```
# Phase 1: Stock Price Prediction using LSTM and GRU

# --- 0. Import Required Libraries ---
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, GRU, Dense

# --- 1. Load Dataset ---
# If using Google Colab, upload the file via the Files tab
df = pd.read_csv("/stock_data.csv", parse_dates=['trading_date'], index_col='trading_date')
print("Dataset Preview:")
print(df.head())

# --- 2. Data Preprocessing ---
def preprocess_data(df):
    if 'closing_price' not in df.columns:
        raise ValueError("Dataset must contain 'closing_price' column")

    df = df.dropna()
    scaler = MinMaxScaler()
    df['scaled_price'] = scaler.fit_transform(df[['closing_price']])
    return df, scaler

# Create sequential data for time series
def create_sequences(data, seq_len):
    X, y = [], []
    for i in range(len(data) - seq_len):
        X.append(data[i:i+seq_len])
        y.append(data[i+seq_len])
    return np.array(X), np.array(y)

# --- 3. Build Model (LSTM or GRU) ---
def build_model(model_type='LSTM', seq_len=30, features=1):
    model = Sequential()
    if model_type == 'LSTM':
        model.add(LSTM(64, return_sequences=True, input_shape=(seq_len, features)))
        model.add(LSTM(32))
    elif model_type == 'GRU':
        model.add(GRU(64, return_sequences=True, input_shape=(seq_len, features)))
        model.add(GRU(32))
    else:
        raise ValueError("Invalid model_type. Choose 'LSTM' or 'GRU'.")

    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mean_squared_error')
    model.summary()
    return model

# --- 4. Evaluation ---
def evaluate(y_true, y_pred):
    mse = mean_squared_error(y_true, y_pred)
    mae = mean_absolute_error(y_true, y_pred)
    print(f"MSE: {mse:.4f}, MAE: {mae:.4f}")
    return mse, mae

# --- 5. Plot Results ---
def plot_results(actual, predicted, dates):
    plt.figure(figsize=(12,6))
    plt.plot(dates, actual, label='Actual Price', color='blue')
    plt.plot(dates, predicted, label='Predicted Price', color='red')
    plt.title("Stock Price Prediction")
    plt.xlabel("Date")
    plt.ylabel("Price")
    plt.legend()
    plt.grid(True)
    plt.show()

# --- 6. Main Execution ---
sequence_length = 30
df, scaler = preprocess_data(df)
data = df['scaled_price'].values
X, y = create_sequences(data, sequence_length)

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)
```

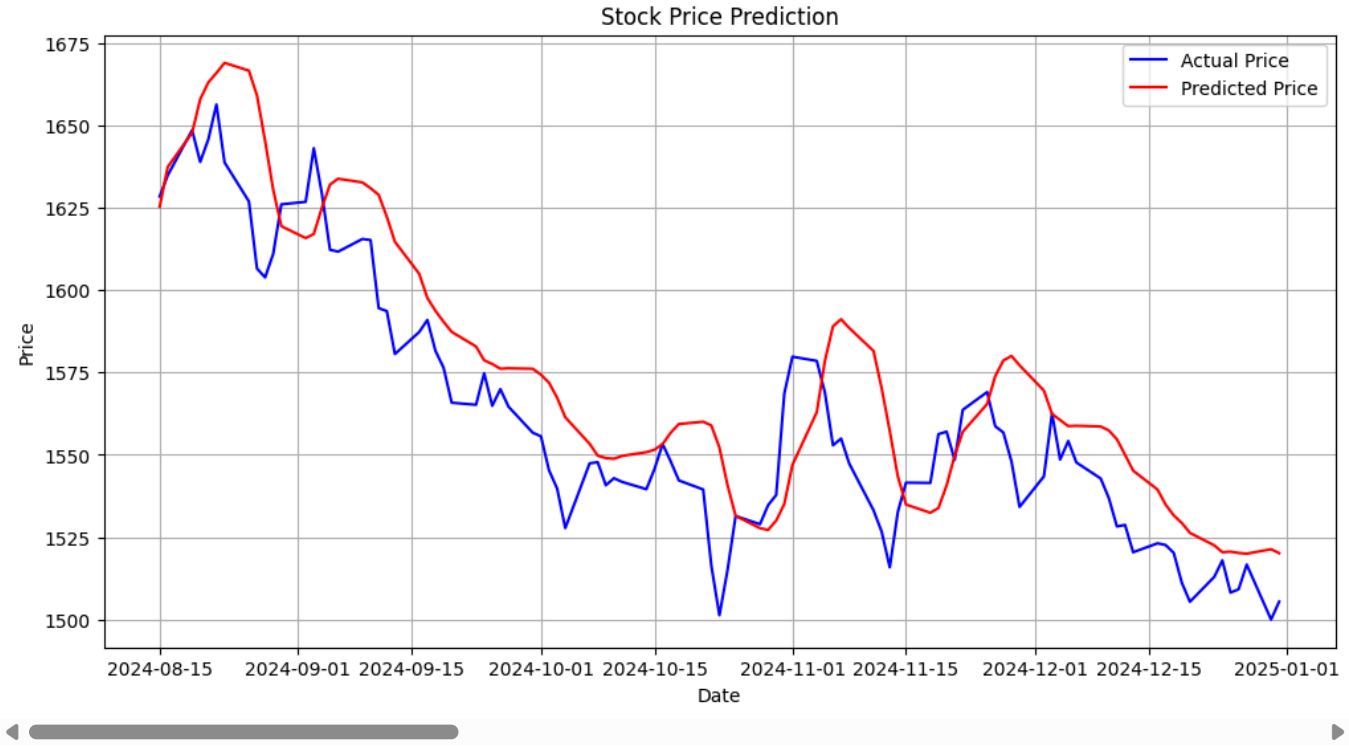
```
# Reshape for LSTM/GRU input
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))

# Choose model type: 'LSTM' or 'GRU'
model = build_model(model_type='LSTM', seq_len=sequence_length)

# Train model
history = model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.1, verbose=1)

# Predict
predictions = model.predict(X_test)
predicted_prices = scaler.inverse_transform(predictions)
actual_prices = scaler.inverse_transform(y_test.reshape(-1, 1))

# Evaluation and Visualization
evaluate(actual_prices, predicted_prices)
plot_results(actual_prices, predicted_prices, df.index[-len(predicted_prices):])
```

```
Dataset Preview:
              closing_price symbol
trading_date
2023-01-02      1504.967142   INFY
2023-01-03      1503.584499   INFY
2023-01-04      1510.061384   INFY
2023-01-05      1525.291682   INFY
2023-01-06      1522.950149   INFY
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning: Do not pass an `input_shape`/`input_dim` argum
  super().__init__(**kwargs)
Model: "sequential_1"
```

| Layer (type) | Output Shape | Param # |
|---|---|---|
| lstm_2 (LSTM) | (None, 30, 64) | 16,896 |
| lstm_3 (LSTM) | (None, 32) | 12,416 |
| dense_1 (Dense) | (None, 1) | 33 |

```
 Total params: 29,345 (114.63 KB)
 Trainable params: 29,345 (114.63 KB)
 Non-trainable params: 0 (0.00 B)
Epoch 1/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 6s 98ms/step - loss: 0.0821 - val_loss: 0.0059
Epoch 2/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 52ms/step - loss: 0.0138 - val_loss: 0.0347
Epoch 3/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 53ms/step - loss: 0.0098 - val_loss: 0.0034
Epoch 4/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 54ms/step - loss: 0.0066 - val_loss: 0.0075
Epoch 5/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step - loss: 0.0055 - val_loss: 0.0036
Epoch 6/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 0.0053 - val_loss: 0.0036
Epoch 7/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0055 - val_loss: 0.0036
Epoch 8/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0052 - val_loss: 0.0041
Epoch 9/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0046 - val_loss: 0.0037
Epoch 10/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - loss: 0.0054 - val_loss: 0.0041
Epoch 11/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - loss: 0.0044 - val_loss: 0.0038
Epoch 12/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.0042 - val_loss: 0.0039
Epoch 13/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0042 - val_loss: 0.0041
Epoch 14/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.0039 - val_loss: 0.0041
Epoch 15/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0030 - val_loss: 0.0044
Epoch 16/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0042 - val_loss: 0.0055
Epoch 17/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 0.0037 - val_loss: 0.0043
Epoch 18/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.0032 - val_loss: 0.0045
Epoch 19/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.0034 - val_loss: 0.0045
Epoch 20/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.0035 - val_loss: 0.0054
Epoch 21/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0034 - val_loss: 0.0063
Epoch 22/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0033 - val_loss: 0.0044
Epoch 23/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.0034 - val_loss: 0.0042
Epoch 24/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 47ms/step - loss: 0.0028 - val_loss: 0.0066
Epoch 25/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 50ms/step - loss: 0.0040 - val_loss: 0.0043
Epoch 26/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 52ms/step - loss: 0.0034 - val_loss: 0.0047
Epoch 27/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 51ms/step - loss: 0.0027 - val_loss: 0.0041
Epoch 28/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 51ms/step - loss: 0.0026 - val_loss: 0.0055
Epoch 29/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0028 - val_loss: 0.0045
Epoch 30/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0030 - val_loss: 0.0059
Epoch 31/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0028 - val_loss: 0.0041
Epoch 32/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.0028 - val_loss: 0.0039
Epoch 33/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - loss: 0.0029 - val_loss: 0.0042
Epoch 34/50
```

```
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - loss: 0.0025 - val_loss: 0.0046
Epoch 35/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - loss: 0.0029 - val_loss: 0.0037
Epoch 36/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0029 - val_loss: 0.0037
Epoch 37/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.0026 - val_loss: 0.0036
Epoch 38/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 0.0024 - val_loss: 0.0044
Epoch 39/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0024 - val_loss: 0.0039
Epoch 40/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - loss: 0.0030 - val_loss: 0.0038
Epoch 41/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - loss: 0.0022 - val_loss: 0.0034
Epoch 42/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.0023 - val_loss: 0.0050
Epoch 43/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0027 - val_loss: 0.0029
Epoch 44/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0022 - val_loss: 0.0030
Epoch 45/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0025 - val_loss: 0.0044
Epoch 46/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - loss: 0.0024 - val_loss: 0.0033
Epoch 47/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 61ms/step - loss: 0.0022 - val_loss: 0.0029
Epoch 48/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 53ms/step - loss: 0.0024 - val_loss: 0.0043
Epoch 49/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 54ms/step - loss: 0.0024 - val_loss: 0.0068
Epoch 50/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0022 - val_loss: 0.0038
4/4 ━━━━━━━━━━━━━━━━━━━━ 1s 111ms/step
MSE: 487.5930, MAE: 17.9634
```



Double-click (or enter) to edit

```
!pip install gradio
```

```
Collecting gradio
  Downloading gradio-5.30.0-py3-none-any.whl.metadata (16 kB)
Collecting aiofiles<25.0,>=22.0 (from gradio)
  Downloading aiofiles-24.1.0-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: anyio<5.0,>=3.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (4.9.0)
Collecting fastapi<1.0,>=0.115.2 (from gradio)
  Downloading fastapi-0.115.12-py3-none-any.whl.metadata (27 kB)
Collecting ffmpy (from gradio)
  Downloading ffmpy-0.5.0-py3-none-any.whl.metadata (3.0 kB)
Collecting gradio-client==1.10.1 (from gradio)
  Downloading gradio_client-1.10.1-py3-none-any.whl.metadata (7.1 kB)
Collecting groovy~=0.1 (from gradio)
  Downloading groovy-0.1.2-py3-none-any.whl.metadata (6.1 kB)
Requirement already satisfied: httpx>=0.24.1 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.28.1)
Requirement already satisfied: huggingface-hub>=0.28.1 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.31.2)
Requirement already satisfied: jinja2<4.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (3.1.6)
Requirement already satisfied: markupsafe<4.0,>=2.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (3.0.2)
Requirement already satisfied: numpy<3.0,>=1.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.0.2)
Requirement already satisfied: orjson~=3.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (3.10.18)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from gradio) (24.2)
Requirement already satisfied: pandas<3.0,>=1.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.2.2)
Requirement already satisfied: pillow<12.0,>=8.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (11.2.1)
Requirement already satisfied: pydantic<2.12,>=2.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.11.4)
Collecting pydub (from gradio)
  Downloading pydub-0.25.1-py2.py3-none-any.whl.metadata (1.4 kB)
Collecting python-multipart>=0.0.18 (from gradio)
  Downloading python_multipart-0.0.20-py3-none-any.whl.metadata (1.8 kB)
Requirement already satisfied: pyyaml<7.0,>=5.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (6.0.2)
Collecting ruff>=0.9.3 (from gradio)
  Downloading ruff-0.11.10-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (25 kB)
Collecting safehttpx<0.2.0,>=0.1.6 (from gradio)
  Downloading safehttpx-0.1.6-py3-none-any.whl.metadata (4.2 kB)
Collecting semantic-version~=2.0 (from gradio)
  Downloading semantic_version-2.10.0-py2.py3-none-any.whl.metadata (9.7 kB)
Collecting starlette<1.0,>=0.40.0 (from gradio)
  Downloading starlette-0.46.2-py3-none-any.whl.metadata (6.2 kB)
Collecting tomlkit<0.14.0,>=0.12.0 (from gradio)
  Downloading tomlkit-0.13.2-py3-none-any.whl.metadata (2.7 kB)
Requirement already satisfied: typer<1.0,>=0.12 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.15.3)
Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (4.13.2)
Collecting uvicorn>=0.14.0 (from gradio)
  Downloading uvicorn-0.34.2-py3-none-any.whl.metadata (6.5 kB)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from gradio-client==1.10.1->gradio) (2025.3.2)
Requirement already satisfied: websockets<16.0,>=10.0 in /usr/local/lib/python3.11/dist-packages (from gradio-client==1.10.1->gra
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.11/dist-packages (from anyio<5.0,>=3.0->gradio) (3.10)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-packages (from anyio<5.0,>=3.0->gradio) (1.3.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from httpx>=0.24.1->gradio) (2025.4.26)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-packages (from httpx>=0.24.1->gradio) (1.0.9)
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.11/dist-packages (from httpcore==1.*->httpx>=0.24.1->gradio) (
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (3.18.0
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (2.32.3
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (4.
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2025.2)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradi
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradio
```

```
import numpy as np
import pandas as pd
import gradio as gr
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense  # or GRU
from tensorflow.keras.callbacks import EarlyStopping

# 2. Load and preprocess data
df = pd.read_csv("/content/stock_data.csv")
close_prices = df['closing_price'].values.reshape(-1, 1)  # Replace 'Close' with your column name

scaler = MinMaxScaler(feature_range=(0, 1))
scaled_prices = scaler.fit_transform(close_prices)

sequence_length = 30  # Or whatever your model uses

def create_sequences(data, seq_length):
    X, y = [], []
    for i in range(len(data) - seq_length):
        X.append(data[i:i+seq_length])
        y.append(data[i+seq_length])
```

```
        return np.array(X), np.array(y)

    X, y = create_sequences(scaled_prices, sequence_length)

    # 3. Split into train/test (simple split)
    split = int(0.8 * len(X))
    X_train, X_test = X[:split], X[split:]
    y_train, y_test = y[:split], y[split:]

    # 4. Build and train the model
    model = Sequential([
        LSTM(50, return_sequences=False, input_shape=(sequence_length, 1)),  # or GRU
        Dense(1)
    ])
    model.compile(optimizer='adam', loss='mse')
    early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
    model.fit(X_train, y_train, epochs=20, batch_size=16, validation_data=(X_test, y_test), callbacks=[early_stop]

    # 5. Define Gradio prediction function
    def predict_next_price(prices):
        try:
            # Convert input to list of floats
            if isinstance(prices, list):
                prices = [float(p[0]) if isinstance(p, list) else float(p) for p in prices]
            else:
                return "Input must be a list."
            if len(prices) != sequence_length:
                return f"Please enter exactly {sequence_length} prices."
            input_scaled = scaler.transform(np.array(prices).reshape(-1, 1))
            X_input = input_scaled.reshape((1, sequence_length, 1))
            pred_scaled = model.predict(X_input)
            pred_price = scaler.inverse_transform(pred_scaled)
            return float(pred_price[0][0])
        except Exception as e:
            return f"Error: {e}"

    # 6. Create Gradio interface
    iface = gr.Interface(
        fn=predict_next_price,
        inputs=gr.Dataframe(
            headers=["Closing Price"],
```