**How Your Keras Model Predicts Stock Prices**

Your **stock price predictor** is a **deep learning model using Long Short-Term Memory (LSTM) networks**, which are well-suited for time-series forecasting. Below is a detailed explanation of how it works.

**🔍 Model Architecture Breakdown**

Your model consists of **four LSTM layers**, each followed by a **Dropout layer**, and finally a **Dense (fully connected) output layer**.

| **Layer Name** | **Type** | **Output Shape** | **Parameters** |
| --- | --- | --- | --- |
| **LSTM 1** | LSTM | (None, 100, 50) | 10,400 |
| **Dropout 1** | Dropout | (None, 100, 50) | 0 |
| **LSTM 2** | LSTM | (None, 100, 60) | 26,640 |
| **Dropout 2** | Dropout | (None, 100, 60) | 0 |
| **LSTM 3** | LSTM | (None, 100, 80) | 45,120 |
| **Dropout 3** | Dropout | (None, 100, 80) | 0 |
| **LSTM 4** | LSTM | (None, 120) | 96,480 |
| **Dropout 4** | Dropout | (None, 120) | 0 |
| **Dense** | Fully Connected | (None, 1) | 121 |

**📌 What Each Layer Does**

1. **LSTM Layers**:
   * These **capture long-term dependencies** in the stock data.
   * Each LSTM layer has a different number of units (50 → 60 → 80 → 120), increasing complexity.
2. **Dropout Layers**:
   * Dropout helps prevent **overfitting** by randomly disabling some neurons during training.
   * Each LSTM layer is followed by a dropout layer (probably with **20% or 30% dropout rate**).
3. **Dense Layer (Final Output)**:
   * A **single neuron with linear activation** to predict the stock price.
   * Takes the output from the final LSTM layer and maps it to **one predicted price value**.

**🧠 How the Model Makes Predictions**

1. **Takes the last 100 days of stock prices** (input shape: 100 x 1).
2. **Scales the data** between 0 and 1 using **MinMaxScaler**.
3. **Passes the scaled data through the LSTM layers**:
   * The model learns **patterns and trends** in the stock's movement.
4. **Outputs a single predicted price** for the next time step.
5. **Rescales the prediction** back to the original stock price range.

**⚡️ How Your App Uses This Model**

Your **Streamlit app** does the following:  
✔ **Fetches stock data** from Yahoo Finance.  
✔ **Splits it into training (80%) and testing (20%)**.  
✔ **Uses MinMaxScaler to normalize** the data before feeding it into the model.  
✔ **Prepares input data** by selecting the last **100 days of stock prices**.  
✔ **Predicts future stock prices** using your LSTM model.  
✔ **Displays actual vs. predicted prices** on a graph.

**📈 How to Improve the Model?**

1. **Use More Features:** Instead of just Close price, try including:
   * Open, High, Low, Volume, and technical indicators like Moving Averages.
2. **Try Different Sequence Lengths:**
   * Right now, you're using **100 previous days**. Experiment with **30, 60, 120** days.
3. **Hyperparameter Tuning:**
   * Adjust the **number of LSTM layers**, dropout rates, and optimizer settings.
4. **Use Bidirectional LSTM:**
   * It allows the model to capture **future trends** better.

**🔥 Final Thoughts**

✅ **Your model is a strong LSTM-based time series predictor.**  
✅ **It learns patterns from historical prices and makes stock price predictions.**  
✅ **To improve accuracy, try incorporating more features and tuning hyperparameters.**

Would you like help implementing any improvements? 🚀