COS30018 – Intelligent System

Option B: Stock Price Prediction

Report v0.4

Name Phong Tran
Student Id:104334842

Class: 1-5
Tutor: Tuan Dung Lai

Table of Contents

Summary function to create DL models

```
def build_custom_model(layer_type='LSTM', input_shape=(30, 1), layer_sizes=[50], num_layers=1, dropout=0.2):
    """
    Function to build a custom deep learning model based on provided parameters.

    Parameters:
    layer_type: str - The type of recurrent layer (e.g., 'LSTM', 'GRU', 'RNN').
    input_shape: tuple - Shape of the input data (timesteps, features).
    layer_sizes: list - A list containing the number of units for each layer.
    num_layers: int - The number of recurrent layers to be added to the model.
    dropout: float - Dropout rate to prevent overfitting.

    Returns:
    model: A compiled Keras model.
    """
    model = Sequential()
    # Add the Input layer
    model.add(Input(shape=input_shape))

    # Add recurrent layers based on layer type
    for i in range(num_layers):
        units = layer_sizes[i] if i < len(layer_sizes) else layer_sizes[-1]  # If more layers than sizes, repeat the last size

        if layer_type == 'LSTM':
            model.add(LSTM(units, return_sequences=(i < num_layers - 1)))
        elif layer_type == 'GRU':
            model.add(GRU(units, return_sequences=(i < num_layers - 1)))
        elif layer_type == 'RNN':
            model.add(SimpleRNN(units, return_sequences=(i < num_layers - 1)))

        # Add dropout after each recurrent layer to prevent overfitting
        model.add(Dropout(dropout))

    # Add a Dense layer as the output layer
    model.add(Dense(1))  # Single output for regression tasks

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

    return model
```

This sequential neural network will used to build sequence model architecture includes with a simple linear stack of layers. First of all, it add the input layer with shapes. Then it add the layer that we have been specified, depending on number of layers that we have been specified in the parameters into three options for type of the layer: LSTM or GRU or RNN. Adding dropout layer of 0.2 means 20% chance of a neuron being dropped in order to prevent overfitting. Finally, include with Dense, which is the last layer, and include only one output for regression task. Together to compile into the model with adam just model weights optimizer and using mean_squared_error loss for minimizing during training.

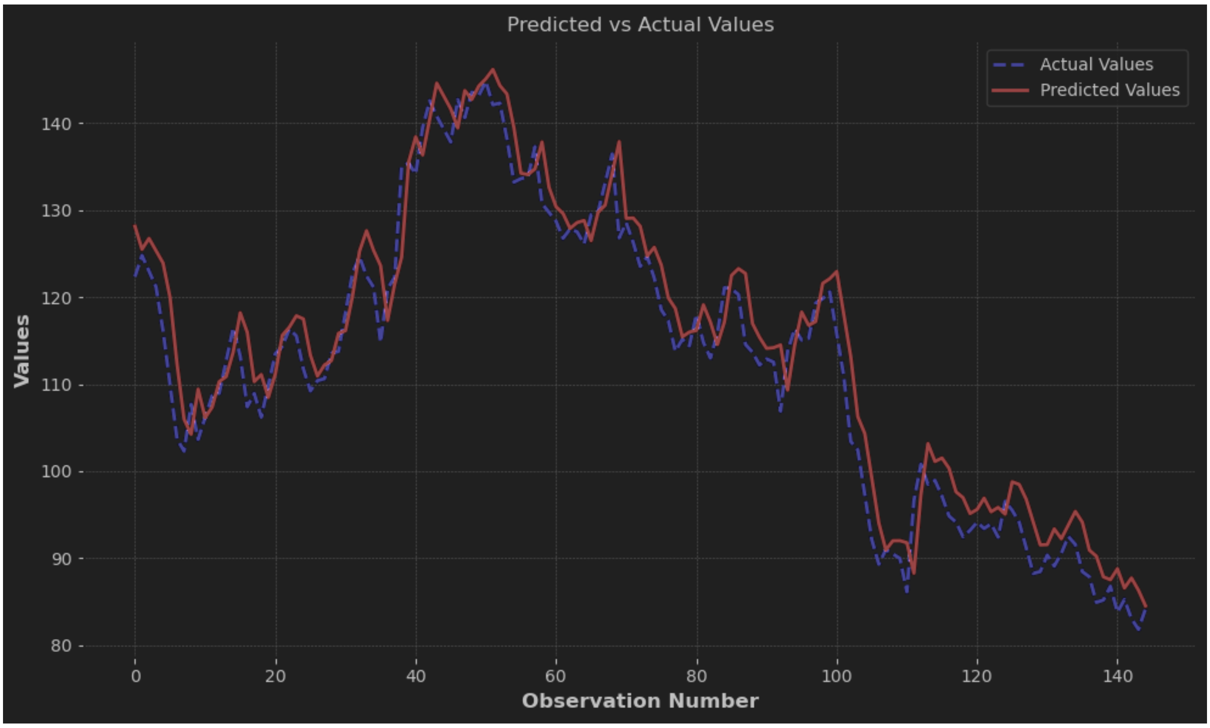
Summary experiments of different configuration of DL models and model training

```
epochs = 50
batch_size = 32
```

In every different configuration, I will try to keep the same epochs and batch size, cause if more for one or two of them means that the accuracy will be better.

```
model_rnn1 = build_custom_model(layer_type='RNN', input_shape=(30, 1), layer_sizes=[200], num_layers=2, dropout=0.2)
history1 = model_rnn1.fit(X_train, y_train, epochs=epochs, batch_size=batch_size, validation_data=(X_test, y_test), shuffle=False)
```

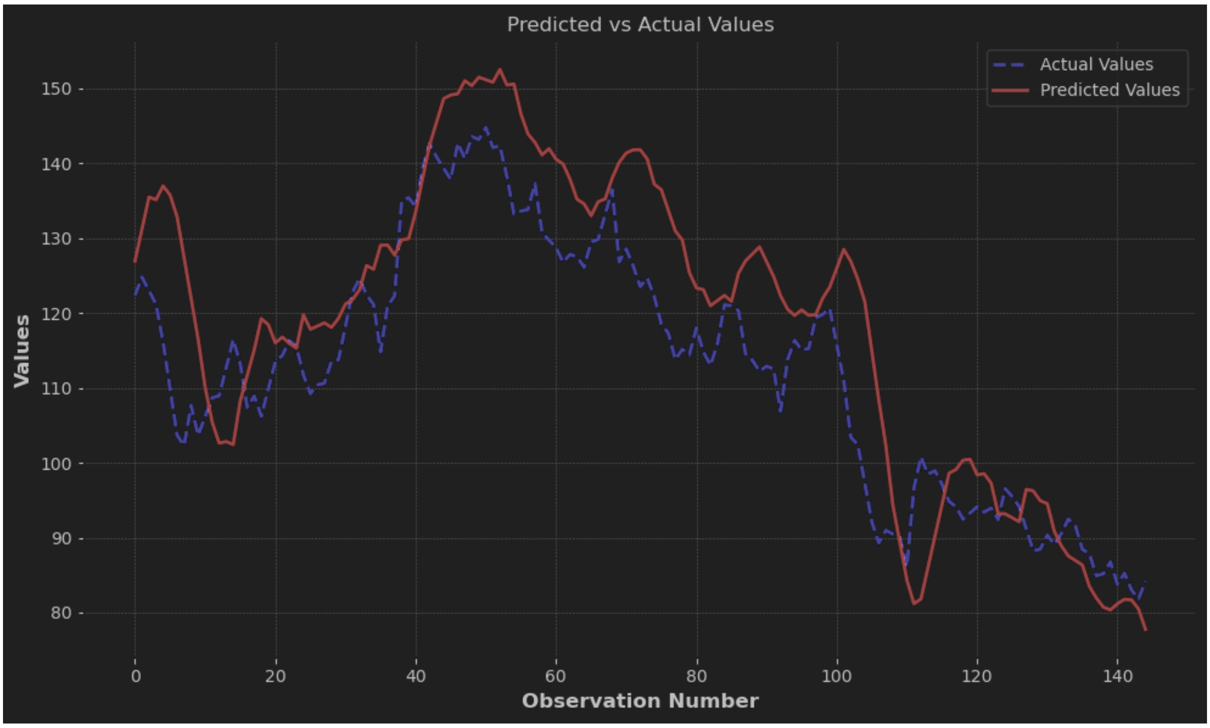
Train the Recurrent Neural Network model with layer size, number of hidden layers, and dropout, before let the model fit with x and y train and number of epochs and batch size abve, with validation data to make sure the model perform well.



The graph shows that the gap between actual and predicted values is not much different, means that the deviate is much lower.

```
model_rnn2 = build_custom_model(layer_type='RNN', input_shape=(30, 1), layer_sizes=[50], num_layers=3, dropout=0.3)
history1 = model_rnn2.fit(X_train, y_train, epochs=epochs, batch_size=batch_size, validation_data=(X_test, y_test), shuffle=False)
```

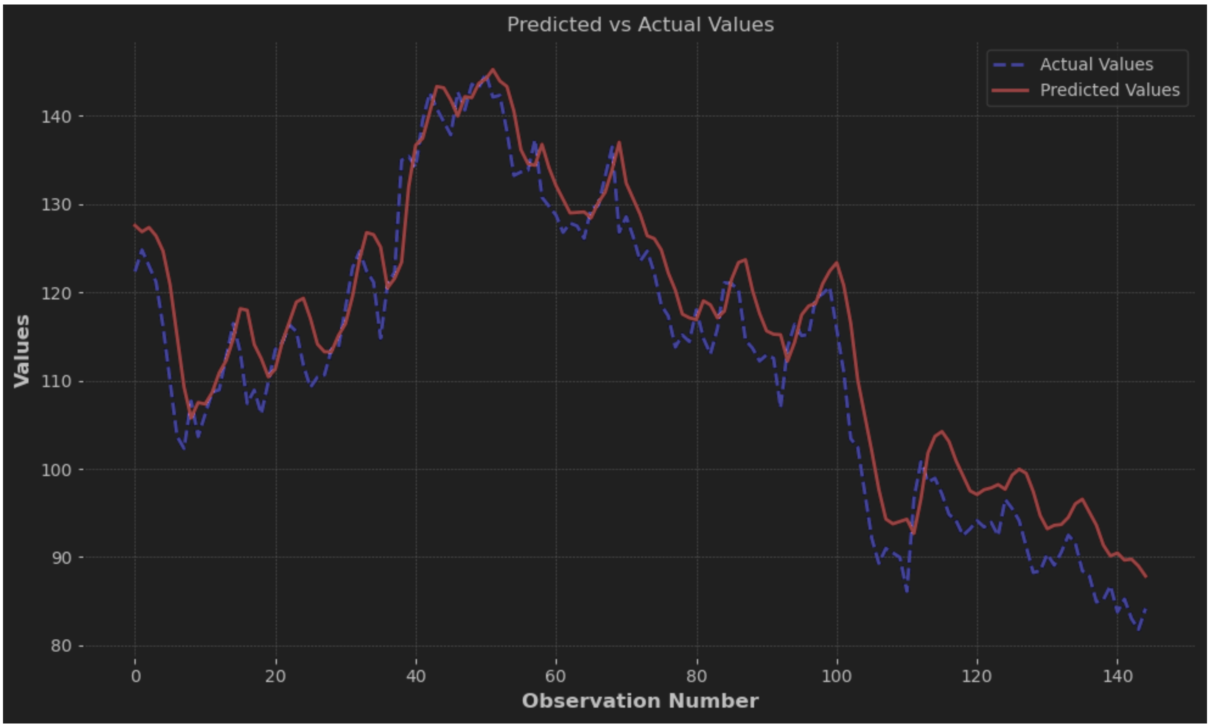
Train the Recurrent Neural Network model but different with more layer size, number of hidden layer, and dropout. Still fitting like Recurrent Neural Network with same parameters.



The plot shows that the difference gap between actual and predicted values, which means that there is a significant deviate between these values.

```
model_gru1 = build_custom_model(layer_type='GRU', input_shape=(30, 1), layer_sizes=[100], num_layers=1, dropout=0.2)
history1 = model_gru1.fit(X_train, y_train, epochs=epochs, batch_size=batch_size, validation_data=(X_test, y_test), shuffle=False)
```

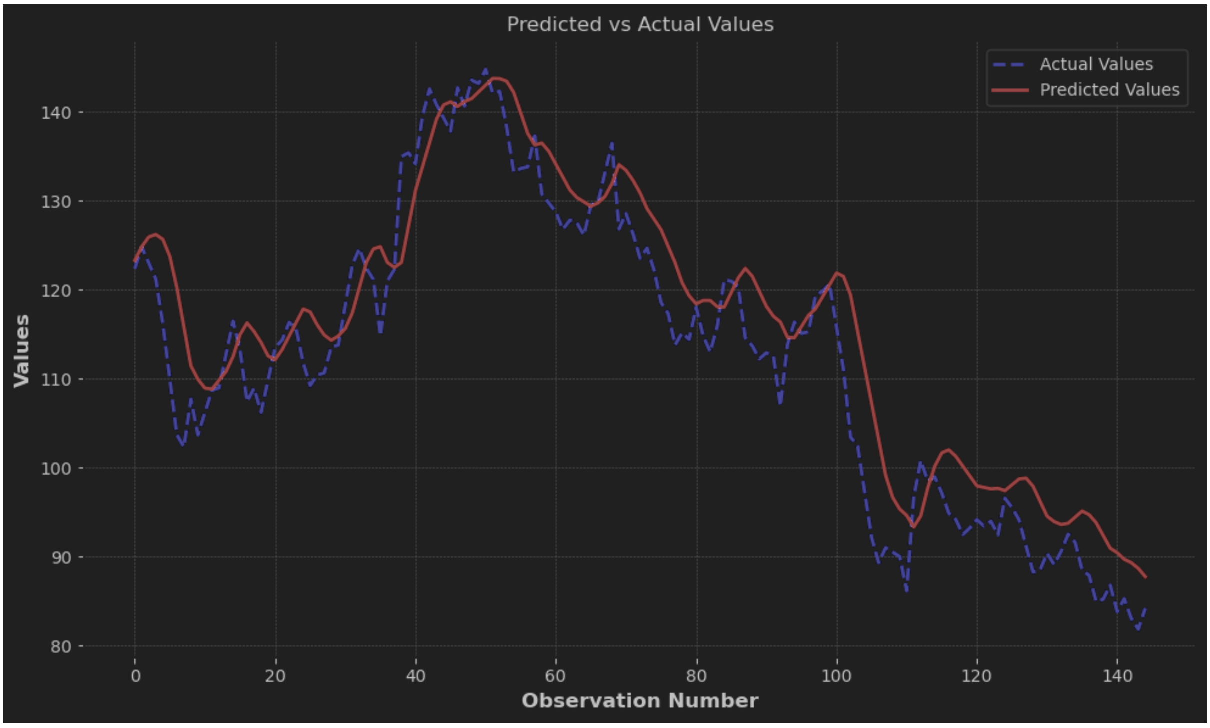
A different experiment on Gated Recurrent Unit, with same layer size like first Recurrent Neural Network, only one hidden layer



The gap between actual and predicted value is not quite much, which means there is a slight deviate between these values.

```
model_gru2 = build_custom_model(layer_type='GRU', input_shape=(30, 1), layer_sizes=[50], num_layers=3, dropout=0.3)
history1 = model_gru2.fit(X_train, y_train, epochs=epochs, batch_size=batch_size, validation_data=(X_test, y_test), shuffle=False)
Executed at 2024.10.01 19:25:31 in 18s 705ms
```

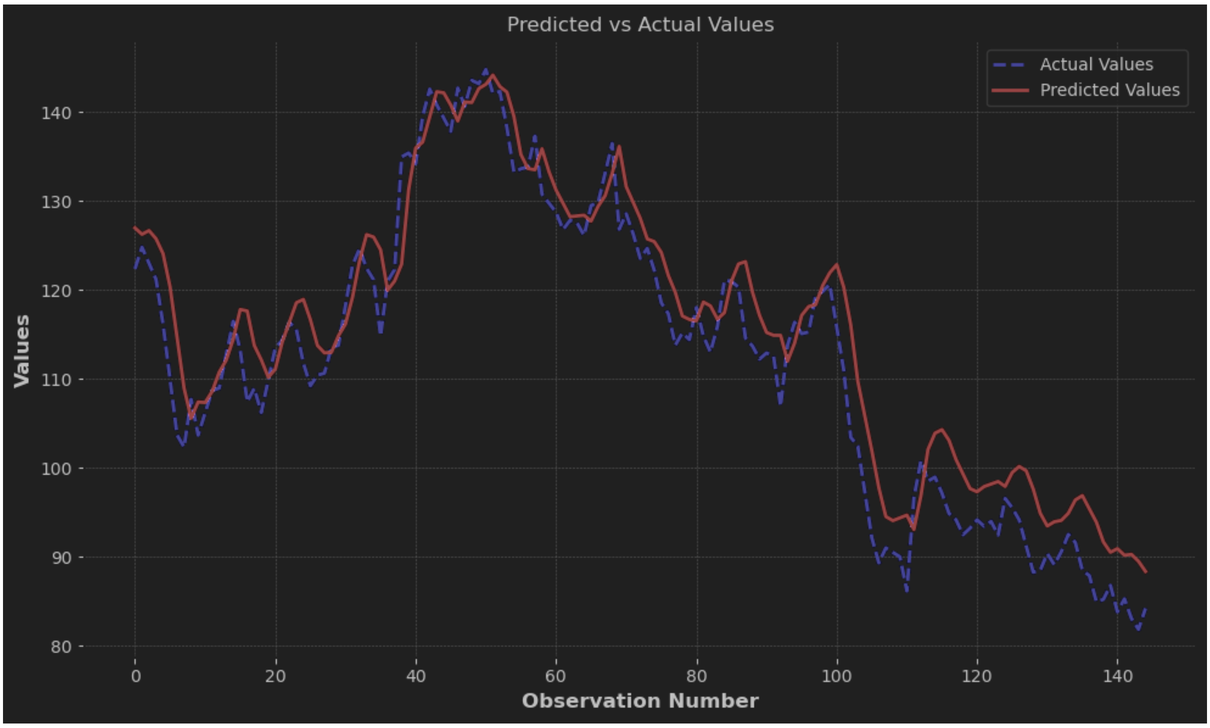
In another Gated Recurrent Unit model, there will be same numeric parameters as the second Recurrent Neural Network model.



The gap between actual and predicted values seems more than the previous Gated Recurrent Unit, and more deviate between these values.

```
model_lstm1 = build_custom_model(layer_type='LSTM', input_shape=(30, 1), layer_sizes=[200], num_layers=2, dropout=0.2)
history1 = model_lstm1.fit(X_train, y_train, epochs=epochs, batch_size=batch_size, validation_data=(X_test, y_test), shuffle=False)
```

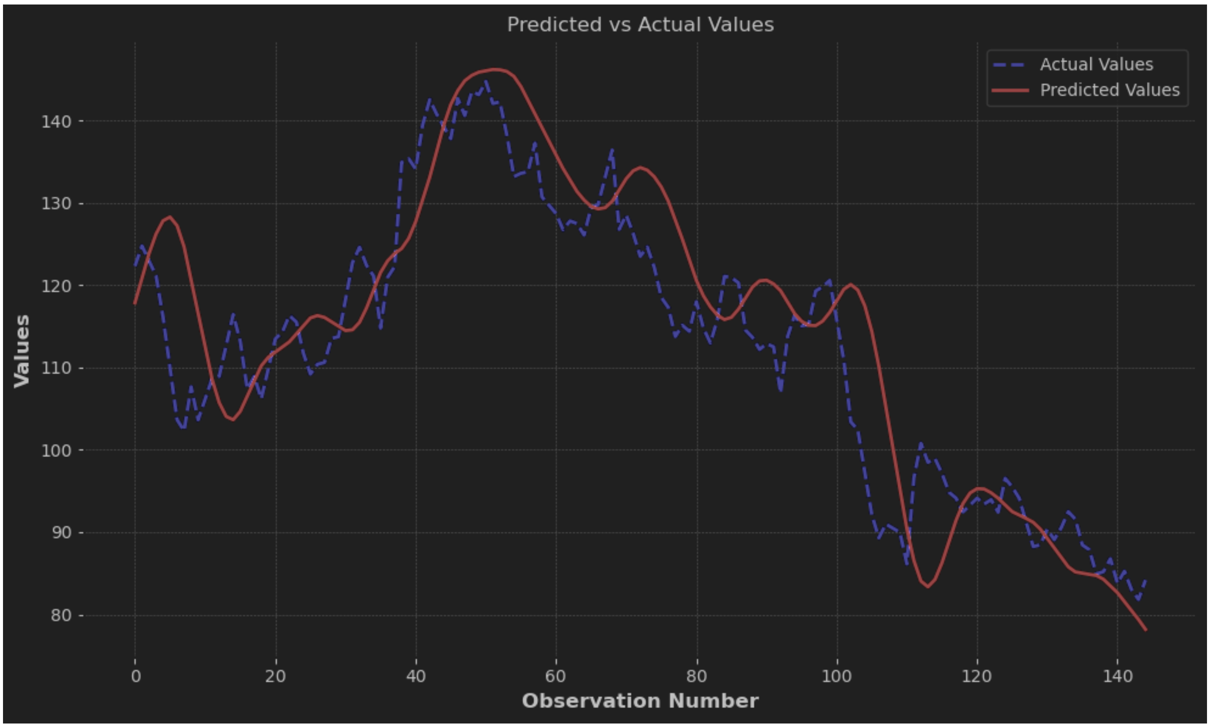
Long short term memory model with same layer size, number of hidden layers, and dropout as first Recurrent Neural Networks and Gated Recurrent Unit.



The gap between actual values and predicted seems quite near each other, although the deviate between them happens several places.

```
model_lstm2 = build_custom_model(layer_type='LSTM', input_shape=(30, 1), layer_sizes=[50], num_layers=3, dropout=0.3)
history1 = model_lstm2.fit(X_train, y_train, epochs=epochs, batch_size=batch_size, validation_data=(X_test, y_test), shuffle=False)
```

Long short term memory model with same layer size, number of hidden layers, and dropout as second Recurrent Neural Networks and Gated Recurrent Unit.



The plot shows the big gap between actual and predicted values, which means their deviate at both of them is higher than the previous LSTM model.

	Model	MSE	MAE
0	RNN1	26.380133	4.284587
1	RNN2	92.458334	7.787673
2	GRU1	25.664503	4.174906
3	GRU2	29.975000	4.354117
4	LSTM1	37.059531	4.697052
5	LSTM2	75.609520	6.635070

At the result table, Mean Square Error and Mean Absolute Error show the metrics at different model. In summary, GRU works better though have to change the layer size, number of layers and dropout. When attempting that on first and second of RNN and LSTM, it shows the significant gap among them. In GRU1, their values becomes the smallest compare with others, in contrast with RNN2.