In [1]:

```
import lasio
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense, LSTM
from sklearn.metrics import mean_squared_error, r2_score

# Load the data
las= lasio.read("ColvilleUnit1.LAS")
well = las.df()
data =well.dropna(how="any")

# Preprocess the data
X = data[['CALI', 'GR', 'ILD', 'ILM', 'LL8', 'RHOB', 'SP', 'SWL'] ].values
y = data['GR'].values

# Normalize the data
X_norm = (X - np.mean(X)) / np.std(X)
y_norm = (y - np.mean(y)) / np.std(y)

# Split the data into training and testing sets
train_size = int(len(X_norm) * 0.8)
X_train, X_test = X_norm[:train_size], X_norm[train_size:]
y_train, y_test = y_norm[:train_size], y_norm[train_size:]

depth = data.index

# Normalize the data
X_norm = (X - np.mean(X)) / np.std(X)
y_norm = (y - np.mean(y)) / np.std(y)

# Split the data into training and testing sets
train_size = int(len(X_norm) * 0.8)
X_train, X_test = X_norm[:train_size], X_norm[train_size:]
y_train, y_test = y_norm[:train_size], y_norm[train_size:]
depth_test = depth[train_size:]

# Define the LSTM model
model = Sequential()
model.add(LSTM(50, input_shape=(X_train.shape[1], 1)))
model.add(Dense(1))

# Compile the model
model.compile(loss='mse', optimizer='adam')

# Train the model
model.fit(X_train.reshape((X_train.shape[0], X_train.shape[1], 1)), y_train, e

# Evaluate the model
y_pred = model.predict(X_test.reshape((X_test.shape[0], X_test.shape[1], 1)))
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```
Epoch 1/50
152/152 - 5s - loss: 0.4348 - 5s/epoch - 35ms/step
Epoch 2/50
152/152 - 1s - loss: 0.0404 - 744ms/epoch - 5ms/step
Epoch 3/50
152/152 - 1s - loss: 0.0227 - 852ms/epoch - 6ms/step
Epoch 4/50
152/152 - 1s - loss: 0.0119 - 1s/epoch - 7ms/step
Epoch 5/50
152/152 - 1s - loss: 0.0071 - 1s/epoch - 8ms/step
Epoch 6/50
152/152 - 1s - loss: 0.0060 - 1s/epoch - 8ms/step
Epoch 7/50
152/152 - 1s - loss: 0.0050 - 1s/epoch - 7ms/step
Epoch 8/50
152/152 - 1s - loss: 0.0040 - 1s/epoch - 7ms/step
Epoch 9/50
152/152 - 1s - loss: 0.0035 - 1s/epoch - 7ms/step
Epoch 10/50
152/152 - 1s - loss: 0.0030 - 1s/epoch - 7ms/step
Epoch 11/50
152/152 - 1s - loss: 0.0029 - 1s/epoch - 7ms/step
Epoch 12/50
152/152 - 1s - loss: 0.0024 - 1s/epoch - 7ms/step
Epoch 13/50
152/152 - 1s - loss: 0.0028 - 1s/epoch - 7ms/step
Epoch 14/50
152/152 - 1s - loss: 0.0028 - 1s/epoch - 7ms/step
Epoch 15/50
152/152 - 1s - loss: 0.0023 - 1s/epoch - 8ms/step
Epoch 16/50
152/152 - 1s - loss: 0.0024 - 949ms/epoch - 6ms/step
Epoch 17/50
152/152 - 1s - loss: 0.0020 - 781ms/epoch - 5ms/step
Epoch 18/50
152/152 - 1s - loss: 0.0023 - 910ms/epoch - 6ms/step
Epoch 19/50
152/152 - 1s - loss: 0.0019 - 1s/epoch - 8ms/step
Epoch 20/50
152/152 - 1s - loss: 0.0021 - 1s/epoch - 7ms/step
Epoch 21/50
152/152 - 1s - loss: 0.0020 - 1s/epoch - 7ms/step
Epoch 22/50
152/152 - 1s - loss: 0.0018 - 1s/epoch - 7ms/step
Epoch 23/50
152/152 - 1s - loss: 0.0017 - 1s/epoch - 8ms/step
Epoch 24/50
152/152 - 1s - loss: 0.0019 - 1s/epoch - 7ms/step
Epoch 25/50
152/152 - 1s - loss: 0.0017 - 1s/epoch - 8ms/step
Epoch 26/50
152/152 - 1s - loss: 0.0017 - 1s/epoch - 8ms/step
Epoch 27/50
152/152 - 1s - loss: 0.0019 - 1s/epoch - 8ms/step
Epoch 28/50
152/152 - 1s - loss: 0.0016 - 1s/epoch - 8ms/step
Epoch 29/50
```

```
152/152 - 1s - loss: 0.0015 - 1s/epoch - 8ms/step
Epoch 30/50
152/152 - 1s - loss: 0.0015 - 1s/epoch - 7ms/step
Epoch 31/50
152/152 - 1s - loss: 0.0024 - 785ms/epoch - 5ms/step
Epoch 32/50
152/152 - 1s - loss: 0.0013 - 932ms/epoch - 6ms/step
Epoch 33/50
152/152 - 1s - loss: 0.0012 - 1s/epoch - 7ms/step
Epoch 34/50
152/152 - 1s - loss: 0.0013 - 1s/epoch - 7ms/step
Epoch 35/50
152/152 - 1s - loss: 0.0013 - 1s/epoch - 8ms/step
Epoch 36/50
152/152 - 1s - loss: 0.0013 - 1s/epoch - 8ms/step
Epoch 37/50
152/152 - 1s - loss: 9.6227e-04 - 1s/epoch - 7ms/step
Epoch 38/50
152/152 - 1s - loss: 0.0013 - 1s/epoch - 7ms/step
Epoch 39/50
152/152 - 1s - loss: 0.0011 - 1s/epoch - 8ms/step
Epoch 40/50
152/152 - 1s - loss: 0.0011 - 1s/epoch - 8ms/step
Epoch 41/50
152/152 - 1s - loss: 0.0010 - 1s/epoch - 7ms/step
Epoch 42/50
152/152 - 1s - loss: 9.8428e-04 - 1s/epoch - 7ms/step
Epoch 43/50
152/152 - 1s - loss: 0.0011 - 1s/epoch - 8ms/step
Epoch 44/50
152/152 - 1s - loss: 8.8409e-04 - 1s/epoch - 7ms/step
Epoch 45/50
152/152 - 1s - loss: 8.4675e-04 - 879ms/epoch - 6ms/step
Epoch 46/50
152/152 - 1s - loss: 8.0548e-04 - 764ms/epoch - 5ms/step
Epoch 47/50
152/152 - 1s - loss: 6.9940e-04 - 934ms/epoch - 6ms/step
Epoch 48/50
152/152 - 1s - loss: 9.4306e-04 - 1s/epoch - 7ms/step
Epoch 49/50
152/152 - 1s - loss: 7.8167e-04 - 1s/epoch - 7ms/step
Epoch 50/50
152/152 - 1s - loss: 7.6704e-04 - 1s/epoch - 8ms/step
38/38 [==============================] - 1s 4ms/step
```

In [2]:

```
mse
```

Out[2]: 0.009986492301903986

In [3]:

Out[3]: 0.991648148175009

In [26]:

```
# Plot the predicted and actual values as subplots
fig, (ax1, ax2) = plt.subplots(2, 1, sharex=True, figsize=(10,10))
ax1.plot(depth_test, y_test, label='Actual')

ax1.legend()
ax1.set_title('GAMA RAY LOG Predictions')
ax1.set_ylabel('GAMA RAY LOG')

ax2.plot(depth_test, y_pred, label='Predicted',color='r')
ax2.set_xlabel('Depth')
ax2.set_ylabel('Predicted GR')
ax2.legend()
```