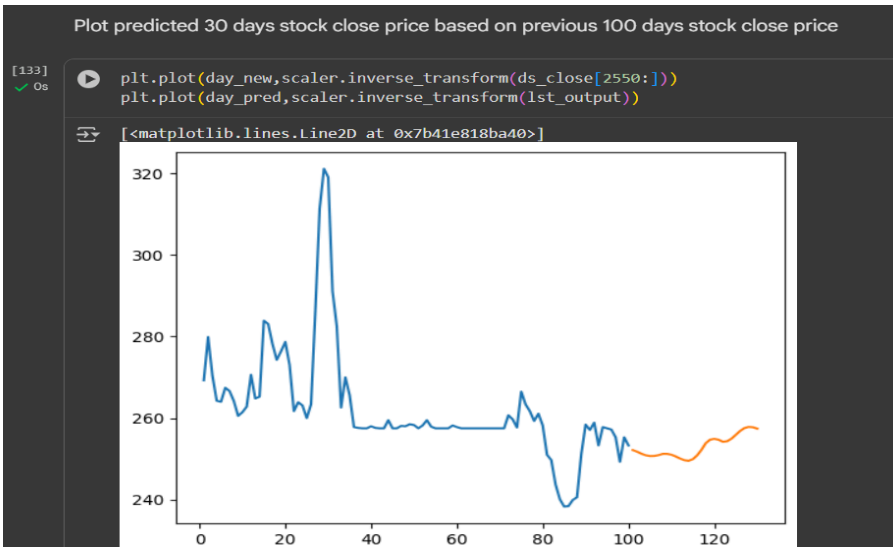

Fig: LSTM model predict next 30 days stock “Close” price base on last 100 Stock Close price from dataset

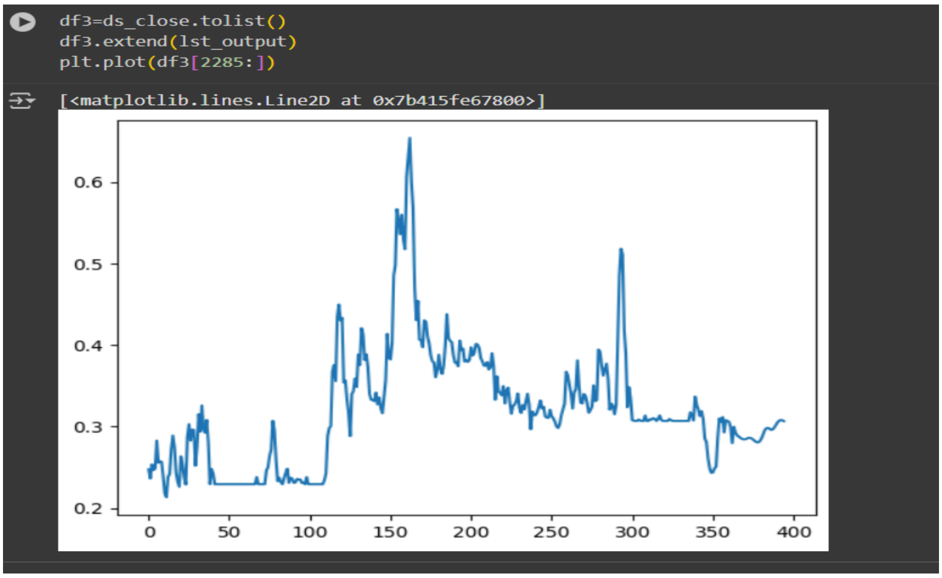

Fig: How 30 days prediction looks like if adjust with last 1 year stock Close price

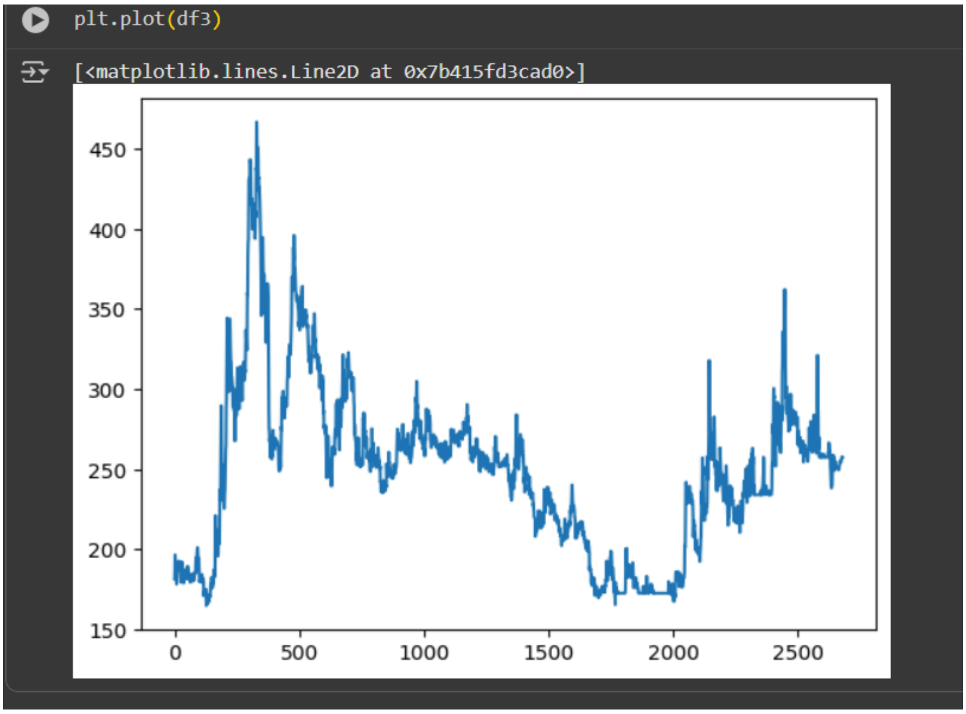

Fig: How 30 days prediction looks like if adjust with last full dataset stock Close price

2. **ARIMA Forecast Plot:** The graph shows the ARIMA model's prediction as a relatively straight line, with a widening pink confidence interval. This indicates that as the forecast extends further into the future, its uncertainty increases dramatically. The forecast does not capture the volatility present in the test data.

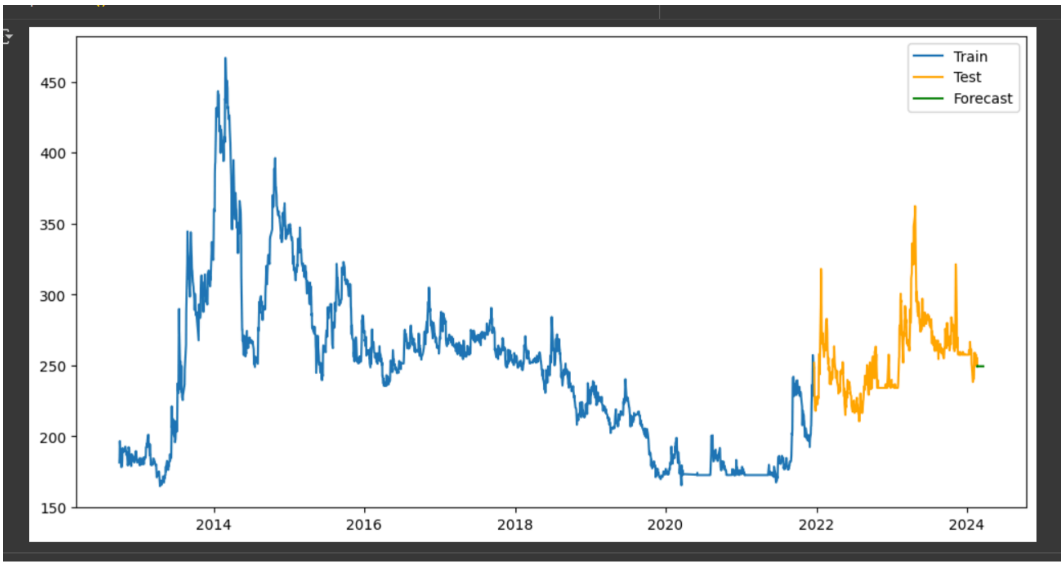

Fig: ARIMA model predict next 30 days stock “Close” price

In conclusion, this analysis definitively shows that the LSTM model provides a significantly more accurate and reliable forecast for stock prices compared to the traditional ARIMA model. The LSTM's superior performance, evidenced by its much lower error metrics, stems from its ability to capture the complex, non-linear patterns inherent in volatile financial data. While ARIMA offers a basic statistical benchmark, the LSTM's deep learning architecture is better suited for generalizing from historical data and is the recommended approach for this forecasting task.