```python
# split data and get all input, output and number of example uses in the train
# here we cannot shuffle because we want to work with ordered past events
train_input, train_output, test_input, test_output, num_examples = sp.split(input, output, train_size, False)

# create tensor for my input and output - [batch-size, timestep, numberOfFeatures] and [batchsize, numberOfClasses]
X = tf.placeholder(tf.float32, [None, timestep, num_input])
Y = tf.placeholder(tf.float32, [None, num_classes])

# create multiple LSTMCells of same size (neurons) for each hidden layers that we want
rnn_layers = [tf.nn.rnn_cell.LSTMCell(n_neurons) for _ in range(n_hiddens)]

# create a RNN cell composed sequentially of a number of RNNCells
multi_rnn_cell = tf.nn.rnn_cell.MultiRNNCell(rnn_layers)

# 'outputs' is a tensor of shape [batch_size, timestep, numberOfClasses]
# 'state' is a N-tuple where N is the number of LSTMCells containing a tf.contrib.rnn.LSTMStateTuple for each cell
outputs, state = tf.nn.dynamic_rnn(cell=multi_rnn_cell,inputs=X,dtype=tf.float32)
```