# 📘 Advanced LSTM Section – Standalone Documentation

Here’s an **enhanced explanation** with **diagrams and intuitive analogies** added to help you understand exactly what’s going on inside the LSTM model. This standalone document is designed for your personal learning clarity.

## 🔁 Why Sequence Data for LSTM?

LSTM (Long Short-Term Memory) is a type of neural network that excels at learning from sequential, time-ordered data — perfect for stock market predictions.

### ✅ LSTM Input Format

LSTM expects input in 3D shape:

[samples, time\_steps, features]

* samples: Total number of training examples.
* time\_steps: How many days we look back in each sequence (e.g., 5).
* features: Number of lag inputs per day (e.g., 3: Lag\_1, Lag\_2, Lag\_3).

### 🔍 Visualization: Input Example

Time Steps → (Past 5 Days)  
 ┌─────┬─────┬─────┬─────┬─────────┐  
Features│ Day1 │ Day2 │ Day3 │ Day4 │ Day5│  
───────────────────────────────────────────  
Lag\_1 │ 0.01 │ 0.03 │ 0.02 │ 0.00 │0.01 │  
Lag\_2 │-0.01 │ 0.02 │ 0.01 │ 0.03 │-0.02│  
Lag\_3 │ 0.04 │-0.02 │ 0.05 │ 0.01 │ 0.00│  
 └──────┴──────┴──────┴──────┴─────┘

🧠 Think of it like trying to predict tomorrow’s return by analyzing patterns over the last 5 days.

### 🧠 Analogy: Trader with Memory

Imagine a trader who remembers how prices moved in the last few days. The LSTM acts like this trader, learning:

* Short-term memory (momentum or reversal patterns)
* Sequence behavior instead of isolated values

Unlike traditional models (e.g., XGBoost), LSTM learns from **stories** in the data.

## 🧪 Why MinMaxScaler (Scaling to 0–1)?

Neural networks require small, uniform data ranges to train properly.

### ⚠️ Without Scaling:

* Inputs like 0.1, 0.0002, -0.04, 0.5 confuse gradient calculations
* Training becomes unstable

### ✅ With MinMaxScaler:

| Original | Scaled |
| --- | --- |
| -0.02 | 0.0 |
| 0.01 | 0.5 |
| 0.04 | 1.0 |

Keeps gradients stable and learning efficient

### ❓ Why scale y too?

* LSTM calculates gradients using both inputs and outputs
* If y isn’t scaled, gradients can misbehave
* We inverse-scale predictions after inference

## 🪮 LSTM Model Flow

┌────────────────────┐  
 │ Past 5 Days │ ← (Lag\_1, Lag\_2, Lag\_3)  
 └────────────────────┘  
 ↓  
 ┌────────────────────┐  
 │ LSTM Layer (64 units)│ ← Learns time-dependent patterns  
 └────────────────────┘  
 ↓  
 ┌────────────────────┐  
 │ Dropout Layer (0.2)| ← Prevents overfitting  
 └────────────────────┘  
 ↓  
 ┌────────────────────┐  
 │ Dense Layer (1 unit)│ ← Predicts next return  
 └────────────────────┘  
 ↓  
 ┌───────────────────┐  
 │Inverse Scaling │ ← Converts prediction to actual return  
 └───────────────────┘

## 📀 Why Save with joblib

We save the following to ensure the model can predict correctly in the future:

* scaler\_X.pkl: How inputs were scaled
* scaler\_y.pkl: How outputs were scaled
* lstm\_model.h5: The trained LSTM weights

### Example:

joblib.dump(scaler\_X, 'lstm\_scaler\_X.pkl')  
joblib.dump(scaler\_y, 'lstm\_scaler\_y.pkl')  
model.save('lstm\_model.h5')

To reuse later:

scaler\_X = joblib.load('lstm\_scaler\_X.pkl')  
X\_new\_scaled = scaler\_X.transform(X\_new)  
y\_pred\_scaled = model.predict(X\_new\_scaled)  
y\_pred = scaler\_y.inverse\_transform(y\_pred\_scaled)

## 🔄 Summary Table

| Step | Meaning |
| --- | --- |
| **3D Input Shape** | [samples, time\_steps, features] — required for LSTM |
| **Rolling Window** | Past 5 days of Lag values to predict next day |
| **MinMax Scaling** | Ensures stability during training |
| **LSTM Pipeline** | LSTM → Dropout → Dense → Inverse Scaler |
| **Save with joblib** | Guarantees reproducibility and correct inference later |

## 🚀 Final Takeaways

* LSTM is powerful for sequential stock data
* Scaling and sequence formatting are essential
* Model learns from short-term memory patterns
* Great for trends, not for sharp market shocks