Surakshith Shetty-53026240013

3a. Stock Price Prediction using LSTM

```
#Reliance stock price prediction, using LSTM model
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
```

```
# Load the dataset
file_path = "500325.csv"  # Ensure the file is in the same directory
df = pd.read_csv(file_path)
```

```
df.head()
```

	Date	Open Price	High Price	Low Price	Close Price	WAP	No.of Shares	No. of Trades	Total Turnover (Rs.)	Deliverable Quantity	% Deli. Qty to Traded Qty	Spread High-Low	Spread Close-Open
0	20-February-2025	1226.25	1239.40	1223.00	1233.05	1235.400020	206262	6415	254816079.0	134142.0	65.03	16.40	6.80
1	19-February-2025	1222.75	1232.80	1217.60	1226.95	1226.458523	110206	3823	135163088.0	41958.0	38.07	15.20	4.20
2	18-February-2025	1225.25	1229.95	1216.40	1224.95	1223.204760	206583	7437	252693309.0	103728.0	50.21	13.55	-0.30
	17-												

```
# Convert 'Date' column to datetime and sort in ascending order
df["Date"] = pd.to_datetime(df["Date"], format="%d-%B-%Y")
df = df.sort_values(by="Date")

# Extract 'Close Price' for training
data = df["Close Price"].values.reshape(-1, 1)
```

```
df.head()
```

	Date	Open Price	High Price	Low Price	Close Price	WAP	No.of Shares	No. of Trades	Total Turnover (Rs.)	Deliverable Quantity	% Deli. Qty to Traded Qty	Spread High-Low	Spread Close-Open
286	2024-01-01	2581.05	2606.00	2573.55	2589.85	2587.308555	67641	6352	1.750081e+08	29008.0	42.89	32.45	8.80
285	2024-01-02	2587.65	2614.90	2573.50	2610.90	2592.670654	86186	8001	2.234519e+08	33574.0	38.96	41.40	23.25
284	2024-01-03	2608.10	2634.00	2577.15	2582.95	2607.804848	107501	7371	2.803416e+08	43758.0	40.70	56.85	-25.15
283	2024-01-04	2589.40	2609.75	2580.00	2597.40	2599.882922	162584	9815	4.226994e+08	103141.0	63.44	29.75	8.00

```
data[0:10]
```

```
array([[2589.85],
       [2610.9 ],
       [2582.95],
       [2597.4 ],
       [2606.75],
       [2586.1 ],
       [2580.6 ],
       [2649.95],
       [2718.4 ],
       [2740.1 ]])
```

```
# Normalize data using MinMaxScaler
scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(data)
```

```
data_scaled[0:10]
```

```
array([[0.69327689],
       [0.70382245],
       [0.68982015],
       [0.69705927],
       [0.7017434 ],
       [0.69139823],
       [0.68864285],
       [0.7233856 ],
       [0.75767747],
       [0.76854867]])
```

```
# Function to create sequences for LSTM
def create_sequences(data, seq_length, prediction_length):
    X, y = [], []
    for i in range(len(data) - seq_length - prediction_length + 1):
        X.append(data[i:i + seq_length])
        y.append(data[i + seq_length:i + seq_length + prediction_length])
    return np.array(X), np.array(y)

# Define sequence and prediction lengths
seq_length = 10  # Use last 10 days to predict
prediction_length = 7  # Predict next 7 days

# Create sequences
X, y = create_sequences(data_scaled, seq_length, prediction_length)

# Split into training and testing sets (80% training, 20% testing)
train_size = int(len(X) * 0.8)
X_train, y_train = X[:train_size], y[:train_size]
X_test, y_test = X[train_size:], y[train_size:]
```

```
X_train.shape
```

```
(216, 10, 1)
```

```
X_test.shape
```

```
(55, 10, 1)
```

```
# Build LSTM model
model = Sequential([
    LSTM(50, activation="relu", return_sequences=True, input_shape=(seq_length, 1)),
    LSTM(50, activation="relu"),
    Dense(prediction_length)  # Predicts for next 7 days
])
```

```
/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.compile(optimizer="adam", loss="mse")

# Train the model
model.fit(X_train, y_train, epochs=20, batch_size=16, verbose=1)
```

```
Epoch 1/20
14/14 ━━━━━━━━━━━━━━━━━━━━ 5s 11ms/step - loss: 0.6354
Epoch 2/20
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.4312
Epoch 3/20
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.1710
Epoch 4/20
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0436
Epoch 5/20
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0239
Epoch 6/20
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0197
Epoch 7/20
```

```
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0149
Epoch 8/20
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0201
Epoch 9/20
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0150
Epoch 10/20
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0174
Epoch 11/20
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0181
Epoch 12/20
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0191
Epoch 13/20
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0118
Epoch 14/20
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0179
Epoch 15/20
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0150
Epoch 16/20
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0172
Epoch 17/20
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0137
Epoch 18/20
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0144
Epoch 19/20
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0191
Epoch 20/20
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - loss: 0.0138
<keras.src.callbacks.history.History at 0x7df6d1894150>
```

```
# Predict next 7 days using the latest sequence
last_sequence = data_scaled[-seq_length:].reshape(1, seq_length, 1)
predicted_scaled = model.predict(last_sequence)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 362ms/step
```

```
# Convert back to actual price values
predicted_prices = scaler.inverse_transform(predicted_scaled.reshape(-1, 1))
```

```
# Generate future dates
last_date = df["Date"].iloc[-1]
future_dates = pd.date_range(start=last_date + pd.Timedelta(days=1), periods=10)
```

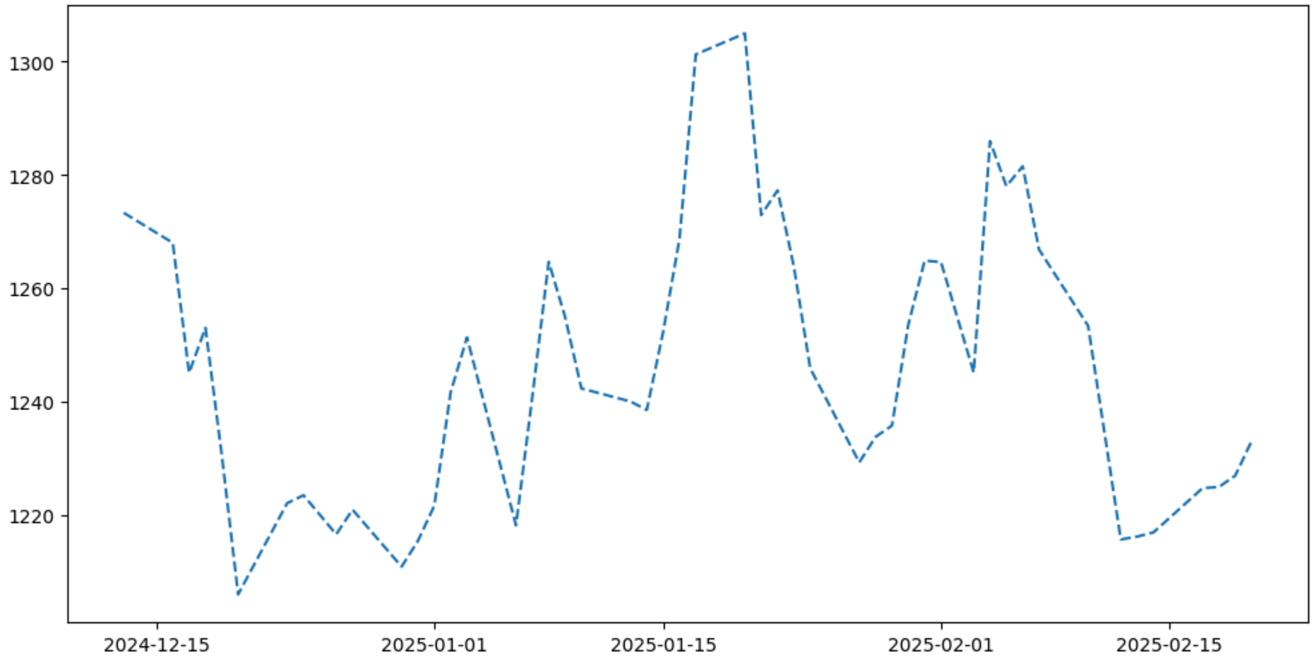
```
# Plot actual vs predicted prices
plt.figure(figsize=(12, 6))
plt.plot(df["Date"].iloc[-50:], df["Close Price"].iloc[-50:], label="Actual Close Prices", linestyle="dashed")
plt.plot(future_dates, predicted_prices, label="Predicted Close Prices", marker="o")
plt.xlabel("Date")
plt.ylabel("Stock Price")
plt.xticks(rotation=45)
plt.legend()
plt.title("Reliance Stock Price Prediction (Next 7 Days)")
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-31-490fffca94da> in <cell line: 0>()
      2 plt.figure(figsize=(12, 6))
      3 plt.plot(df["Date"].iloc[-50:], df["Close Price"].iloc[-50:], label="Actual Close Prices", linestyle="dashed")
----> 4 plt.plot(future_dates, predicted_prices, label="Predicted Close Prices", marker="o")
      5 plt.xlabel("Date")
      6 plt.ylabel("Stock Price")
```

3 frames

```
/usr/local/lib/python3.11/dist-packages/matplotlib/axes/_base.py in _plot_args(self, axes, tup, kwargs, return_kwargs,
ambiguous_fmt_datakey)
    492
    493         if x.shape[0] != y.shape[0]:
--> 494             raise ValueError(f"x and y must have same first dimension, but "
    495                              f"have shapes {x.shape} and {y.shape}")
    496         if x.ndim > 2 or y.ndim > 2:

ValueError: x and y must have same first dimension, but have shapes (10,) and (7, 1)
```



Start coding or generate with AI.