

```
# 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 ---
df = pd.read_csv("/content/drive/MyDrive/stock_data(1) .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` argument to the first layer of a model. Passing an `input_shape` is deprecated, and will be removed in a future version of Keras. Use `input_shape` only for single-layer models.

Model: "sequential_2"

Layer (type)	Output Shape	Param #
lstm_4 (LSTM)	(None, 30, 64)	16,896
lstm_5 (LSTM)	(None, 32)	12,416
dense_2 (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 ————— 8s 155ms/step - loss: 0.0610 - val_loss: 0.0078

Epoch 2/50

12/12 ————— 1s 99ms/step - loss: 0.0095 - val_loss: 0.0131

Epoch 3/50

12/12 ————— 1s 108ms/step - loss: 0.0069 - val_loss: 0.0058

Epoch 4/50

12/12 ————— 3s 114ms/step - loss: 0.0065 - val_loss: 0.0041

Epoch 5/50

12/12 ————— 1s 82ms/step - loss: 0.0052 - val_loss: 0.0035

Epoch 6/50

12/12 ————— 1s 58ms/step - loss: 0.0050 - val_loss: 0.0043

Epoch 7/50

12/12 ————— 0s 33ms/step - loss: 0.0047 - val_loss: 0.0039

Epoch 8/50

12/12 ————— 1s 33ms/step - loss: 0.0040 - val_loss: 0.0038

Epoch 9/50

12/12 ————— 1s 36ms/step - loss: 0.0042 - val_loss: 0.0063

Epoch 10/50

12/12 ————— 0s 33ms/step - loss: 0.0038 - val_loss: 0.0045
Epoch 11/50
12/12 ————— 0s 37ms/step - loss: 0.0042 - val_loss: 0.0043
Epoch 12/50
12/12 ————— 1s 34ms/step - loss: 0.0038 - val_loss: 0.0104
Epoch 13/50
12/12 ————— 1s 35ms/step - loss: 0.0050 - val_loss: 0.0042
Epoch 14/50
12/12 ————— 1s 36ms/step - loss: 0.0043 - val_loss: 0.0057
Epoch 15/50
12/12 ————— 0s 35ms/step - loss: 0.0038 - val_loss: 0.0052
Epoch 16/50
12/12 ————— 0s 34ms/step - loss: 0.0034 - val_loss: 0.0040
Epoch 17/50
12/12 ————— 1s 35ms/step - loss: 0.0029 - val_loss: 0.0043
Epoch 18/50
12/12 ————— 1s 33ms/step - loss: 0.0029 - val_loss: 0.0056
Epoch 19/50
12/12 ————— 0s 35ms/step - loss: 0.0033 - val_loss: 0.0049
Epoch 20/50
12/12 ————— 0s 33ms/step - loss: 0.0031 - val_loss: 0.0091
Epoch 21/50
12/12 ————— 1s 35ms/step - loss: 0.0030 - val_loss: 0.0041
Epoch 22/50
12/12 ————— 0s 37ms/step - loss: 0.0026 - val_loss: 0.0041
Epoch 23/50
12/12 ————— 1s 52ms/step - loss: 0.0028 - val_loss: 0.0046
Epoch 24/50
12/12 ————— 1s 50ms/step - loss: 0.0026 - val_loss: 0.0046
Epoch 25/50
12/12 ————— 1s 54ms/step - loss: 0.0025 - val_loss: 0.0045
Epoch 26/50
12/12 ————— 1s 40ms/step - loss: 0.0025 - val_loss: 0.0039
Epoch 27/50
12/12 ————— 1s 33ms/step - loss: 0.0027 - val_loss: 0.0038
Epoch 28/50
12/12 ————— 0s 36ms/step - loss: 0.0025 - val_loss: 0.0042
Epoch 29/50
12/12 ————— 1s 33ms/step - loss: 0.0027 - val_loss: 0.0039
Epoch 30/50
12/12 ————— 0s 38ms/step - loss: 0.0027 - val_loss: 0.0040
Epoch 31/50
12/12 ————— 0s 36ms/step - loss: 0.0025 - val_loss: 0.0054

12/12 _____ 0s 30ms/step - loss: 0.0023 - val_loss: 0.0034
Epoch 32/50
12/12 _____ 0s 36ms/step - loss: 0.0028 - val_loss: 0.0037
Epoch 33/50
12/12 _____ 0s 32ms/step - loss: 0.0025 - val_loss: 0.0037
Epoch 34/50
12/12 _____ 0s 34ms/step - loss: 0.0026 - val_loss: 0.0032
Epoch 35/50
12/12 _____ 1s 33ms/step - loss: 0.0022 - val_loss: 0.0036
Epoch 36/50
12/12 _____ 0s 35ms/step - loss: 0.0025 - val_loss: 0.0061
Epoch 37/50
12/12 _____ 0s 36ms/step - loss: 0.0025 - val_loss: 0.0032
Epoch 38/50
12/12 _____ 1s 35ms/step - loss: 0.0024 - val_loss: 0.0032
Epoch 39/50
12/12 _____ 0s 35ms/step - loss: 0.0023 - val_loss: 0.0034
Epoch 40/50
12/12 _____ 0s 34ms/step - loss: 0.0020 - val_loss: 0.0045
Epoch 41/50
12/12 _____ 1s 36ms/step - loss: 0.0028 - val_loss: 0.0035
Epoch 42/50
12/12 _____ 0s 35ms/step - loss: 0.0023 - val_loss: 0.0030
Epoch 43/50
12/12 _____ 1s 34ms/step - loss: 0.0021 - val_loss: 0.0029
Epoch 44/50
12/12 _____ 1s 38ms/step - loss: 0.0022 - val_loss: 0.0029
Epoch 45/50
12/12 _____ 0s 34ms/step - loss: 0.0021 - val_loss: 0.0033
Epoch 46/50
12/12 _____ 1s 54ms/step - loss: 0.0023 - val_loss: 0.0031
Epoch 47/50
12/12 _____ 1s 52ms/step - loss: 0.0023 - val_loss: 0.0031
Epoch 48/50
12/12 _____ 1s 58ms/step - loss: 0.0022 - val_loss: 0.0047
Epoch 49/50
12/12 _____ 1s 37ms/step - loss: 0.0020 - val_loss: 0.0039
Epoch 50/50
12/12 _____ 0s 33ms/step - loss: 0.0021 - val_loss: 0.0025
4/4 _____ 1s 115ms/step
MSE: 298.8003, MAE: 13.9100

Stock Price Prediction



