Experimental Modifications for Deep Learning Stock Forecasting

EXPERIMENTAL MODIFICATIONS FOR DEEP LEARNING STOCK FORECASTING

1. Different number of layers of recurrent networks

Change: Modify build_lstm() and build_gru() functions to take an argument n_layers and stack layers accordingly.

```
def build_lstm(sequence_length, dropout_rate=0.1, n_layers=1):
    global model_lstm
model_lstm = Sequential()
for i in range(n_layers):
        return_seq = (i < n_layers - 1)
model_lstm.add(LSTM(50, activation=activation,
return_sequences=return_seq,
input_shape=(sequence_length, 1) if i == 0 else None))
if dropout_rate > 0:
            model_lstm.add(Dropout(dropout_rate))
model_lstm.add(Dense(1))
model_lstm.compile(optimizer=Adam(learning_rate=0.01),
loss='mse')
```

2. Different items in history_length used for training

Change: Adjust the lengths list in the loop.

```
lengths = [30, 60, 120, 250, 500, 1000]
```

3. Different activation function for each layer

Change: Pass a list of activations and assign them to layers during model construction.

```
def build_gru(sequence_length, activations=['relu', 'tanh']):
    global model_gru
model_gru = Sequential()
for i, act in
enumerate(activations):
```

Experimental Modifications for Deep Learning Stock Forecasting

```
        return_seq = (i < len(activations) - 1)
model_gru.add(GRU(50, activation=act,
return_sequences=return_seq,
input_shape=(sequence_length, 1) if i == 0 else None))
model_gru.add(Dense(1))
model_gru.compile(optimizer=Adam(learning_rate=0.01),
loss='mse')
```

4. Resetting the models after each training (training from scratch)

Change: Always reinitialize before each forecast.

Remove

```
       if phase == 0:
                if length_of_history == 1000:
```

5. With constant history length, apply variant forecasting step lengths

Change: Introduce a new list of forecast_horizons and loop over it while keeping history constant.

```
steps=10
```

6. With constant history length forecasting steps length apply variant sequence lengths in models

Change: Add an inner loop over sequence_length values.

7. Compare various normalization methods in data preprocessing phase

Change: Create a list of alternative scalers and integrate them into preprocessing.

```
from sklearn.preprocessing import MinMaxScaler, StandardScaler, RobustScaler,
MaxAbsScaler scalers = {
    "MinMax": MinMaxScaler(),
    "Standard": StandardScaler(),
    "Robust": RobustScaler(),
    "MaxAbs": MaxAbsScaler()
}
```